An online study database

Version 0.63

J.M. van Schalkwyk, L.A. Hopley

January 14, 2008

Contents

1 Overview — the STrICT study

This document describes the STrICT study, the associated database (implemented in mySQL), and web-based (PHP) scripts used to access the database. This document is Copyright © J.M. van Schalkwyk 2008. Documentation and code are released under the GNU Public Licence, as shown in Section 6.

The STrICT study ('Simulation Training of Intensive Care Teams') compares conventional teaching and immersive, scenario-based (ISB) teaching of teams in a simulated ICU setting. A strength of the study is that team performance in *standardised scenarios* will be assessed before and after teaching. Because identical METI 'physiological models' will underly all scenarios to be compared, subjective effects on outcomes will be minimised.

A team consists of four participants, one registrar who might come from a variety of disciplines but is currently 'on' ICU, and three ICU nurses with whom the registrar is currently working in ICU. On each study day there will be two teams. One team will be exposed to ISB teaching for respiratory problems, and 'conventional' teaching for cardiovascular problems; the other team will have the reverse exposure. There will be twenty study days.

Each standardised scenario will be *assessed* by each of the participants. These standardised scenarios will take place before and after teaching. In addition, independent assessors will view video-tapes of the standardised scenarios.

Assessors (both local and overseas) will be provided with a DVD containing video clips of all standardised scenario for a given day, that is, eight clips per day, presented in random order. The assessors will not know whether a particular scenario was filmed before or after teaching took place. The assessors will rate the performance of teams according to a multi-dimensional rating scale. Ratings will be numeric from 1–7.

1.1 Information about participants

Details will be collected on each team, and in particular, on each participant in each team. Desired information about all participants is:

- Date of first qualification;
- Years of experience in ED;
- Years of experience in HDU;
- Years of experience in ICU;
- Years of experience in OR;

- Years of experience in other acute areas [listed, may be several]
- Previous simulation exposure: none OR:
- Previous simulation exposure: basic mannequin CPR AND/OR:
- Previous simulation exposure: full resuscitation course AND/OR:
- Previous simulation exposure: full day simulation centre course, AND/OR:
- Previous simulation exposure: multi-day simulation centre course, AND/OR:
- Previous simulation exposure: other (multiple options?)
- What is their current ICU location? (Cardiovascular ICU, HDU, DCCM, Middlemore ICU, Middlemore HDU, North Shore Hospital ICU).

Specific information desired from the registrar will be:

- What is the current specialty of the registrar? (ICU, anaesthesia, or the Emergency Department);

1.2 Rating information

There are four 'patients', which can be presented in any order on a given day. The patients are called 'Alan', 'Donald', 'Georgina' and 'Helen'. Multidimensional rating scales for behavioural performance of *teams* are identical for all patients. Assessment tools for the technical aspects of respiratory and cardiovascular patients are broadly similar, but of necessity there are small differences between the two assessment tools. These data will be recorded on paper forms immediately after completing scenarios, and the paper forms need to be captured into the database.

Each of four assessors will assess each team on each patient using the same rating scales as those used by the participants. As noted, each assessor is blinded to the order of the patients, each of which will simply be identified by the number of the patient on the relevant DVD, and which of the four patients is featured.

At present there is lack of clarity about what analysis is contemplated. Important components are perhaps:

1. The individuals' perception of how the team performed on the day;
2. Assessors' perception of team performance;
3. How the above items changed over the course of the day;

4. The study authors anticipate greater improvement in team performance on patients where prior teaching involved ISB teaching than with conventional teaching. Comparison will be made between improvement in the cardiovascular scenarios and respiratory scenarios depending on team exposure to ISB or conventional teaching.

5. How good is inter-rater reliability (? kappa).

   An initial comparison will be made for the pilot study, comprising three preliminary days (12 videos), an initial study day (8 videos), and and 9 intervention videos (29 in all). This comparison should ratify the rating scale. [Need more detail here on how 'ratification'/validation/benchmarking is to be performed].

   In this pilot study, analysis will compare not only the four independent raters, but also these raters and the unblinded team of five facilitators involved in this first teaching session.[1]

The null hypothesis is that there is no difference between conventional training and immersive scenario-based training. This hypothesis will be tested at a 5% confidence level, comparing the assessors' numeric rating of teams' performance on a multi-dimensional scale before and after teaching.[2] As a given team is assessed four times on a day (before and after each standardised CVS and respiratory patient, one teaching episode being traditional and the second being ISB), the change in performance for the 20 groups who underwent ISB on the CVS patient can be compared with that of those who had conventional training, and likewise for the respiratory patient.

Median scores for the various scenarios can be compared to determine the 'degree of difficulty' of the CVS versus respiratory patients. In addition, it will be possible to compare the improvement in performance for a given team. Such improvement can then be related to whether the team received conventional or ISB training for that type of patient.

Data within the database must be readily exportable, perhaps as a CSV table, or a set of several CSV tables.

It would seem logical to use the existing Agate database for all data entry, after suitable modification to permit entry of relevant data as outlined above and described in more detail below, and use of radio-button based scales rather than the current VAS rating scales.

[1] This initial assessment will be published as a preliminary paper.

[2] It is unclear at this stage how the multi-dimensional data will be combined, if this is to be done, and if not, what statistical compensation will be made for the multiple comparisons being used.

1.3 Behavioural & Technical Rating scales

For cardiovascular scenarios:

1. Individuals are asked to rate their team on ACLS technical dimensions (12) on a rating scale of 1–7. The last of these is a global assessment.
2. They also rate CRM in 24 dimensions. The last of these is also a global assessment.
3. They perform a final 'Subjective Global Overall Assessment' (SGOA).

Respiratory scenarios have twelve similar technical dimensions, followed by 24 identical CRM dimensions, and an SGOA.

Raters' assessment tools are identical to the above.

1.4 Timetable

We also must record who is conducting each teaching session and debriefing session with each team, as the teacher/debriefer might conceivably have a substantial impact on outcomes, and we wish to compare outcomes by teacher/debriefer.

Teaching/debriefing sessions we wish to document are:

1. An initial scenario, debriefing and assessment;
2. A second scenario, debriefing and assessment;
3. A Defibrillator skill station;
4. A CRM lecture;
5. An Airway skill station;
6. An Immersive intervention;
7. Problem-centred learning (PCL);
8. A penultimate scenario, debriefing and assessment;
9. A final scenario, debriefing and assessment.

One of the first two scenarios will be cardiovascular, the other respiratory, and likewise for the final two. The order of the rest of the day will be fixed, apart from the immersive intervention and problem-centred learning, which will be either first or second in a randomised fashion. If the immersive intervention is cardiovascular, then the PCL will be respiratory, and vice versa.

2 Database specification

2.1 Entities within the database

These are as follows:

- A *person*, who might be a registrar, nurse, teacher, or rater;
- A *study day*. Each study day is associated with two teams, and each *team* is made up of three nurses and one registrar;
- A *patient*, either a respiratory or cardiovascular patient, may be pre- or post-teaching, and if the latter may follow either conventional or ISB teaching;
- A *rating*, which may be one of 12 technical rating types, 24 CRM rating types, or an SGOA rating. A rating may be performed by any person, and may be associated with any patient. Each person apart from teachers/debriefers should perform one of each rating type on each patient.

The following tables will be created:

2.2 PERSON

The PERSON table might have the following attributes (columns)

- person (primary key: integer)
- Forename
- Surname
- Role (1=nurse, 2=registrar, 20=assessor, 10=teacher)

We will not make frantic attempts at normalising the PERSON table, as we don't anticipate continuing the database to the point where roles, surnames and so forth will alter. In addition, a single person cannot share multiple roles. However, as we will have a subsidiary table, PERSDATA which will in any case contain further information about people, we will cunningly move the Forename, Surname and Role entries to this table, allowing us to retain a degree of compatibility with the PHP code for the older (and better-normalised) AGATE database. Let's examine entries in the PERSDATA table:

2.2.1 PERSDATA

- persdata (primary key, integer)
- FirstQualified
- CurrentSpecialty (for registrars only; coded as 1=ICU, 2=Anaesthesia, 3=ED, 4=other)
- CurrentLocation (coded as 1=CV ICU, 2=CV HDU, 3=DCCM, 4=Middlemore ICU, 5=Middlemore HDU, 6=North Shore ICU)

The CurrentSpecialty will be NULL for nurses.

2.2.2 EXPERIENCE

We will also have an EXPERIENCE table which can potentially have numerous values for one particular person:[3]

- experience (primary key, integer)
- eRole (1=ED, 2=HDU, 3=ICU, 4=OR, 5=other)
- eText (role description if 'other' selected)
- eYears (number of years of experience in this role)
- Person (foreign key in PERSON table)

2.2.3 SIMEXPOSURE

Finally there is a SIMEXPOSURE table, documenting previous simulation exposure:

- simexposure (primary key, integer)
- simnature (coded nature of exposure: 1=basic mannequin CPR; 2=full resuscitation course; 3=full-day simulation; 4=multi-day simulation; 5=other)
- xText (role description if 'other' selected)
- Person (foreign key in PERSON table)

'No previous simulation exposure' will be indicated by an absent entry in the SIMEXPOSURE table, but we should also consider denormalising things by having an appropriate flag field in the PERSON table.

[3] Should we record when this 'experience' terminated?

2.3 STUDYDAY, TEAM

A study day is characterised by a date and a unique ID. The TEAM table defines a team characterised by a unique primary key, and the date on which the team participated. Team-associated data are:

- team (primary key)
- date (date stamp, the day on which the team participated)
- Registrar (reference to PERSON table)
- Nurse1 (reference to PERSON table)
- Nurse2 (reference to PERSON table)
- Nurse3 (reference to PERSON table)

2.4 PATIENT & STUDYDAY

A given patient is attached to a particular team, and may be a 'before' or 'after' patient, and either cardiovascular or respiratory. Here are the attributes:

- patient (primary key)
- Timing (first–fourth).[4]
- PatientNature (1 if cardiovascular, 2 if respiratory)
- WhichPatient (1 of 4: 1=Alan, 2=Donald, 3=Georgina, 4=Helen)
- Team (foreign key reference to TEAM)
- Teacher (foreign key: PERSON conducting the scenario)

Assessors other than the participants will be blinded as to the status of the Timing field.

In addition, individual video clips of each patient scenario for each team will be stored on a DVD (containing eight such 'tracks'[5]). We therefore need a DVD-related entity (but this is simply the study day!) and subsidiary 'track identifiers', each of which will be associated with a particular patient.

[4] Formerly we called this field 'IsAfter' (1 if before, 2 if after), but this approach doesn't allow us to determine the exact sequence of presentation of the scenarios.

[5] We've called these 'tracks' but they don't necessarily correspond to physical tracks on the DVD.

2.5 RATING

Performance of each team must be rated by each of the team participants after each patient, as well as being rated by assessors (at least four). We thus associate particular assessors and patients (the team being derived from the Team field in the PATIENT table). There are twelve technical dimensions and 24 CRM dimensions, as well as an SGOA rating. All are in the range of 1–7. Here's the RATING table:

- rating (primary key);
- Patient (foreign key reference to PATIENT table);
- Person (person doing the assessment);
- Tech1–Tech12 (integer rating, 1–7);
- Crm1–Crm24 (1–7);
- SGOA.

Note that three CRM rating scores are reversed, according to the specification of the creators of the study. This peculiarity doesn't affect our database design.

There is no limit to the number of assessors. We could consider normalising the above table further.

2.6 SQL coding of tables

In the following SQL code, we have kept things very light, with few constraints on field values. Our reasoning is that we tightly control entry of values, so extra checks are unlikely to help much. If you find this approach unacceptable, please feel free to add extra checks (as always, within the constraints of the GNU Public licence). The table coding is straightforward, with the tiny wrinkle that our PHP log-in requires several extra fields. In addition, we've varied the code slightly from the above (initial) specifications for reasons of logic, and to a lesser degree for (sensible) compatibility with the code we wrote for Agate.

2.6.1 PERSON

```
CREATE TABLE PERSON (
  person integer,
  constraint badPerson primary key (person),
  pExpiration integer,
  pLoginName varchar(16),
  pPassword varchar(32),
```

```
  pSID varchar(32),
  pValue varchar(255),
  FirstQualified integer,
  pMade timestamp
                    );
```

We've moved the Forename, Surname and PersonRole fields from the PERSON table to the PERSDATA table below, in keeping with our original Agate database. We've also moved **FirstQualified** from the PERSDATA table to here, as it can only ever have one value, and belongs here.

The additional fields required by our PHP program are:

pExpiration is a UNIX timestamp used to handle time-out if a user overstays their welcome;

pLoginName is the log-in name of a user (if appropriate);

pPassword which will accommodate an md5-encoded password;

pSID is a session ID; and

pValue is used to store 'stuff' (Details of prior page for this user).

We've also added **pMade**, a timestamp for the creation of a person entry, as a 'bookkeeping' entry.

Here's the PERSONROLE table, used below. As noted above possible values for **PersonRole** range from 1–4 (nurse, registrar, assessor, and teacher respectively) and we populate this table now.

```
CREATE TABLE PERSONROLE (
  personrole integer,
  constraint badPersonRole primary key (personrole),
  rText varchar(32)
                    );

INSERT INTO PERSONROLE (personrole, rText)
   VALUES (0, 'nobody'),
          (1, 'nurse'),
          (2, 'registrar'),
          (20, 'assessor'),
          (10, 'teacher'),
          (192, 'superuser'),
          (202, 'teacher+superuser');
```

We add the role of 'superuser' for the database administrator to use, and we have a special value of 'nobody' (who has no privileges) as a fill in for various roles (see later)!

2.6.2 PERSDATA

```
CREATE TABLE PERSDATA (
  persdata integer,
  constraint badPersdata primary key (persdata),
  pdCreated timestamp,
  Person integer,
  constraint badpdPerson foreign key (Person)
    references PERSON,
  pdForename varchar(32),
  pdSurname varchar(32),
  pdGender integer,
  PersonRole integer,
  constraint badpdPersonRole foreign key (PersonRole)
    references PERSONROLE,
  CurrentSpecialty integer,
    constraint BadpdCurrentSpecialty foreign key (CurrentSpecialty)
      references CURRENTSPECIALTY,
  CurrentLocation integer,
    constraint BadpdCurrentLocation foreign key (CurrentLocation)
      references CURRENTLOCATION
                  );
```

We toss in **pdGender** for reasons of common sense, and we've added a **pdCreated** field to allow us to determine when a particular entry was made. Gender codes are 1 for female and 2 for male.

CurrentSpecialty is 'for registrars only', coded as 1=ICU, 2=Anaesthesia, 3=ED, 4=other. **CurrentLocation** is coded as 1=CV ICU, 2=CV HDU, 3=DCCM, 4=Middlemore ICU, 5=Middlemore HDU, 6=North Shore ICU. Here are the relevant tables:

```
CREATE TABLE CURRENTSPECIALTY (
  currentspecialty integer,
  constraint badCurrentSpecialty primary key (currentspecialty),
  SpecialText varchar(32)
                  );

INSERT INTO CURRENTSPECIALTY (currentspecialty, SpecialText)
   VALUES (0, 'nil'),
          (1, 'ICU'),
          (2, 'Anaesthesia'),
          (3, 'ED'),
          (4, 'other');

CREATE TABLE CURRENTLOCATION (
  currentlocation integer,
  constraint badCurrentLocation primary key (currentlocation),
```

```
  LocationText varchar(32)
                   );

INSERT INTO CURRENTLOCATION (CurrentLocation, LocationText)
   VALUES (1, 'CV ICU'),
          (2, 'CV HDU'),
          (3, 'DCCM'),
          (4, 'Middlemore ICU'),
          (5, 'Middlemore HDU'),
          (6, 'North Shore ICU');
```

2.6.3 EXPERIENCE

```
CREATE TABLE EXPERIENCE (
  experience integer,
  constraint badExperience primary key (experience),
  eRole integer,
    constraint badexErole foreign key (eRole)
      references EROLE,
  eText varchar(32),
  eYears integer,
  Person integer,
  constraint badExperiencePerson foreign key (Person)
    references PERSON
                   );
```

The **eRole** field is one of 1=ED, 2=HDU, 3=ICU, 4=OR, 5=other. Here's the table:

```
CREATE TABLE EROLE (
  erole integer,
  constraint baderole primary key (erole),
  rText varchar(32)
                   );

INSERT INTO EROLE (erole, rText)
   VALUES (1, 'ED'),
          (2, 'HDU'),
          (3, 'ICU'),
          (4, 'OR'),
          (5, 'other');
```

2.6.4 SIMEXPOSURE

```
CREATE TABLE SIMEXPOSURE (
  simexposure integer,
    constraint badSimExposure primary key (simexposure),
```

```
  Simnature integer,
    constraint badxSimNature foreign key (Simnature)
      references SIMNATURE,
  xText varchar(32),
  Person integer,
  constraint badExposurePerson foreign key (Person)
    references PERSON
                  );
```

The value in **simnature** is one of: 1=basic mannequin CPR; 2=full resuscitation course; 3=full-day simulation; 4=multi-day simulation; 5=other. **xText** is the role description if 'other' was selected.

```
CREATE TABLE SIMNATURE (
  simnature integer,
  constraint badsimnature primary key (simnature),
  SimText varchar(32)
                  );

INSERT INTO SIMNATURE (simnature, SimText)
   VALUES (1, 'basic mannequin CPR'),
          (2, 'full resus course'),
          (3, 'full-day simulation'),
          (4, 'multi-day simulation'),
          (5, 'other');
```

2.6.5 STUDYDAY, TEAM

```
CREATE TABLE STUDYDAY (
  studyday integer,
    constraint badStudyday primary key (studyday),
  studydayDate date,
  dName varchar(16),
  dText varchar(64)
                );
```

The **dText** field is arbitrary, descriptive text which can be entered at the time of creation of the STUDYDAY table entry by an administrator. The **dName** field is a shorter label intended to be used (as the name suggests) as the *name* of the study. Initially, in versions below 0.62, we simply used the **studyday** key value as the 'number of the study' but this approach is inflexible. We have made similar changes to the TEAM table.

In creating a TEAM, it's tempting to normalise things further, creating a TEAM-MEMBER table, associating a TEAM with a PERSON in a particular role. Aww, what the heck, let's do this:

```
CREATE TABLE TEAMMEMBER (
  teammember integer,
    constraint badTeamMember primary key (teammember),
  Team integer,
    constraint badTmTeam foreign key (Team)
      references TEAM,
  Person integer,
    constraint badTmPerson foreign key (Person)
      references PERSON,
  tmRole integer        );
```

At present we have two possible values for tmRole (2=registrar, 1= nurse), but won't create a subsidiary TMROLE table. We have no constraints in the database on how many people make up a team, simply building this constraint into the user interface. Now the TEAM table is a lot simpler.[6]

```
CREATE TABLE TEAM (
  team integer,
    constraint badTeam primary key (team),
  Studyday integer,
    constraint badtStudyday foreign key (Studyday)
      references STUDYDAY,
  ISBnature integer,
  tName varchar(16),
  tNote varchar(64)
                   );
```

We add a **tNote** field to permit association of a text comment with a given team. As for the STUDYDAY table, we also have a shorter (**tName**) field intended to replace use of the **team** primary key as a label for the team. So it is now possible (from ver 0.62) to label the pilot teams as teams A–F (or whatever) and then start labelling teams with numbers, so the team with a primary key value of 7 can have a **tName** of, say, '1'.

In version 0.61, we added the **ISBnature** field to identify whether the immersive scenario-based teaching a team received was cardiovascular or airway-related.[7] The value is 1 for airway-related ISB scenarios, and 0 for CVS ISB scenarios.

We also note that each team will be taught by up to nine potentially different teachers[8] during the course of a day-long assessment. Here's a list of roles:

[6] We must however insert a uTeammember entry in the UIDS table below!

[7] Previously, this was fixed, the first team of the day receiving airway ISB and the second CVS, but clearly this approach might permit the assessor to become unblinded.

[8] Although often several roles will be shared by the same teacher.

```
CREATE TABLE TEACHROLE (
  teachrole integer,
    constraint badTeacherRole primary key (teachrole),
  trText varchar(32)   );

INSERT INTO TEACHROLE (teachrole, trText)
  VALUES (2, 'Debrief + assessment 1'),
         (3, 'Debrief + assessment 2'),
         (4,'Defibrillator skill station'),
         (5,'CRM lecture'),
         (6,'Airway skill station'),
         (7,'Immersive intervention 1'),
         (8,'Immersive intervention 2'),
         (9,'Problem-centred learning'),
         (10,'Debrief + assessment 3'),
         (11,'Debrief + assessment 4'),
         (1,'Coordinator');
```

We added the role of 'coordinator' in version 0.62. Let's next associate a particular team with a teacher in a particular role:

```
CREATE TABLE TEACHING (
  teaching integer,
    constraint badTeaching primary key (teaching),
  Team integer,
    constraint badTchTeam foreign key (Team)
      references TEAM,
  Teacher integer,
    constraint badTchPerson foreign key (Teacher)
      references PERSON,
  Teachrole integer,
    constraint badTchTeachrole foreign key (Teachrole)
      references TEACHROLE
                    );
```

The above table flexibly allows us to associate teachers with a team in a particular role; constraints on how many teachers can take a particular role for a given team will be imposed by the PHP code, not within the database.

2.6.6 PATIENT

```
CREATE TABLE PATIENT (
  patient integer,
    constraint badPatient primary key (patient),
  Timing integer,
  WhichPatient integer,
  Team integer,
```

```
  constraint badPatientTeam foreign key (Team)
    references TEAM,
  pTLC varchar(3),
  frozen integer DEFAULT 0
                    );
```

Each PATIENT scenario will be recorded and placed on a DVD as a particular 'track'. Each track will have a unique, generated track ID (**pTLC**). This will consist of three case-independent alphabetical characters (e.g. AQE) which will be unique for that track. The assessor will be required to enter this character sequence before they can assess the patient scenario on DVD, in order to prevent erroneous assessment of the wrong scenario!

[FIX ME] We have revised the database structure to include to the more flexible TEACHING table.

Timing can take on values of 1–4 (first to fourth); **WhichPatient** is in the range of 1–4 (1=Alan, 2=Donald, 3=Georgina, 4=Helen). We will permit an initial NULL value in the Timing field, but the WhichPatient value will be preset from 1–4 when a PATIENT entry is created.

The administrator will be able to alter the sequence of patients for a particular group on a particular day (Timing field value) at some point he/she will have to 'freeze' the track arrangement, to prevent somebody at a later stage fiddling and messing up the sequence. This is the point of the **frozen** field. When this value is set, we will disallow further alterations.

We will only create a table for **WhichPatient**, as the other two fields are trivial.

```
CREATE TABLE WHICHPATIENT (
  whichpatient integer,
  constraint badwhichpatient primary key (whichpatient),
  PatientNature integer,
  PatientName varchar(32)
                    );

INSERT INTO WHICHPATIENT (whichpatient, PatientNature, PatientName)
   VALUES (1, 1, 'Alan Gordon'),
          (2, 2, 'Donald Johnson'),
          (3, 1, 'Georgina Battenburg'),
          (4, 2, 'Helen Anderson'),
          (5, 3, 'Peter Smeaton'),
          (6, 3, 'Joan Digger'),
          (7, 3, 'Julia Robertson'),
          (8, 4, 'Frederica Fetherstone'),
          (9, 4, 'Stephan Irwin'),
          (10, 4,'Victor Hemingway');
```

PatientNature is 1 if cardiovascular, 2 if respiratory. The six added patients (IDs 5–10) represent ISB patients, who will also receive assessments in the initial

evaluation of the STrICT study. They were added in version 0.62. PatientNature codes of 3 and 4 correspond to cardiovascular and airway ISB scenarios, respectively.

2.6.7 RATING

First, let's create a small table which contains the text describing each rating item. We'll call it RORDER:

```
CREATE TABLE RORDER (
  rorder integer,
    constraint BadRorder primary key (rorder),
  ratText varchar(255)
                   );
```

Later on we'll find out how to import comma-delimited (CSV-style) data directly into this table.

Denormalising things by having multiple *Tech* and *Crm* fields in the RATING table is most unappetising. We therefore create a small subsidiary table which contains these Tech/Crm values:

```
CREATE TABLE ONERATING (
  onerating integer,
    constraint BadOneRating primary key (onerating),
  Rorder integer,
    constraint BadOrRorder foreign key (Rorder)
      references RORDER,
  rType integer,
  rValue integer,
  Rating integer,
    constraint BadOrRating foreign key (Rating)
      references RATING
                    );
```

The above table makes things more flexible at the relatively minor expense of some table joins and CPU time. The actual rated value is stored in **rValue**. Values in **rType** are hard coded without any subsidiary table reference as 1=Tech and 2=CRM. The **Rorder** field refers to the order (and text) of these fields and can be 51–62 for Tech (cardiovascular), 1–12 for Tech (respiratory), and 101–124 for CRM.[9] We cheat a little and insert our 'overall assessment' (SGOA) as item 125. Now for the TABLE specification:

[9] We add this extra denormalisation to prevent silly mistakes.

```
CREATE TABLE RATING (
  rating integer,
    constraint badRating primary key (rating),
  Patient integer,
    constraint badRatingPatient foreign key (Patient)
      references PATIENT,
  Assessor integer,
  constraint badAssessor foreign key (Assessor)
    references PERSON,
  raText varchar(255)
                   );
```

2.6.8 UIDS

In this final table, we create seeds for generating unique IDs for all of the other tables. The SQL standard doesn't directly provide automatic key generation, and although dialects do provide thinks like auto-incrementing fields or permit user-defined functions to fill the gap we will not use these facilities as they vary between databases. We will rather take keys as we need them from the UIDS table. Each new primary key for a new table entry will be fetched from the UIDS table:

```
CREATE TABLE UIDS (
  uids integer,
   constraint badUids primary key (uids),
  uPerson integer,
  uPersdata integer,
  uExperience integer,
  uSimexposure integer,
  uTeam integer,
  uTeammember integer,
  uTeaching integer,
  uStudyday integer,
  uPatient integer,
  uOnerating integer,
  uRating integer,
  auxPerson varchar(32),
  auxPersdata varchar(32),
  auxExperience varchar(32),
  auxSimexposure varchar(32),
  auxTeam varchar(32),
  auxTeammember varchar(32),
  auxTeaching varchar(32),
  auxStudyday varchar(32),
  auxPatient varchar(32),
  auxOnerating varchar(32),
  auxRating varchar(32)
                 );
```

```
INSERT INTO UIDS (uids, uPerson, uPersdata, uExperience,
                  uSimexposure, uTeam, uTeammember,
                  uStudyday, uPatient,
                  uOnerating, uRating, uTeaching )
   VALUES        (1, 1000,    1000,      1000,
                  1000,         1,     1000,
                  1,    1000,
                  1000,       1000,    1000);

INSERT INTO PERSON (person, pLoginName, pPassword)
   VALUES          (1, 'Superuser', 'PasswordGoesHere');

INSERT INTO PERSDATA (persdata, Person, pdForename, pdSurname,
                      PersonRole)
   VALUES            (1, 1, 'Mr', 'Superuser',
                      192);
```

The last few statements create a single super-user, who will be able to log on with the username 'Superuser', and the relevant password. Note that it will be smart to replace the text 'PasswordGoesHere' with a 32-character md5 encrypted password![10]

We don't have key seed fields for the PERSONROLE, CURRENTSPECIALTY, CURRENTLOCATION, EROLE, SIMNATURE and WHICHPATIENT TABLES, as these tables will not ordinarily acquire new rows.[11]

The process of updating ('auto-incrementing') these source fields is rather tricky, and database 'features' designed to prevent errors often get in our way [TO EXPAND, See FetchKey fx below]! Problems only really arise where several users are concurrently accessing the same field in UIDS, but we must account for this eventuality. The accessory fields beginning with 'aux' are used to accommodate this necessity — they start of as NULL fields.

[10] As things stand, the stored password in the database is not actually md5 encrypted, so we need to fix this, and put relevant md5 javascript on the user side. Of course, this isn't as good as a secure connection, but it's better than having unencrypted passwords visible within the database!

[11] They would have to be manually updated within the database.

3 Online data entry

The following sections describe in detail a complete user interface written in PHP. There are two types of user, the ‘superuser’ who has complete control, and the ability to enter new teams, people and patient scenarios, as well as viewing results; and the ordinary user (assessor) who can only enter ratings for a particular patient scenario presented to him/her on a DVD.

3.1 Preliminary scripts

3.1.1 strict_GLOBALS.php

The following are a few globals (mainly constants) required by our program.

```
# 1. time-out before forcing log-off:
define ( 'strict_TIMEOUT', 60*60);    # 1 hour
define ( 'COOKIE_EXTRA_WAIT', 2000); # extra, before cookie expiry
# [database expires 1st, so can see and say "You have timed out"]

#2. useful constants:
define ( 'SHOW_USER', 1);
define ( 'MAX_TEXT_LENGTH', 255); # peculiar to mySQL

#3. Used to test status of person logging on:
define ( 'strict_ADMINISTRATOR_MIN', 128);
define ( 'strict_ASSESSOR_MIN', 19);
define ( 'strict_ASSESSOR_MAX', 63);

define ( 'ADMINISTRATOR', 192);
define ( 'ASSESSOR', 20);
define ( 'EVERYONE', 1000);
define ( 'NOBODY', 0);

#4. Convenient globals:
$CONNECTIONPATH = '../../STrICT_database_connect.php';
$handDB = '';
$USERKEY = '';
$DEBUGGING = 0;

#5. nasty hacks:
 $CLUMSY_INDEX=0;
 $CLUMSY_INDEX2=0;
```

3.2 Core code

3.2.1 strict_database_connect.php

Here's a simple script to allow connection to the database:

```
function STrICT_database_connect ()
{
 GLOBAL $DEBUGGING;
 $handDB = @mysql_pconnect ("localhost",
                        "anaes2_sysadmin",
                        "PASSWORDHERE") or die
   ('I cannot connect to the database because: ' . mysql_error());
 if ($DEBUGGING)
    { print "\n You <i>did</i> manage to connect to the database.";
    };
 if (! mysql_select_db ("anaes2_STrICT", $handDB))
    { die (" You didn't manage to connect to STrICT.");
    } else
    { if ($DEBUGGING)
         { print " You are connected to STrICT.";
         };
    };
 return ($handDB);
}
```

The above password script must be placed in a directory *closer to the root* those directly accessible via the web, enhancing security. See usage in the log-in script!

3.2.2 InitialLogon.php

The following script can either be accessed directly, or from the form contained in the HTML file *login.htm*. In the former case, the POST variable *username* will not be set in which case we simply direct the user to the *login.htm* file! If the submitted parameters (username and password) are garbage, we do the same. Only after we've sanitised these values do we try to connect to the database, after which we log on (if we can) using the *processLogon* function.

```
 require_once('ValidFx.php'); # our login validation script

 if (! isset($_POST['username']) )
   { Force_Html_InitialLogon(''); # go to login.htm page!
     exit();
   };

  $username = $_POST['username'];
```

```
  $pswd = $_POST['pswd'];
  if (   (strlen($username) > 32)
      || (strlen($pswd) > 32)
     )
   { Force_Html_InitialLogon(
      'Max password, username length 32 characters');
     exit();
   };
  if (   (strlen($username) < 4)
      || (strlen($pswd) < 6)
     )
   { Force_Html_InitialLogon('Username or password too short.');
     exit();
   }; # prevent trickery

  Sanitise($username);
  Sanitise($pswd);
  $handDB = STrICT_database_connect();

 $OLDPAGE = '';
  list($USERKEY, $OLDPAGE) =
      processLogon($username, $pswd, $handDB);
  if (! $USERKEY)
   { Force_Html_InitialLogon('');
     exit();
   };

# Determine user status:
 $qry = "SELECT MAX(PersonRole) AS userStatus FROM PERSDATA
         WHERE Person = $USERKEY";
 list($userstatus) = GetSQL($handDB, $qry, 'get user status');
# print "<br>Login user status: $userstatus for user $USERKEY";

  $success =
 "USER=\"$username\"|STATUS=\"$userstatus\"|OLDPAGE=\"$OLDPAGE\"";
  # clumsy duplication of validate_login (ValidFx.php):
  # print ("Welcome to STrICT, $username. Successful log on.<hr>");

$LOGGED_IN = 1;
require ('mainpage.php');
```

If everything worked according to plan, we set the global LOGGED_IN variable to 1 and load the main page (Section 3.7). Here's that processLogon function:

processLogon

```
function processLogon($username, $pswd, $handDB)
{
```

```
  # GLOBAL $OLDPAGE; # yuk.
  # we are now definitely connected so simply check user/pwd:
  # (later we will set things up to store pPassword as md5)
  $qry = "SELECT person, pValue FROM PERSON WHERE
            pLoginName = '$username' AND
            pPassword = '$pswd'"; # later need md5 !
  $r = GetSQL($handDB, $qry, 'get user,page');
  $userkey = $r[0];
  if (! $userkey)
     { return(0); # fail
     };

  $expiry = time() + strict_TIMEOUT;
  # here, generate SID
  $t = microtime();
  $SID = substr(md5("$userkey$t"), 0, 30); # make a hash
  # JavaScript MD5 is here: http://pajhome.org.uk/crypt/md5/

  $qry = "UPDATE PERSON
            SET pSID = '$SID',
                pExpiration = $expiry
            WHERE person = $userkey";
  $handleQ = mysql_query($qry, $handDB);
  if (! $handleQ) # hmm.
     { print "<br>Update error: " . mysql_error();
       return(0);
     };
  # and store SID in a cookie!
  setcookie('strict_SID', $SID, $expiry+COOKIE_EXTRA_WAIT);
  return ($r); # success.
}
```

Force_Html_InitialLogon

All the following does is read the login file. It would be possible to fancy things up a bit, for example by taking the submitted message (which at present is simply discarded) and writing it within login.htm.

```
function Force_Html_InitialLogon ($msg)
{ # fancy dressing up might go here #
  # print($msg);
  readfile('login.htm');
};
```

3.2.3 HTML file: *login.htm*

Here's *login.htm* which is a swanky login screen (well, almost swanky):

```
<head><title>Log in to STrICT</title>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>
<body>
<div align="center">
<h2>Log in to STrICT</h2>

<form method="post" action="InitialLogon.php">
<table>
<tr><td>User Name:
</td><td colspan="2"> <input type="text" name="username" size="16" />
</td></tr>
<tr><td>Password:</td><td>
  <input type="password" name="pswd" size="16">
</td>
<td><input type="submit" value="Login"></td></tr>
</table>
</form>

<p><img width="400" height="380" src="images/biglogo.png">
</div>
</body>
```

We do not include the stated PNG image here as a UUencoded file as it's a bit bulky.

3.3 Creating the SQL database

Here we will create a combined HTML/PHP script called *database_create.php*. To create the database, ensure the file *strict.sql* is in the correct location specified below, and then simply access the file *database_create.php* with your web browser. This is a 'one-off' page which will do nothing once the database has been created. In fact, it might be wise to completely remove this page from our server once we've made the database!

```
<head><title>Create the STrICT database</title>
</head><body>

<h2>Create a database</h2>
<p>Here we read a SQL script file, parse it, and submit the CREATE
statements contained within to mySQL.
We will use this facility to create an entire database!

<p>First, let's connect to the database.
We use the STrICT_database_connect script, which must
be installed in the correct directory.
<?php
 GLOBAL $handDB;
 $DEBUGGING = 1; # 0=off.
 require_once('strict_GLOBALS.php');
 require_once($CONNECTIONPATH);
 $handDB = STrICT_database_connect();
 if (! is_null($handDB) )
    { print "<p><b> Connected to STrICT database...</b>";
    } else
    { print "<p> Database connection *FAILED*";
    };
?>

<p>We now have an open connection to the database,
with a handle in <i>$handDB</i>.
We'll test to see whether the database creation script
has already been run using a function
called <b>is_strict_populated</b>,
which checks for the existence of the UIDS table,
and if the table doesn't exist,
we run the whole creation script.
<?php
function is_strict_populated ($handDB)
{
 $query = "SELECT uids FROM UIDS WHERE uids = 1";
 $ok = mysql_query( $query,
                    $handDB);
 return($ok);
```

```
}

function batch_sql_statements($handDB, $fname)
{
 GLOBAL $DEBUGGING;
 $fdata = file($fname);         # slurp in array of lines
 # here process lines as per our usual parser:
 # In the call to FetchNextLine, $query is by reference.
 $query = '';
 $linecount = 0;
 while (FetchNextLine ($query, $fdata))
   { $linecount ++;
     if ($DEBUGGING)
        { print ("<br> Debugging: $query");
        } else
        { print '.';
        };
     if (! mysql_query( $query, $handDB) )
        { print ( "\n Error during SQL execution: " .
            mysql_error() );
          return 0; # fail
        };
     $query = '';
   };
 print ("<br> Number of lines submitted = $linecount");
 return (1); # success
}

function FetchNextLine(&$query, $fdata)
{ STATIC $LINE; # on first pass made zero!
  while ($LINE < count($fdata))
    { $txt = $fdata[$LINE]; # get next line
      $LINE ++;
      if (! preg_match ( '/^--/', $txt) ) # if not comment
         { $query = $query . $txt;
           if (preg_match ( '/\;\s*$/', $txt)) # terminal ';'?
              { return (1);
    };   };   };
  $LINE = 0; # in case re-invoke FetchNextLine !!
  return 0; # no more lines
}
?>

The relevant SQL initialisation file is called
<i>strict.sql</i>, and must be present in
the sql subdirectory of the current directory,
or creation will fail.
```

```
<?php
 if (! is_null($handDB) ) # provided connected
    { if (! is_strict_populated($handDB))
         { if (batch_sql_statements($handDB, 'sql/strict.sql'))
              { print "<p> SUCCESS! STrICT database created.";
                print "<p> <a href='InitialLogon.php'>GO THERE!</a>";
                # here should commit!
                # mysql_query( 'COMMIT;', $handDB);
                # but mySQL auto-commits by default (ugh)!
              } else
              { print "<p> *ERROR*.
                           Failed to create STrICT database!";
              };
         } else
         { print "<p> STrICT database already exists (Duh)!";
         };
     # mysql_close($handDB); # removed: keep database open...
    };
?>
```

The above should be largely self-explanatory.

3.4 Read a CSV file

It might also be useful to be able to read pre-prepared data into our database. Rather than having a whole lot of INSERT statements, we'll create the facility to import data from a common CSV file, which is readable (and writeable) using Excel and a whole lot of other applications besides!

We will continue the above HTML code and read a CSV file into our database. This file *RORDER.csv* will be used to populate the RORDER table.

We will write PHP scripts to do just this, but two cautions:

1. The CSV files will temporarily be stored in a 'csv' subdirectory of the same directory as the scripts. This is generally *not* a good idea, as we then have to remember to delete the files, which somebody might otherwise pick up off the 'net. We should avoid putting sensitive data into such files, anyway.

2. Not all CSV files are compatible, as there is no formal standard. The biggest culprit actually seems to be MS Excel!

With the above in mind, let's get writing. First we'll write the main routine to read in and parse a CSV file, then we'll fill in the details. The function ReadCsv will take the name of a CSV file (*without* the CSV suffix), and read the data into the table with the same name! We will adhere to our clumsy convention that all table names are UPPERCASE (Although most databases don't care, mySQL does, at least on Unix-like systems).

3.4.1 CSV conventions

The CSV file will have a very particular structure. Here are the rules:

1. In any one line, data items are separated by commas.
2. Any line *starting* with a double percentage sign (%%) will be ignored as it is a comment line. (If you don't like this rule, then don't insert comments into the CSV file. There is no CSV 'comment standard'. We have introduced this ability because our DogWagger utility generates comment lines for annotation of most files.
3. Any line made up solely of whitespace is regarded as a 'null line' and ignored.
4. There is a single CSV header line, and this is the first line which is:
   - Not a 'null line'; and
   - Not a comment.

   This header line specifies column names, which must be *identical* to column names in the actual database (including case sensitivity). Data import will *fail* if the column names are not identical.
5. Column names are separated by commas, as with data items. There is however extra information contained within the header line. If the column is numeric, then just the name is provided. If the name is of type varchar, then the column name is itself contained within 'single quotes'! For example, the column name *pSID* which is of type *varchar* will be rendered 'pSID'. Similarly if the column is a date, time or timestamp, the relevant keyword will precede the column name in quotes. For example, a DATE column called myDate will be rendered as follows in the header line:

   ```
   DATE 'myDate',
   ```

6. In populating the CSV file, *you* must provide the unique primary key value. In addition, because for most tables we will generate incremental keys, it's your responsibility to ensure that the key values you supply will not conflict with subsequent generated keys (or you must modify the following code to insert generated keys). This shows that our rationale for using the CSV option is to *initially* populate tables, and not use it for subsequent importing of data!

7. No backticks are permitted in data, nor do we allow double quotes within data. (Single quotes will be duplicated to prevent trouble with INSERT statements in SQL).

8. A terminal comma at the end of a line is permitted but ignored.

9. Line feeds should not be used within items, but if they are necessary render them as usual in C (backslash followed by lower case n).

10. If you must use a comma *within* a data item, you must precede it with a backslash;

11. Each data item may be encased in double quotes, but this is not required.

12. Leading and trailing spaces in CSV data fields are suppressed, so if these are required, then the double quote character must be used at the " start and end " of the field to prevent blank suppression. The double quote itself will be ignored.

3.4.2 The main code

Here's the code which lists the CSV files to read in, and actually reads them. We are already connected to the database.

```
<?php
 require_once('csv_read.php');
 $DEBUGGING = 1; # 0=off.

 $csvfiles = array ('RORDER');
 $TOTALLINESREAD = 0;
 $errcount = 0;
 mysql_query('START TRANSACTION', $handDB);

 foreach ($csvfiles as $thisfile)
   { $errcount += ReadCsv($thisfile, $handDB);
   };

 if ($errcount > 0)
    { # rollback if error(s) occurred:
      mysql_query('ROLLBACK', $handDB);
      print ("<br> ***************ERROR**************** <br>
      There was/were $errcount error(s)!
      <br>Database rolled back. NOT saved!
              <br> ***********************************");
    } else
    { print ("<br>Successfully read in $TOTALLINESREAD lines
```

```
               from the files: ");
      print_r($csvfiles);
    }
   mysql_close($handDB);
?>
</body>
```

3.4.3 csv_read.php

Now the principal function, ReadCsv:

```
function ReadCsv ($csvname, $handDB)
{
  GLOBAL $DEBUGGING;
  $errcount = 0;
  $CSVFILE = "csv/$csvname.csv";
  if (! @stat($CSVFILE)) # if not exists
     { print ("<br>File not found: $CSVFILE");
       return (1);      # signal 1 error
     };
  $csvlines = file($CSVFILE); # get all lines
  print ( "<br><br><hr>" );
  print ( "<br>--File is &lt;$CSVFILE&gt;, line count " .
             count($csvlines) );

  $ishead = 1;
  $headarray= '';
  $thisline='';
  $LINE = 0;

  while( $LINE < count($csvlines) )  # until EOF
    {
      $thisline = $csvlines[$LINE];
      # trim leading and trailing whitespace
      $matches = '';
      preg_match ( '/^\s*(.*?),?\s*$/', $thisline, $matches);
      # we also remove terminal comma
      $thisline = $matches[1]; # get $1.
      # (or simply use ltrim, rtrim)
      if ($DEBUGGING)
         { print ("<br>:-- $thisline");
         };

      # if length substantial and not a comment:
      if (   ( strlen($thisline) > 1)
          && (! preg_match ( '/^%%/', $thisline ) ) #slow
         )
         { # if the header line
```

```
           if ($ishead)
              { $headarray = explode(',', $thisline);
                $ishead = 0; # header done
                if ($DEBUGGING)
                   { print ( '<br>--HEADER:' );
                     print_r ( $headarray );
                   };
              } else
              { # otherwise, split data, format and save
                $errcount += SaveCsvLine(strtoupper($csvname),
                             $headarray,
                             $thisline,
                             $handDB);
              };
         };
      $LINE ++; # bump line
    }; # end "while $LINE < count.."

  if ($errcount > 0)
     { print ("<br>There were $errcount insertion errors
          in parsing $CSVFILE<br><br>");
       return ($errcount);
     };
  return 0; # oddly enough, success!
}
```

The variable $TOTALLINESREAD is declared as global in SaveCsvLine and incremented there for each successful line inserted. Note that the header array is exploded willy nilly, so you're likely to get duff SQL code if the format of this line isn't exactly correct! (Another good reason not to allow users to submit such CSV files). Here's the meaty routine to parse lines and insert them as database rows:

```
function SaveCsvLine($TABLENAME, $headarray, $thisline, $handDB)
{
  GLOBAL $DEBUGGING;
  GLOBAL $TOTALLINESREAD;

  $cols = count($headarray);
  if ($cols < 1)
     { print ( "<br>Column has no length for table $TABLENAME" );
       return 1; # fail
     };

  # first fix up format of thisline:
  #   convert \, to a temporary backtick
  #   We will convert this back to comma later(FormatOneItem)
```

```
  $thisline = preg_replace( '/\\\\,/', '', $thisline);
  # split up the line at each comma:
  $itemarray = explode (',', $thisline);
  if ( count($itemarray) != $cols)
     { $c = count($itemarray);
       print ( "<br><b>Error</b>--
 Mismatch between column count($cols) and line($thisline)[$c]" );
       print_r ( $itemarray );
       return (1); # fail.
     };
  if ($DEBUGGING)
     { print ( "<br>--line data:" );
       print_r ( $itemarray );
     };

  $left = '';
  $right = '';
  $i = 0;
  while ($i < $cols)
    { list($col, $val) = FormatOneItem($headarray[$i],
                                       $itemarray[$i]);
      $left  .= "$col,";
      $right .= "$val,";
      $i ++;
    };
  # remove final commas:
  $left  = rtrim ($left, ',');
  $right = rtrim ($right, ',');
  $qry = "INSERT INTO $TABLENAME ($left) VALUES ($right)";
  if ($DEBUGGING)
     { print ( "<br>--QUERY: $qry" );
     };
  if (! mysql_query( $qry, $handDB ) )
     { print ( "<br> ** SQL ERROR:" . mysql_error() );
       return 1; # fail
     };
  $TOTALLINESREAD ++; # bump count of successful inserts
  return 0; # success
}
```

A subsidiary routine, FormatOneItem formats a single data item:

```
function FormatOneItem ($col, $itm)
{
  $retval;  # return array of 2 elements
  $retval[0] = $col; # default.

  $itm = preg_replace ( '/\'/', ',', $itm );
```

```
    # restore commas
  $itm = preg_replace ( '/\\n/', "\n", $itm);
    # fix carriage returns
  $itm = preg_replace ( "/'/", "''", $itm);
    # duplicate single quotes

  $mtch;
  if (preg_match ( '/^\s*"(.*)"\s*$/', $itm, $mtch) )
    #encasing quotes
    { $itm = $mtch[1]; # get $1
    } else
    { $itm = rtrim($itm);
      $itm = ltrim($itm);
    };
  if (strlen($itm) < 1)
    { $itm = 'NULL';
    };
  $retval[1] = $itm; #default return value

  # next pull out column name:
  # NOTE: column name must NOT contain double quotes!
  if (! preg_match ( "/\s*(\w*)\s*'(.+)'\s*/", $col, $mtch ) )
  # [check the above line in view of the double quotes ???]
    { # if no match, must be integer, simply return defaults:
      return ($retval);
    };

  $nature = $mtch[1]; # $1, can be null
  $retval[0] = $mtch[2];  # column name is $2
  if ($retval[1] == 'NULL')
    { return ($retval);
    };

  $retval[1] = "$nature '$itm'";
  # an example is "DATE '2006-01-03'"
  # simpler than complex switch/case stmt!
  return ($retval); # return both values.
}
```

This concludes our CSV reader script.

3.5 Validate user: ValidFx.php

The following script contains the function *validate_login* which should be invoked at the start of all PHP scripts which wish to access STrICT. This function checks that the current user is actually logged on, and if so, permits access to the relevant page. If the user *isn't* logged on, then a log-in screen is popped up. Do not confuse this with the initial log-on performed by the script *InitialLogon.php*. The reason why we need to validate the user for *every page* is that HTML is stateless — we need to identify a particular user by comparing a cookie stored on their machine with a value in the database. This value is the session ID (SID).

Once the invoking page has said require_once (ValidFx.php), it can actually invoke *validate_login*. We submit the parameter value SHOW_USER to this function, although at present this value isn't checked or used. The script sets the global $USERKEY to the user primary key from the SQL database table PERSON, and $handDB becomes an open handle on the database. It's a good idea to set the variable $THISPAGE prior to calling routines from ValidFx.php, although a null value (the default) is also acceptable. Here's the initial code:

```
header ('Cache-control: no-cache' );
require('strict_GLOBALS.php');
require('ancillary.php');
require($CONNECTIONPATH);
```

3.5.1 validate_login

Now for the function that actually performs the log in validation. If an SID already exists for this user (stored in a cookie, and checked against the database) then validate_login succeeds, but otherwise we pop up the file *login.htm*. On success, we not only provide a handle to the database, but we also provide information about the user (including their status), and the last page they visited!

```
function validate_login ($showuser)
{
  GLOBAL $handDB;
  GLOBAL $USERKEY;
  GLOBAL $THISPAGE;

 #1. connect to database
 $handDB = STrICT_database_connect();
 if (! $handDB)
    { mysql_close($handDB);
      print ( "\n Failed to connect to database!" );
            # here have rescue HTML code:
      readfile ('rescue.htm');
```

```
      return(0);
    };

#2. retrieve cookie. If null, present NEW login screen
 $SID = $_COOKIE['strict_SID'];
 if (strlen($SID) < 1)
    { mysql_close($handDB);
      ForceLogin('Welcome to STrICT!');
      return(0);
    };

#3. If fail, login
 $qry = "SELECT person, pExpiration, pLoginName, pValue
            FROM PERSON WHERE pSID = '$SID'";
 $handQ = mysql_query($qry, $handDB);
  if (! is_resource($handQ))
    { # debug:
      print ( "<br>DEBUG: SQL error at login:" . mysql_error() );
    };
 if (! mysql_num_rows($handQ))
    { # force cookie to evaporate:
      mysql_close($handDB);
      setcookie('strict_SID', '', time()-100);
      ForceLogin('No match for cookie');
      return(0);
    };

#4. if cookie stale, present login
  $row = mysql_fetch_assoc($handQ);
  $exptime = $row['pExpiration']; # $row is associative array
  $now = time();
  if ($exptime < $now)
     { # here force cookie to evaporate:
       mysql_close($handDB);
       setcookie('strict_SID', '', time()-100);
 # print ("<br>Debug: exp time is &lt;$exptime&gt; TIME is $now");
       print ("<br>Session has expired<br>");
       ForceLogin('');
      return(0);
     };

#5. Retrieve previous page
  $OLDPAGE = $row['pValue']; # most recent page
  $USERKEY = $row['person'];
  $expiry = time() + strict_TIMEOUT;
  if (strlen($THISPAGE) < 2) # if invalid
     { $THISPAGE = $OLDPAGE;
     };
```

```
  $qry = "UPDATE PERSON
            SET pValue = '$THISPAGE',
                pExpiration = $expiry
            WHERE person = $USERKEY";
  if (! mysql_query($qry, $handDB) )
     { print (
  "<br>DEBUG: SQL error in storing page:" . mysql_error() );
     };

  $user = $row['pLoginName'];

#6. Determine user status:
 $qry = "SELECT MAX(PersonRole) AS userStatus FROM PERSDATA
         WHERE Person = $USERKEY";
 list($userstatus) = GetSQL($handDB, $qry, 'get user status');
# print "<br>Dbg: User status for $USERKEY is $userstatus";

# 7. Set cookie, return user details
  setcookie('strict_SID', $SID, $expiry+COOKIE_EXTRA_WAIT);
  $outvalue =
 "USER=\"$user\"|STATUS=\"$userstatus\"|OLDPAGE=\"$OLDPAGE\"";
  return($outvalue); # success
}
```

3.5.2 ForceLogin

This simple function reads the html file *login.htm*, described in section 3.2.3.

```
function ForceLogin ($msg)
{ # fancy dressing up might go here #
  # print($msg);
  readfile('login.htm');
};
```

3.6 Logging out: logout.php

```
 $THISPAGE = ''; # irrelevant.
 require('ValidFx.php'); # our login validation script
 $success = validate_login(SHOW_USER);  # -> $USERKEY, $handDB
 if (! $success)
    { exit();
    };
 $MINUSERSTATUS = NOBODY;
 $MAXUSERSTATUS = EVERYONE;
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                                        'Logging out...', 0);
```

```
  # now $USERKEY, $handDB set, so clean up DB (log out):
  setcookie('strict_SID', '', time()-100); # clear cookie
  $qry = "UPDATE PERSON
            SET pSID = NULL,
                pExpiration = 0
            WHERE person = $USERKEY";
  $handleQ = mysql_query($qry, $handDB);
  if (! $handleQ) # hmm. debugging only.
     { print "<br>Logout error: user key was &lt;$USERKEY&gt; " .
        mysql_error();
     };
  mysql_close($handDB);

print <<<HTML1
<html>
<head><title>Log out from STrICT</title>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>

$MYHEADER

<p>Logged out. Thanks for using STrICT.

<p><a href='InitialLogon.php'>Click here</a> to log in again!
HTML1;
```

3.7 The main page: mainpage.php

[WHEN FINALISED, WE'LL HAVE A GRAPHIC OF OUR MAIN PAGE HERE]

```
 header( 'Cache-control: no-cache' );
 require_once('ValidFx.php'); # our login validation script

# if invoked by InitialLogon.php, then IT defined $success,
# and set $LOGGED_IN; otherwise:
if (! $LOGGED_IN)
   { $success = validate_login(SHOW_USER);
        # returns $USERKEY, $handDB for our use
     if (! $success)
     { exit();
     };
   };
# print "<br>Debug: login is &lt;$success&gt;>";

$MINUSERSTATUS = 0;
$MAXUSERSTATUS = 1000;
```

```
$THISPAGE = 'mainpage.php';
list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
       'STrICT: an online assessment database', 1);
print <<<HTML1
<html>
<head>
<title>STrICT Main Page</title>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type="text/css"
      rel="stylesheet">
</head>
<body>
 <table width=100% height=98%>
  <tr><td class='middling' align='center'>
   <!-- OUTER TBL still no good CSS equivalent ?? -->

<table class='heavybox'><tr><td>
<!-- middle TBL -->
       $MYHEADER
HTML1;

if ($USERSTATUS < 64)
   {
print <<<HTML2a
  <p><a href="strict_assess.php">Assess a Patient Scenario</a>
HTML2a;
   } else
   {
  print <<<HTML2b
<p>Please choose one of the following options:
 <table width=80%"><!-- inner table -->
 <tr>
  <td width='50%'>
   <ol>
    <li><a href="strict_add_studyday.php">
    Create a study day</a>

    <p><li><a href="strict_edit_session.php">
    Edit team data</a>

    <p><li><a href="strict_roster.php">
    Print a roster</a>

    <p><li><a href="strict_assess_super.php">
    Enter form data (registrar/nurse)</a>
```

```
    <p><li><a href="strict_view.php">
    View /delete data</a>

    <p><li><a href="strict_view_results.php">
    Back-up, view and export results</a>
  </td width='50%'>
  <td>
   <ul>
    <li><a href="strict_add_person.php">
    Add a person</a>
    <p><li><a href="strict_edit_person.php">
    Edit a person</a>
    <p><li><a href="strict_edit_logon.php">
    Edit log-on</a>
    <p><li><a href="strict_add_team.php">
    Add participants to a team</a>
    </ul>
 </td></tr></table><!-- end inner tbl -->
HTML2b;
   };

print <<<HTML3
  <p><table width='100%'>
      <tr><td>[<a href="logout.php">Log out</a>]</td>
        <td align='right'>
  [<a href="pdf/AnOnlineDb_0_63.pdf">Documentation</a>] 
  <a href="tex/AnOnlineDb_0_63.tex">src</a></td>
      </tr>
     </table>

   </td></tr></table><!-- end middle tbl -->
 </td></tr></table><!-- end OUTER TBL -->
</body></html>
HTML3;
```

3.8 Creating a study day

We initially present the administrator with a list of current Study days, with their dates They can view (but in no way edit) these study days by clicking on the appropriate link, or they can add a new study day (with Javascript confirmation).

Here we can create a new study day, complete with automatic creation of two associated teams and the associated patient scenarios (four for each team). The script is *strict_add_studyday.php*.

```
## Screen layout: ################################################
#   View/Create study day:                                      #
#                                                               #
#   To obtain track information, please select a day:           #
#                                                               #
#   Date [          ]                                           #
#   Comment [                    ]                              #
#   [click here] to irreversibly create a NEW day!              #
#                                                               #
#################################################################
 header( 'Cache-control: no-cache' );
 $THISPAGE = 'strict_add_studyday.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };

 # if post data for itself, create new study day:
 $DTEXT = $_POST['studydaytext'];
 $TEAMDATE = $_POST['teamdate'];
 if (strlen($TEAMDATE) > 0)
    { Sanitise($TEAMDATE);
      # submitted date format is DD/MM/YYYY:
      # ? make this a routine:
      if (! preg_match ( '/(\d\d?)\/(\d\d?)\/(\d\d\d\d)/',
         $TEAMDATE, $matches))
         { print "<p>Invalid date: $TEAMDATE";
           exit();
         };
      $YYYY = $matches[3];
      $MM = $matches[2];
      $DD = $matches[1];
      if (strlen($MM) == 1)
         { $MM = "0$MM";
         };
      if (strlen($DD) == 1)
```

```
    { $DD = "0$DD";
    };
 $TEAMDATE = "$YYYY-$MM-$DD"; # standard format.
 # print ("<p>Debug: date is $TEAMDATE");

$dvkey = FetchKey($handDB, 'Studyday');
$DNAM = $dvkey-5; # name is different
$qry = "INSERT INTO STUDYDAY (studyday, dText, studydayDate, dName)
       VALUES ($dvkey, '$DTEXT', DATE '$TEAMDATE', '$DNAM')";
DoSQL ($handDB, $qry, 'new studyday entry');
```

In the following section we create two teams for a given study day. Up to version 0.60, the ordering of the ISB scenarios was fixed, but as this might lead to unblinding, we randomly allocate the first team to either an airway or CVS-orientated set of ISB scenarios. We therefore introduce the tiny routine IsbRandomise. The ISBnature value is 1 if we are dealing with airway ISB, and 0 for CVS ISB.

```
# ordering is sequential BUT fetch keys:
$t = 0;
$ISB = IsbRandomise();

while ($t < 2)
  { $t += 1;
    $newtm = FetchKey($handDB, 'Team'); # new key
    $TNAM = $newtm - (2*5);
    $qry = "INSERT INTO TEAM (team, Studyday, ISBnature, tName)
       VALUES ($newtm, $dvkey, $ISB, '$TNAM')";
    DoSQL($handDB, $qry, 'make new team');

    # and create 4 associated PATIENT scenarios!
    $ENTRIES = 0;
    while ($ENTRIES < 4)
      { $ENTRIES += 1;
        $pskey = FetchKey($handDB, 'Patient'); # uPatient
        $pTLC = AutoGenerateTrackId($handDB);
        $qry = "INSERT INTO PATIENT
          (patient, WhichPatient, Team, pTLC)
          VALUES
          ($pskey, $ENTRIES, $newtm, '$pTLC')";
        DoSQL ($handDB, $qry, 'new patient scenario');
      };
    if ($newtm < 11) # v 0.63 amended to 1st 5 days
       { $EXTRAPT = 5;
         if ($ISB == 1)      # if airway scenario..
            { $EXTRAPT += 3; # refer to airway patients.
            };
```

```
                $ENTRIES = 0;
                while ($ENTRIES < 3) # add 3 more patients!
                  { $ENTRIES += 1;
                    $pskey = FetchKey($handDB, 'Patient'); # uPatient
                    $pTLC = AutoGenerateTrackId($handDB);
                    $qry = "INSERT INTO PATIENT
                      (patient, WhichPatient, Team, pTLC, frozen)
                      VALUES
                      ($pskey, $EXTRAPT, $newtm, '$pTLC', 1)";
                    DoSQL ($handDB, $qry, 'new patient scenario');
                    $EXTRAPT += 1;
                  };
              };
          # toggle ISB value:
          $ISB ^= 1; # Exclusive OR to toggle 1->0, 0->1
        };

    };

 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                               "View/add Day with codes", 0);
print <<<HTML1
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
  <script type='text/javascript'>
  <!--

 // DHTML date validation script.
 // Courtesy of SmartWebby.com (http://www.smartwebby.com/dhtml/)
 var dtCh= "/";
 var minYear=2007;  // was 1900
 var maxYear=2100;

function isInteger(s)
  { var i;
    for (i = 0; i < s.length; i++)
        { // Check that current character is number.
          var c = s.charAt(i);
          if (((c < "0") || (c > "9"))) return false;
        }
    return true; // all chars numeric
  }

function stripCharsInBag(s, bag)
  { var i;
    var returnString = "";
```

```
    // Search through string's characters one by one.
    // If character is not in bag, append to returnString.
    for (i = 0; i < s.length; i++)
      { var c = s.charAt(i);
        if (bag.indexOf(c) == -1) returnString += c;
      }
    return returnString;
  }

function daysInFebruary (year)
  { // February has 29 days in any year evenly divisible by four,
    // EXCEPT for centurial years which are not also divisible by 400.
    return (((year % 4 == 0) && ( (!(year % 100 == 0)) || (year % 400 == 0))) ? 29
  }

function DaysArray(n)
  { for (var i = 1; i <= n; i++)
        { this[i] = 31
          if (i==4 || i==6 || i==9 || i==11) {this[i] = 30}
          if (i==2) {this[i] = 29}
        }
    return this
  }

function isDate(dtStr) // modified to accept dd/mm/yyyy:
{
  var daysInMonth = DaysArray(12)
  var pos1=dtStr.indexOf(dtCh)
  var pos2=dtStr.indexOf(dtCh,pos1+1)
  var strDay=dtStr.substring(0,pos1) // modified
  var strMonth=dtStr.substring(pos1+1,pos2) // modified
  var strYear=dtStr.substring(pos2+1)
  strYr=strYear
  if (strDay.charAt(0)=="0" && strDay.length>1) strDay=strDay.substring(1)
  if (strMonth.charAt(0)=="0" && strMonth.length>1) strMonth=strMonth.substring(1)
  for (var i = 1; i <= 3; i++)
     { if (strYr.charAt(0)=="0" && strYr.length>1) strYr=strYr.substring(1)
     }
 month=parseInt(strMonth)
 day=parseInt(strDay)
 year=parseInt(strYr)
 if (pos1==-1 || pos2==-1)
    { alert("The date format should be : dd/mm/yyyy")
      return false
    }
 if (strMonth.length<1 || month<1 || month>12)
    { alert("Please enter a valid month")
      return false
```

```
    }
 if (strDay.length<1
     || day<1
     || day>31
     || (month==2 && day>daysInFebruary(year))
     || day > daysInMonth[month]
     ){ alert("Please enter a valid day")
        return false
      }
 if (strYear.length != 4
     || year==0
     || year<minYear
     || year>maxYear
     ){ alert("Please enter a valid 4 digit year between "+minYear+" and "+maxYear
        return false
      }
 if (dtStr.indexOf(dtCh,pos2+1)!=-1
    || isInteger(stripCharsInBag(dtStr, dtCh))==false)
    { alert("Please enter a valid date")
      return false
    }
 return true
} // end date validation script. Ta!

function ConfirmNewStudyday(myform)
 {
   if ( ! isDate(myform.teamdate.value) )
       { // alert ( 'Invalid date' );
         return (false);
       };
    if (! confirm ("Is the study DATE " + myform.teamdate.value + " correct?"))
       { return (false);
       };
    return (confirm (
    "Please click CANCEL unless you really want a new study day!"
          )        );
 };
  //-->
  </script>
</head>
<body>
$MYHEADER
<p>[<a href='mainpage.php'>Return to main page</a>]

  <p>Each day will have a single associated DVD.
  Each DVD has up to eight 'tracks', each track
  corresponding to a particular patient scenario and team
HTML1;
```

```
$link =
  "<a href='strict_view_studyday.php?studydaynumber=STUDYDAYID'>
  View</a>";
PrintStudydayList($handDB, $link);

print <<<HTML3
  </table></div>
 <p>
 <FORM name="strict_add_studyday"
   ACTION="strict_add_studyday.php"
   METHOD="POST"
   onSubmit="return ConfirmNewStudyday(this)" >
     Optional STUDYDAY text:
     <input type="text" name="studydaytext" size="60">

   <p>
   Date on which team will participate:
    <input type='text' name="teamdate" size='15' value=''> (dd/mm/yyyy)

     <P>
     <INPUT TYPE="submit" NAME="submit"
       VALUE="Create a NEW study day!">
  </FORM>
<p><a href='mainpage.php'>Return to main page</a>
  </body></html>
HTML3;

function AutoGenerateTrackId($handDB)
{ $tries = 0;
  $done = 0;

 # [or move the following chunk UP!]
 # srand(time()); # seed random number generator [ugh]
 $mt = microtime();
 if (! preg_match ( '/0.(.*) /', $mt, $matches))
    { $mt = time(); # ugh.
    } else
    { $mt = $matches[1];
      # print "<br>Debug: microtime is $mt";
    };
 # FOR NOW: TEMP HACK FOR LARA: GENERATE FIXED SEQUENCE:
 $mt = 12345678;
 srand($mt);

  while (! $done
         && ($tries < 2048)
        )
```

```
    { $tries += 1;
      $newid = AlphaString(3);
      # print "<br>Debug: generated ID is '$newid'";
      $qry = "SELECT pTLC FROM PATIENT
        WHERE pTLC = '$newid'"; # ensure is unique!
      list ($dup) = GetSQL($handDB, $qry, "check track id <$newid>");
      if ($dup != $newid)
          { $done = 1;
          };
    };
  if (! $done)
     { print "<p> Oops. could not generate Track ID. FAILED!";
       # stub, have HTML here.
       exit();
     };
  return ($newid);
};

function AlphaString($lgth)
{ # generate alphabetic character string [clumsy]
  $i = 0;
  $str = '';
  while ($i < $lgth)
    { $i  += 1;
      $c = GenAlpha();
      $str = "$str$c";  # ugh.
    };
  return($str);
};

function GenAlpha()
{ # generate single random alpha A..Z
  $i = 65 + (rand()%26);
  return (chr($i));
};

function IsbRandomise()
{ # determine whether ISB scenario is airway or CVS
  return (rand()%2);
}
```

3.8.1 strict_view_studyday.php

The following code simply permits viewing of study day details. We submit (POST) the day number as *studydaynumber*.

```
## Screen layout: ##################################################
#   View details of study day:                                    #
#                                                                 #
#   List of data here:                                            #
#                                                                 #
#   [Return to STUDY DAY menu]                                    #
#                                                                 #
#   [Return to Main menu]                                         #
#                                                                 #
###################################################################
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_view_studyday.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };

 $STUDYDAYNUMBER = $_POST['day']; # first try POST
 if (strlen($STUDYDAYNUMBER) < 1)
    { $STUDYDAYNUMBER = $_GET['studydaynumber']; # NB GET!
      CheckCode($STUDYDAYNUMBER, 'bad day No.');
    } else
    { $newtext = $_POST['newtext'];
      $newname = $_POST['newname']; # ver 0.62
      Sanitise($newtext);
      Sanitise($newname);
      $qry = "UPDATE STUDYDAY
        SET dText = '$newtext',
            dName = '$newname'
        WHERE studyday = $STUDYDAYNUMBER";
      DoSQL($handDB, $qry, 'update study txt');
    };

$qry = "SELECT dText, dName FROM STUDYDAY
          WHERE studyday = $STUDYDAYNUMBER";
list ($STUDYCOMMENT, $STUDYNAME) =
     GetSQL($handDB, $qry, 'get study comment');

 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
          "Study day '$STUDYNAME' (ID: $STUDYDAYNUMBER)", 0);
```

```
print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>
<body>
$MYHEADER
<p>Current TEXT associated with Study day $STUDYDAYNUMBER is:
'$STUDYCOMMENT'.
HTML0;

PrintStudydayTrackList($handDB, $STUDYDAYNUMBER);

print <<<HTML8

 <tr>
  <td colspan='5'>
   <FORM name="strict_view_studyday"
     ACTION="strict_view_studyday.php"
     METHOD="POST"
     onSubmit="return true;" >
     New Text:
    <input type='text' name='newtext' value='$STUDYCOMMENT' size='60'>
    <br> New Name:
    <input type='text' name='newname' value='$STUDYNAME' size='12'>
    <input type='hidden' name='day' value='$STUDYDAYNUMBER'>
    <INPUT TYPE="submit" NAME="submit" VALUE="Alter Text/Name"></td>
   </FORM>
  </td>
 </tr>

  </table>
 </div>
<p><a href='strict_add_studyday.php'>Back to 'Study day' page</a>...
<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML8;

# [rename this]
function PrintStudydayTrackList($handDB, $STUDYDAYNUMBER)
{
  print <<<HTMLpat
  <p><div align='center'>
  <table width='90%' border='2'>
  <tr><td><i>PATIENT ID</i></td>
      <td><i>Team(ID/ISB)</i></td>
      <td><i>Patient name</i></td>
```

```
      <td><i>Timing</i></td>
      <td><i>Unique code</i></td>
  </tr>
HTMLpat;

  $tracks = array();
  $qry = "SELECT patient, CONCAT(TEAM.tName,' (',TEAM.team,'/',ISBnature,')'),
                 PatientName, Timing, pTLC
            FROM PATIENT, TEAM, STUDYDAY, WHICHPATIENT
             WHERE PATIENT.Team = TEAM.team
               AND TEAM.Studyday = STUDYDAY.studyday
               AND STUDYDAY.studyday = $STUDYDAYNUMBER
               AND PATIENT.Whichpatient = WHICHPATIENT.whichpatient
            ORDER BY patient";
  $tms = SQLManySQL($handDB, $qry, 'get study day details');
  PrintDetailTable($tms, 5); # print table with 5 columns
  # note nonstandard mySQL use of CONCAT to add in ISBnature!
}
```

3.9 Edit team sessions

Here we list teams (displaying the number, date, and associated text note). The administrator clicks on one, and this allows him/her to view the sessions for that team, and eventually edit them. When such a link is clicked, the script *strict_do_session.php* is invoked with a single GET argument.

3.9.1 strict_edit_session.php

```
## Screen layout: ###############################################
#   ENTER A PATIENT SESSION:                                    #
#                                                               #
#   Please select one of the following teams:                   #
#                                                               #
#   [insert list of teams]                                      #
#                                                               #
#################################################################
 header( 'Cache-control: no-cache' );
 $THISPAGE = 'strict_edit_session.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };

 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                                        'Edit teams', 0);
print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>
<body>
$MYHEADER
<p>[<a href='mainpage.php'>Return to main page</a>]

  <p>Select one of the following teams:
  <p><div align='center'>
  <table width='90%' border='2'>
  <tr><td><i>Team No. (ID/ISB)</i></td>
      <td><i>Date</i></td>
      <td><i>Note</i></td>
      <td><i>Click here</i></td>
  </tr>
```

```
HTML0;

$link = "<a href='strict_do_session.php?teamkey=TEAMID'>Go</a>";
PrintActiveTeamlist($handDB, $link);  # ugh

print <<<HTML8
  </table>
 </div>

<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML8;

function PrintActiveTeamlist($handDB, $link)
{
  $tms = array();
  $qry = "SELECT CONCAT(tName, ' (',team,'/',ISBnature,')'), studydayDate, tNote
          FROM TEAM, STUDYDAY
          WHERE TEAM.STUDYDAY = STUDYDAY.studyday";
  $tms = SQLManySQL($handDB, $qry, 'get team data');
  $tms = FixupTeamLink($handDB, $tms, $link);
  PrintDetailTable($tms, 4); # print table with 4 columns
}

function FixupTeamLink($handDB, $data, $link)
{ # similar to GetLinkedUserDetails, but simpler.
  # $data is multidimensional data array
  # link must be updated with first element
  # in each row replacing TEAMID in link text.
  # returns array of arrays including link.

  $opt = array();
  $i = 0;
  foreach ($data as $p)
    { $p[3] = $link;    # now get (ID/isb):
        preg_match ( '/\((\d+)\/(\d)\)/', $p[0], $match);
      $p[3] = str_replace ('TEAMID', $match[1], $p[3]);
      $p[0] = str_replace ('/0', '/CVS', $p[0]);
      $p[0] = str_replace ('/1', '/airway', $p[0]);
      # isb: 0 is cardiovascular, 1 is airway
      $opt[$i] = $p;
      $i += 1;
    };
  return($opt);
}
```

3.10 Enter a team

In creating a new team of four members (one registrar, three nurses) we only select potential team members who are not already a member of another team (as our study structure forbids such duplication). In addition, we must check that there is no duplication in our selection of nurses using JavaScript. The creation of a team is otherwise straightforward.[12]

3.10.1 strict_add_team.php

First we select our team members, submitting the POST data to *strict_new_team.php*.

```
## Screen layout: ##############################################
#                                                              #
#    Select team members:                                      #
#                                                              #
#    [registrar]   [nurse1]  [nurse2]  [nurse3]                #
#                                                              #
#    Enter optional comment: _____________________             #
#                                                              #
#    [Create a new team]                                       #
#                                                              #
#   [Return to main menu]                                      #
#                                                              #
################################################################
 header( 'Cache-control: no-cache' );
 $THISPAGE = 'strict_add_team.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                                 "Add team participants", 0);
print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
  <script type='text/javascript'>
  <!--
function CheckAllInputs(myform)
```

[12] At present we have no menu-based method for dissolving and re-creating a team once one is created.

```
  { if (  (myform.teamreg.selectedIndex < 1)
        ||(myform.teamnurse1.selectedIndex < 1)
        ||(myform.teamnurse2.selectedIndex < 1)
        ||(myform.teamnurse3.selectedIndex < 1)
       ) { alert ('Please select 4 team members');
           return(false);
         };
    if (  (myform.teamnurse1.selectedIndex ==
           myform.teamnurse2.selectedIndex)
        ||(myform.teamnurse1.selectedIndex ==
           myform.teamnurse3.selectedIndex)
        ||(myform.teamnurse2.selectedIndex ==
           myform.teamnurse3.selectedIndex)
       ) { alert ('You need 3 distinct nurses (no duplicates)!');
           return(false);
         };
    if ( myform.teamid.selectedIndex < 1)
       { alert('Please select a team');
         return(false);
       };
    if (! confirm('Are you SURE you wish to make this team?'))
       { return(false);
       };
    return (true);
  };
  //-->
  </script>
</head>
<body>
$MYHEADER
(Participants will only appear in the relevant boxes
if they have been added using
'<a href='strict_add_person.php'>Add a person</a>' in the main menu)
<p>
HTML0;

 # also allow addition of new team:
  $qry = "SELECT Person FROM TEAMMEMBER"; # is 'DISTINCT'
  $commalist = &SQLManySQL( $handDB, $qry, 'get team member list');
  # print_r ($commalist);
  if ( count($commalist) > 0)
     { $commalist = DeepImplode($commalist); # turn into list
     } else
     { $commalist = '0';
     };
  # use $commalist as MySQL query fails if inner SELECT null!

 $qry =  "SELECT Person, CONCAT(pdForename, ' ', pdSurname)
```

```
          FROM PERSDATA WHERE PersonRole = 2
            AND Person NOT IN ($commalist)";
  # note idiosyncratic mySQL concatenation syntax CONCAT ipo ||
  # http://dev.mysql.com/doc/refman/5.0/en/string-functions.html
  # perhaps write an SqlStrConcat fx depending on flavour!
  # note hard coding of registrar as 2
  $REGPOPLIST = TextPoplist ($handDB,
     "teamreg", $qry );

  $qry =  "SELECT Person, CONCAT(pdForename, ' ', pdSurname)
           FROM PERSDATA WHERE PersonRole = 1
             AND Person NOT IN ($commalist)";
  # note idiosyncratic mySQL syntax CONCAT ipo ||
  # note hard coding of nurse as 1
  $NURSEPOPLIST1 = TextPoplist ($handDB,
     "teamnurse1", $qry );
  $NURSEPOPLIST2 = str_replace ('teamnurse1', 'teamnurse2',
    $NURSEPOPLIST1);
  $NURSEPOPLIST3 = str_replace ('teamnurse1', 'teamnurse3',
    $NURSEPOPLIST1);

  # create list of available teams (with their dates):
    # first get 'full' teams:
    $qry = "SELECT distinct Team FROM TEAMMEMBER";
    $USEDTEAMS = SQLManySQL($handDB, $qry, 'get full teams');
      # (at present, we don't allow 'half-full' teams)!
  if ( count($USEDTEAMS) > 0) # hmm. make this a fx (other usage)
     { $USEDTEAMS = DeepImplode($USEDTEAMS); # turn into list
     } else
     { $USEDTEAMS = '0';
     };  # MySQL query fails if inner SELECT null!
    # next, create a list of 'empty' teams:
  $qry = "SELECT team, CONCAT(tName, ': ', studydayDate)
    FROM TEAM, STUDYDAY
      WHERE TEAM.studyday = STUDYDAY.studyday
      AND team NOT IN ($USEDTEAMS)";
  # note idiosyncratic mySQL syntax CONCAT ipo ||
  $AVAILABLETEAMS = TextPoplist($handDB, 'teamid', $qry);

print <<<HTML3

<div align='center'>
 <FORM name="strict_new_team"
   ACTION="strict_new_team.php"
   METHOD="POST"
   onSubmit="return CheckAllInputs(this)" >
  <table width=80%>
   <tr><td><i>Registrar</i></td>
```

```
        <td><i>Nurse 1</i></td>
        <td><i>Nurse 2</i></td>
        <td><i>Nurse 3</i></td>
   </tr>
   <tr><td>$REGPOPLIST</td>
        <td>$NURSEPOPLIST1</td>
        <td>$NURSEPOPLIST2</td>
        <td>$NURSEPOPLIST3</td>
   </tr>
   <tr><td colspan='4'>
   Also select a team:
   $AVAILABLETEAMS
   </td></tr>
   <tr><td colspan='2'>
     <INPUT TYPE="submit" NAME="submit" VALUE="Go!"></td>
   </tr>
  </table>
 </form>
</div>
HTML3;

 #addition v0.63: display list of existing teams:
 $EXISTINGTEAMS = ShowExistingTeams($handDB);
 print <<<HTML5
 <div align='center'>
 <table border='1' width='80%'>
   <tr><td><i>Team</i></td>
        <td><i>Registrar</i></td>
        <td><i>Nurse</i></td>
        <td><i>Nurse</i></td>
        <td><i>Nurse</i></td>
   </tr>
   $EXISTINGTEAMS
 </table></div>

HTML5;

print <<<HTML8
<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML8;

function ShowExistingTeams($handDB)
{
  $OPT = ''; # output!
  $qry = "SELECT tName,
                CONCAT(pdForename, ' ', pdSurname,
                  ':<br><i>', LocationText, '<i>'),
```

```
                TEAM.team
            FROM TEAMMEMBER, PERSDATA, TEAM, CURRENTLOCATION
            WHERE TEAMMEMBER.Person = PERSDATA.Person
              AND TEAMMEMBER.Team = TEAM.team
              AND PERSDATA.CurrentLocation = CURRENTLOCATION.currentlocation
             ORDER BY TEAM.team, tmRole DESC";
  $PARTICIP = SQLManySQL($handDB, $qry, 'get ordered team members');
  $oldtid = 0;
  $begin = 1;
  foreach ($PARTICIP as $P)
    { $T = $P[0];
      $M = $P[1];
      $tid = $P[2];
      if ($tid == $oldtid)
         { $OPT .= "<td>$M</td>";
         } else
         { if ($begin)
           { $begin = 0;
           } else
           { $OPT .= "<td><a href='strict_del_grp.php?delid=$oldtid'>
              Delete</a></td></tr>\n";
           };
           $OPT .= "<tr><td>$T</td><td>$M</td>";
           $oldtid = $tid;
         };
    };
  if (! $begin)
     { $OPT .= "<td><a href='strict_del_grp.php?delid=$tid'>
               Delete</a></td></tr>\n";
     };
  return ($OPT);
}
```

3.10.2 strict_del_grp.php

Here's the minor subsidiary routine for deleting a group:

```
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_del_grp.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
```

```
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                                       "Team deleted!", 0);
$grp = $_GET['delid']; # [nasty]
CheckCode($grp, 'Bad team deletion specified');
$qry = "SELECT tName FROM TEAM WHERE team = $grp";
list($THISGRP) = GetSQL($handDB, $qry, 'get del team');
$qry = "DELETE FROM TEAMMEMBER WHERE Team = $grp";
DoSQL($handDB, $qry, 'del grouping');

print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>
<body>
$MYHEADER
You have DELETED all participant entries for Team '$THISGRP'!
(The actual participants still exist, they simply aren't associated with
this Team any more).


<p><a href='strict_add_team.php'>Return to previous page</a>

<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML0;

function ShowExistingTeams($handDB)
{
  $OPT = ''; # output!
  $qry = "SELECT tName,
                CONCAT(pdForename, ' ', pdSurname),
                TEAM.team
           FROM TEAMMEMBER, PERSDATA, TEAM
           WHERE TEAMMEMBER.Person = PERSDATA.Person
             AND TEAMMEMBER.Team = TEAM.team
            ORDER BY TEAM.team, tmRole DESC";
  $PARTICIP = SQLManySQL($handDB, $qry, 'get ordered team members');
  $oldtid = 0;
  $begin = 1;
  foreach ($PARTICIP as $P)
    { $T = $P[0];
      $M = $P[1];
      $tid = $P[2];
      if ($tid == $oldtid)
         { $OPT .= "<td>$M</td>";
         } else
         { if ($begin)
```

```
           { $begin = 0;
           } else
           { $OPT .= "<td><a href='strict_del_grp.php?id=$oldtid'>
              Delete</a></td></tr>\n";
           };
           $OPT .= "<tr><td>$T</td><td>$M</td>";
           $oldtid = $tid;
         };
    };
  if (! $begin)
     { $OPT .= "<td><a href='strict_del_grp.php?id=$tid'>
              Delete</a></td></tr>\n";
     };
  return ($OPT);
}
```

3.10.3 strict_new_team.php

Here we write the submitted team members to the database. POST parameters supplied to this script are:

teamid

teamreg

teamnurse1

teamnurse2

teamnurse3

```
## Screen layout: ##################################################
#                                                                  #
#   New team created                                               #
#   (As a frill we might list the names)                           #
#                                                                  #
#   [Return to main menu]                                          #
####################################################################
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_new_team.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
```

```
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                              "New team created!", 0);
print <<<HTML0
<html>
<head><title>New team created</title>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>
<body>
$MYHEADER
HTML0;

$TEAMID = $_POST['teamid'];
$TEAMREG = $_POST['teamreg'];
$TEAMNURSE1 = $_POST['teamnurse1'];
$TEAMNURSE2 = $_POST['teamnurse2'];
$TEAMNURSE3 = $_POST['teamnurse3'];

CheckCode($TEAMID, 'Bad team ID code');
CheckCode($TEAMREG, 'Bad registrar code');
CheckCode($TEAMNURSE1, 'Bad nurse code 1');
CheckCode($TEAMNURSE2, 'Bad nurse code 2');
CheckCode($TEAMNURSE3, 'Bad nurse code 3');

if (  (TEAMNURSE1 == TEAMNURSE2)
    ||(TEAMNURSE1 == TEAMNURSE3)
    ||(TEAMNURSE2 == TEAMNURSE3)
   ) { print "<p>Error. Duplicate person!";
       exit();
     };

# get names, and check CurrentLocations:
 $qry = "SELECT CONCAT(pdForename , ' ', pdSurname, ': ', LocationText),
        CURRENTLOCATION.currentlocation
           FROM PERSDATA, CURRENTLOCATION
           WHERE CURRENTLOCATION.currentlocation = PERSDATA.CurrentLocation
           AND Person = $TEAMREG";
 $DR =  GetSQL ($handDB, $qry, 'get reg name');
 $qry = "SELECT CONCAT(pdForename , ' ',  pdSurname, ': ', LocationText),
         CURRENTLOCATION.CurrentLocation
           FROM PERSDATA, CURRENTLOCATION
           WHERE CURRENTLOCATION.currentlocation = PERSDATA.CurrentLocation
           AND Person = $TEAMNURSE1";
 $DN1 = GetSQL ($handDB, $qry, 'get nurse1 name');
 $qry = "SELECT CONCAT(pdForename , ' ',  pdSurname, ': ', LocationText),
        CURRENTLOCATION.currentlocation
```

```
           FROM PERSDATA, CURRENTLOCATION
           WHERE CURRENTLOCATION.currentlocation = PERSDATA.CurrentLocation
           AND Person = $TEAMNURSE2";
 $DN2 = GetSQL ($handDB, $qry, 'get nurse2 name');
 $qry = "SELECT CONCAT(pdForename , ' ',  pdSurname, ': ', LocationText),
        CURRENTLOCATION.currentlocation
           FROM PERSDATA, CURRENTLOCATION
           WHERE CURRENTLOCATION.currentlocation = PERSDATA.CurrentLocation
           AND Person = $TEAMNURSE3";
 $DN3 = GetSQL ($handDB, $qry, 'get nurse3 name');
 $DATAR = $DR[0];
 $DATAN1= $DN1[0];
 $DATAN2= $DN2[0];
 $DATAN3= $DN3[0];
 if (  ($DR[1]  != $DN1[1])
     ||($DN1[1] != $DN2[1])
     ||($DN2[1] != $DN3[1])
    ) { print "<p><b class='emphatic'>Warning!</b> The people you have selected
    appear to have originated from different ICUs. Please check this.";
      };


# NOW associate names with team:
  $newmember = FetchKey($handDB, 'Teammember');
  $qry = "INSERT INTO TEAMMEMBER
          (teammember, Team, Person, tmRole)
          VALUES
          ($newmember, $TEAMID, $TEAMREG, 2)";
  DoSQL($handDB, $qry, 'enter registrar');

  $newmember = FetchKey($handDB, 'Teammember');
  $qry = "INSERT INTO TEAMMEMBER
          (teammember, Team, Person, tmRole)
          VALUES
          ($newmember, $TEAMID, $TEAMNURSE1, 1)";
  DoSQL($handDB, $qry, 'enter nurse1');
  $newmember = FetchKey($handDB, 'Teammember');
  $qry = "INSERT INTO TEAMMEMBER
          (teammember, Team, Person, tmRole)
          VALUES
          ($newmember, $TEAMID, $TEAMNURSE2, 1)";
  DoSQL($handDB, $qry, 'enter nurse2');
  $newmember = FetchKey($handDB, 'Teammember');
  $qry = "INSERT INTO TEAMMEMBER
          (teammember, Team, Person, tmRole)
          VALUES
          ($newmember, $TEAMID, $TEAMNURSE3, 1)";
  DoSQL($handDB, $qry, 'enter nurse3');
```

```
print <<<HTML3
<p>Team $TEAMID has been associated with the following team members:
  <ul>
    <li>$DATAR (registrar)
    <li>$DATAN1
    <li>$DATAN2
    <li>$DATAN3
<p><a href='strict_add_team.php'>Previous page</a>
<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML3;
```

3.11 Entering an assessment

There are two options here. An assessor can assess a scenario, but *only if* a DVD has been sent out to them. This implies that a DVD number (which is the same as a study day number) and track must have been entered against that patient scenario.

Alternatively, a super-user can enter data from a form completed by a participant. This is quite different, as the assessor is merely transcribing data (they must still, however, identify the assessor and scenario correctly). Two forms are thus required:

3.11.1 strict_assess.php

Assessor enters data. We will start off by confirming the video clip to be assessed (that it exists, and that this particular assessor hasn't already assessed the clip). We will then obtain the multi-dimensional ratings from the user, and write the results to the database.

First, let's choose a study day:

```
## Screen layout: ##################################################
#   Assess Video of Patient Scenario:                             #
#                                                                 #
#   Please select a DVD from the following list                   #
#   (list goes here)                                              #
#                                                                 #
#   [return to main menu]                                         #
###################################################################
 header( 'Cache-control: no-cache' );
 $THISPAGE = 'strict_assess.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = 1;
 $MAXUSERSTATUS = strict_ASSESSOR_MAX;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                              'Assess Video: select DVD', 0);
print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>
<body>
$MYHEADER
  <p>select one of the following DVDs:
```

```
HTML0;

$link =
 "<a href='strict_assess_studyday.php?studydaynumber=STUDYDAYID'>
 Go</a>";
PrintStudydayList($handDB, $link);

print <<<HTML3
  </table></div>

<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML3;
```

3.11.2 strict_assess_studyday.php

This page accepts a DVD (day) number. (supplied by a GET, not a POST) We now display all patients on the given DVD, provided this assessor hasn't already assessed the video track *and* the associated PATIENT table entry has been committed ('frozen').

Our plan of attack is:

1. Get patients for that study day;
2. Display valid patients, each with an associated radio button. Display the patient as eg. 'Helen 5'. In other words we need to obtain both the **Team** and the **WhichPatient** entry.
3. Require entry of a three letter code (TLC) and check it against pTLC.

If and only if valid track and TLC, POST to *strict_assess_track.php*.

```
## Screen layout: ################################################
#   Choose a single patient:                                    #
#                                                               #
#   Please select a patient from the following list             #
#   (list goes here; can only choose 1 ie radio button)         #
#                                                               #
#   NOW enter the three letter code for that patient: [   ]     #
#                                                               #
#   [return to main menu]                                       #
#################################################################
 header( 'Cache-control: no-cache' );
 $THISPAGE = 'strict_assess_studyday.php';
 require_once('ValidFx.php'); # our login validation script
## $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MINUSERSTATUS = 1;
```

```
$MAXUSERSTATUS = EVERYONE;
$success = validate_login(SHOW_USER); # provides $USERKEY
if (! $success)
   { exit();
   };
list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                            $MINUSERSTATUS, $MAXUSERSTATUS,
                         'Assess Video: select patient', 0);
 $THISSTUDYDAY = $_GET['studydaynumber'];
 CheckCode($THISSTUDYDAY, 'Bad study day number'); # or fail

 $ALLPATIENTS = array(); # [redundant]
 $qry = "SELECT patient FROM PATIENT, TEAM
    WHERE PATIENT.Team = TEAM.team
      AND TEAM.Studyday = $THISSTUDYDAY
    ORDER BY patient";
 $ALLPATIENTS = SQLManySQL ($handDB, $qry, 'get relevant pts');
   # [now clumsy]
 $ALLPATIENTS = Flatten ($ALLPATIENTS); # 1-D array
 $RADIOROWS = '';
 $TRACKCODES ='';
 $MINPT = $ALLPATIENTS[0]; # first patient
 $AVAILABLE = 0;

 foreach ($ALLPATIENTS as $PT)
   { # get patient, team, name ref, and whether entry is committed:
     $qry = "SELECT Team, WhichPatient, frozen, pTLC
            FROM PATIENT WHERE patient = $PT";
     list ($TM, $WP, $FRZ, $UID) =
       GetSQL($handDB, $qry, 'get team etc.');
     $TRACKCODES .= "'$UID',"; # if paranoid, might encrypt[no]

     if ($FRZ == 1)
        { # has user assessed pt?
          $qry = "SELECT rating FROM RATING WHERE
             Patient = $PT AND
             Assessor = $USERKEY";
          list ($DONE) = GetSQL($handDB, $qry, 'already rated?');
          $DISABLED = '';
          if (strlen($DONE)> 0)
             { $DISABLED = ' disabled';
             } else
             { $AVAILABLE += 1;
             };
          $qry = "SELECT PatientName FROM WHICHPATIENT
            WHERE whichpatient = $WP";
          list ($WhichPatientTxt) =
             GetSQL($handDB, $qry, 'get pt name');
```

```
          $RADIOROWS .=
  "<tr><td> $WhichPatientTxt $TM </td><td>
    <input type='radio' name='track' value='$PT' $DISABLED
    onClick=\"javascript:if(this.checked)" .
    "{document.strict_assess_track.radioval.value='$PT';}\">
  </td></tr>";
        };
    };
 $TRACKCODES .= "'0'";  # terminate list
 $AVAILTEXT = 'There is just one patient ';
 if ($AVAILABLE != 1)
    {  $AVAILTEXT = "There are $AVAILABLE patients ";
    };

print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
  <script type='text/javascript'>
  <!--
  trackids = new Array($TRACKCODES);
  var mintrack = $MINPT; // filled in by PHP.

  function CheckInput (myform)
    { tracknum = myform.radioval.value;
      tracknum -= mintrack;
      tlc = myform.tlc.value.toUpperCase();
      if (trackids[tracknum] == tlc)
         { return (true);
         };
      alert (
 'Invalid three letter code for selected track. Please check!');
      return (false);
    }
  //-->
  </script>
</head>
<body>
$MYHEADER
  <p>$AVAILTEXT to assess on this DVD:

<div align='center'>
 <FORM name="strict_assess_track"
      ACTION="strict_assess_track.php"
      METHOD="POST"
      onSubmit="return CheckInput(this)" >
  <input type=hidden name="radioval" value="0">
  <input type=hidden name="assessor" value="$USERKEY">
```

```
  <input type=hidden name="thisstudyday" value="$THISSTUDYDAY">
  <table>
   $RADIOROWS
HTML0;

if ($AVAILABLE > 0)
   { print <<<HTML1
   <tr><td colspan='2'>
   Enter three letter code from video:
     <input type='text' name='tlc' size='4' value=''>
   </td></tr>
   <tr><td colspan='2'>
     <INPUT TYPE="submit" NAME="submit" VALUE="Assess!">
   </td></tr>
HTML1;
   };

print <<<HTML2
  </table>
 </form>
</div>
HTML2;

if ($AVAILABLE > 0)
   { print "<p>Please select a patient by clicking on a
 radio button. Then enter the three-letter code from the
 video, and click 'Assess!'";
   };

print <<<HTML3
  </table></div>
<p><a href='strict_assess.php'>Back to Previous (Study day) page</a>
<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML3;
```

3.11.3 strict_assess_track.php

Here we get down to business — given the unique id of the patient to assess (POST data: *radioval*) (as well as the study day —*thisstudyday*, and the assessor — *assessor*) we enter the looong list of assessment variables. These are all radio buttons, and all rating scales are 1–7.

At the end we check that all variables are completed and submit a mass of POST variables to the unfortunate page: *strict_done* which writes the data to the database.

```
## Screen layout: ##############################################
#   Assess a patient scenario:                                 #
#                                                              #
#  (Long table full of radio buttons goes here)                #
#                                                              #
#   [submit]                                                   #
#                                                              #
#   [return to assessment menu]                                #
#   [return to main menu]                                      #
################################################################
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_assess_track.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = 1;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER); # provides $USERKEY
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                            $MINUSERSTATUS, $MAXUSERSTATUS,
                          'Enter full assessment', 0);
  $PT = $_POST['radioval'];
  CheckCode($PT, 'Bad patient I.D.');
  $STUDYDAY = $_POST['thisstudyday']; # redundant
  CheckCode($STUDYDAY, 'Bad STUDYDAY number'); # or fail
  $ASSESSOR = $_POST['assessor'];
  CheckCode($ASSESSOR, 'Bad assessor ID');

  # if post from supervisor form, we STILL must get the
  # corresponding STUDYDAY No and $PT value:
  if ($STUDYDAY == 0)
     { $WHICHPATIENT = $_POST['whichpatient'];
       CheckCode ($WHICHPATIENT, 'Bad pt??');
       $TEAM = $_POST['team'];
       CheckCode ($TEAM, 'Bad team??');
       $qry = "SELECT patient, Studyday FROM PATIENT, TEAM
```

```
          WHERE PATIENT.Team = TEAM.team
            AND TEAM.team = $TEAM
            AND Whichpatient = $WHICHPATIENT";
      list ($PT, $STUDYDAY) = GetSQL($handDB, $qry, 'get pt,whichpt');
    } else
    { $TEAM = 0; # our little signal ('not supervisor')
    };

# display study name (again), patient and group No. and TLC!
$qry = "SELECT dName FROM STUDYDAY WHERE studyday = $STUDYDAY";
list ($STUDYNAME) = GetSQL($handDB, $qry, 'get sty day name');
# a) get pTLC (TLC):
$qry = "SELECT pTLC FROM PATIENT WHERE patient = $PT";
list ($TLC) = GetSQL ($handDB, $qry, 'get uid');
# b) get team, name, and whether entry is committed:
$qry = "SELECT TEAM.team, PatientName, tName
        FROM PATIENT, WHICHPATIENT, TEAM
        WHERE PATIENT.whichpatient=WHICHPATIENT.whichpatient
          AND PATIENT.Team = TEAM.team
          AND patient = $PT";
list ($TM, $PNAM, $TEAMNAME) = GetSQL($handDB, $qry, 'get team..');
$RADIOHEADER =
  "<h3>Assessing patient '$PNAM $TEAMNAME' on study day $STUDYNAME</h3>
  Three letter code is '$TLC'. ";

# NOTE: WE MUST DISTINGUISH BETWEEN CVS/RESP in:
#  (i) headings
#  (ii)text       ... for technical skills rating:
# For now, we hard-code this (ugh): [fix me]
$ISCVS = 1;
if ( (PatientName == 'Alan') || (PatientName == 'Georgina') ) #
   { $ISCVS = 0; # respiratory(airway)
   };

$t = 'Airway Control'; #
if ($ISCVS)
   { $t = 'ACLS';
   };
$RADIOROWS = "<tr><td width='65%'><b>Please rate TEAM on $t
   Technical Dimensions</b></td>
    <td  colspan='7' width='35%'>
      <table width='100%'>
        <tr><td width='50%'>Never/Rarely</td>
            <td width='50%' align='right'>Consistently</td></tr>
      </table>
    </td></tr>";
$HIDDENVALS ='';
$JAVACHECK = '';
```

```
# first write twelve technical scales Tech1--12
$i =0;
$DT = array();
$qry = 'SELECT ratText FROM RORDER WHERE
     rorder between 1 and 12 ORDER BY rorder'; # nasty
if ($ISCVS)
   {  $qry = 'SELECT ratText FROM RORDER WHERE
     rorder between 51 and 62 ORDER BY rorder'; #
   };
$DT = SQLManySQL ($handDB, $qry, 'get tech dimensions');
$DT = Flatten ($DT); # 1-D

# LATER WE WILL IMPORT TEXT DESCRIPTIONS FROM DATABASE [fix me]
while ($i < 12)
  { $tx = $DT[$i];
    $tx = str_replace("'", "\\'", $tx); # don't mess javascript
    $i += 1;
    $CL = '';
    $RADIOROWS.=
     WriteRadioRow(7, "$i $tx", "Tech$i", "Tval$i", $CL);
    $HIDDENVALS.= "<input type='hidden' name='Tval$i' value='0'>\n";
    $j = " if (myform.Tval$i.value==0){return('$tx');}; \n";
    $JAVACHECK .= $j;
    if ($i == 11)
       { $RADIOROWS .= "<tr><td align='center'>
           <i>SUBJECTIVE GLOBAL ASSESSMENT:</i></td>
           <td  colspan='7'>
             <table width='100%'>
              <tr><td width='50%'>POOR</td>
                  <td width='50%' align='right'>EXCELLENT</td></tr>
             </table>
           </td></tr>";
       };
  };

# next write 24 CRM scales Crm1--24 + 1 for overall assessment!
$qry = 'SELECT ratText FROM RORDER WHERE
     rorder between 101 and 125 ORDER BY rorder'; # nasty
$DT = SQLManySQL ($handDB, $qry, 'get tech dimensions');
$DT = Flatten ($DT); # 1-D
$i =0;
$RADIOROWS .= "<tr><td><b>Please rate how you consider your TEAM
<br>to have performed on the following CRM dimensions</b></td>
     <td  colspan='7'>
       <table width='100%'>
        <tr><td width='50%'>Never/Rarely</td>
            <td width='50%' align='right'>Consistently</td></tr>
```

```
        </table>
      </td></tr>";

  # WE IMPORT TEXT DESCRIPTIONS FROM DATABASE
  while ($i < 25)
    { $tx = $DT[$i];
      $tx = str_replace("'", "\\'", $tx); # don't mess javascript
      $i += 1;
      $j = " if (myform.Cval$i.value==0){return('$tx');}; \n";
      $CL = '';
      # v 0.63: hack to permit non-entry of items 18--20:
      if (($i > 17) && ($i < 21))
         { $j = ''; # no check for 18--20
           $CL = " class='stressed'";
         };
      $JAVACHECK .= $j;
      $RADIOROWS.=
      WriteRadioRow(7, "$i $tx", "Crm$i", "Cval$i", $CL);
      $HIDDENVALS.= "<input type='hidden' name='Cval$i' value='0'>\n";
      if ($i == 20)
         { $RADIOROWS .= "<tr><td align='center'><b>
      <i>The following items are phrased NEGATIVELY for TEAM
      performance:<br> 1 is a GOOD team and 7 is a POOR team</i></b></td>
       <td  colspan='7'>
         <table width='100%'>
          <tr><td width='50%'>Never/Rarely</td>
              <td width='50%' align='right'>Consistently</td></tr>
         </table>
       </td>";
         };
      if ($i == 23)
         { $RADIOROWS .= "<tr><td align='center'>
             <i>SUBJECTIVE GLOBAL ASSESSMENT:</i></td>
             <td  colspan='7'>
               <table width='100%'>
                <tr><td width='50%'>POOR</td>
                    <td width='50%' align='right'>EXCELLENT</td></tr>
               </table>
             </td>";
         };
    };

print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
  <script type='text/javascript'>
  <!--
```

```
function ConfirmClear ()
  { if (confirm (
 'Are you sure you want to clear the form? (Click OK to clear it!)') )
       { return true;
       };
    return false;
  }

  function CheckAllItems (myform)
   { // PHP-generated check here:
     $JAVACHECK;
     return('');
   };

  function CheckInput (myform)
    { var chk = CheckAllItems(myform);
      if (chk.length > 0)
         { alert ("Incomplete data line: " + chk);
           return(false);
         };
         if (! confirm(
   "Are you ENTIRELY happy with the data you've entered?"))
            { return(false);
            };
      return (true); //
    }
  //-->
  </script>
</head>
<body>
$MYHEADER
<p>$RADIOHEADER

<p><div align='center'>
 <FORM name="strict_done"
      ACTION="strict_done.php"
      METHOD="POST"
      onSubmit="return CheckInput(this)" >
  <input type=hidden name="thispatient" value="$PT">
  <input type=hidden name="thisstudyday" value="$STUDYDAY">
  <input type=hidden name="cardiovascular" value="$ISCVS">
  <input type=hidden name="assessor" value="$ASSESSOR">
  <input type=hidden name="team" value="$TEAM">
  $HIDDENVALS
  <table width='85%' border='1'>
   $RADIOROWS
   <tr><td colspan='8'>Optional comment:
     <input type='text' name='comment' value='' size='80'>
```

```
   </td></tr>
   <tr><td colspan='1'>
     <INPUT TYPE="submit" NAME="submit" VALUE="SUBMIT">
       <td colspan='7' align='right'>
     <INPUT TYPE="reset" VALUE="Clear Form"
        onClick="return ConfirmClear()">
   </td></tr>
  </table>
 </form>
</div>
HTML0;

print <<<HTML3
  </table></div>
<p>[<a href='mainpage.php'>ABORT! (Return to main page)</a>]
</body></html>
HTML3;

# --WriteRadioRow: ------------------------------------
# write $cnt radio buttons in a row with numeric values:
# NOTE this is specific for 'strict_done' form!
#
function WriteRadioRow($cnt, $title, $name, $final, $CL)
{ $opt = "<tr$CL><td>$title</td>";
  $i = 0;
  while ($i < $cnt)
    { $i += 1;
      $v = "<td><input type='radio' name='$name' value='$i' " .
           " onClick=\"javascript:if(this.checked)" .
           "{document.strict_done.$final" .
           ".value='$i';}\"></td>";
      $opt .= $v;
    };
  $opt .= "</tr>\n";
  return ($opt);
}
```

3.11.4 strict_done.php

We are now faced with the significant task of assimilating all assessment data, and inserting it into the database. We have the following POST data:

- thispatient : the PATIENT entry;
- cardiovascular : 1 if a CVS scenario, 0 if respiratory;
- assessor : the PERSON id of the assessor;

- Tval1 – Tval12 : technical scores. Note that if we are dealing with a respiratory scenario, then these correspond to RORDER table entries, but for a CVS scenario, we must add 50 to get the RORDER table entry (key) value.
- Cval1 – 25. To get RORDER codes, we must add 100 to these values.

We first create a RATING entry, and then attach multiple ONERATING table entries to this.

```
## Screen layout: ##############################################
#                                                             #
#   Thank you for the assessment. It has been saved.          #
#                                                             #
#   [Return to main menu]                                     #
#                                                             #
###############################################################
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_done.php';
 require_once('ValidFx.php'); # our login validation script
# $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MINUSERSTATUS = 1;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                              "Thank you!", 0);
print <<<HTML0
<html>
<head><title>Rating completed</title>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>
<body>
$MYHEADER
<h3>Rating completed</h3>
HTML0;

 $THISPATIENT = $_POST['thispatient'];
 CheckCode($THISPATIENT, 'Bad patient ID');
 $STUDYDAY = $_POST['thisstudyday']; # no need to check!?
 $ISCVS = $_POST['cardiovascular'];
 CheckCode($ISCVS, 'Missing CVS/respiratory flag');
 $ASSESSOR = $_POST['assessor'];
 CheckCode($ASSESSOR, 'Invalid assessor ID');
 $TEAM = $_POST['team'];
```

```
  # value will be zero unless supervisor completing form
$COMMENT = $_POST['comment'];
Sanitise($COMMENT);

$rk = FetchKey($handDB, 'Rating');
$qry = "INSERT INTO RATING (rating, Patient, Assessor, raText)
  VALUES ($rk, $THISPATIENT, $ASSESSOR, '$COMMENT')";
DoSQL ($handDB, $qry, 'insert rating');

$offset = 0;
if ($ISCVS)
   { $offset += 50;
   };

# first, Technical ratings:
$i = 0;
while ($i < 12)
  { $i += 1;
    $rorder = $i+$offset;
    $rValue = $_POST["Tval$i"];
    $ork = FetchKey($handDB, 'Onerating');
    $qry = "INSERT INTO ONERATING (onerating,
      Rorder, rType, Rating, rValue)
                          VALUES ($ork,
      $rorder, 1, $rk, $rValue)";
      DoSQL($handDB, $qry, '1 rating');
  };

# CRM ratings:
$i = 0;
$offset = 100;
while ($i < 25)
  { $i += 1;
    $rorder = $i+$offset;
    $rValue = $_POST["Cval$i"];
    if (strlen($rValue) > 0) # don't save nulls
       { $ork = FetchKey($handDB, 'Onerating');
         $qry = "INSERT INTO ONERATING (onerating,
           Rorder, rType, Rating, rValue)
           VALUES ($ork, $rorder, 2, $rk, $rValue)";
         DoSQL($handDB, $qry, '2 rating');
       };
  };

if ($TEAM == 0)
   { print
  "<p><a href='strict_assess_studyday.php?studydaynumber=$STUDYDAY'>
     Assess another track on DVD $STUDYDAY</a>";
```

```
   } else
   { print "<p><a href='strict_participant_super.php?teamval=$TEAM'>
     Enter another assessment for TEAM $TEAM</a>";
   };

print <<<HTML3
<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML3;
```

3.11.5 strict_assess_super.php

This script is similar to *strict_assess.php* but the supervisor has in his/her hands a form completed by a team member, and needs to enter data (from the form) *as if* he/she were that person. Our plan of attack is:

1. Select the team (as for strict_edit_session);
2. Select the name of this participant in the session (registrar or one of the nurses), and the patient scenario they are reporting on;
3. Present the editing screen.

After editing has taken place, we give the option of returning to the main page, or going back to select the patient and participant for another form coming from the same team.[13]

In the following script, which starts the above process, we select the team. We then GET this datum to the next page (*strict_participant_super.php*).

```
header( 'Cache-control: no-cache' );
$THISPAGE = 'strict_assess_super.php';
require_once('ValidFx.php'); # our login validation script
$MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
$MAXUSERSTATUS = EVERYONE;
$success = validate_login(SHOW_USER);
if (! $success)
   { exit();
   };
list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                         'Enter form data: select Team', 0);

print <<<HTML0
<html> <head>
```

[13] Note that this differs from the link associated with the DVD assessment.

```
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>
<body>
$MYHEADER
<p><a href='mainpage.php'>Return to main page</a> or ...

  <p>select one of the following teams:

  <p><div align='center'>
  <table width='90%' border='2'>
  <tr><td><i>Team No. (ID)</i></td>
      <td><i>Date</i></td>
      <td><i>Note</i></td>
      <td><i>Click here</i></td>
  </tr>
HTML0;

$link =
  "<a href='strict_participant_super.php?teamval=TEAMID'>Go</a>";
 PrintFrozenTeamlist($handDB, $link);

print <<<HTML8
  </table>
 </div>
<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML8;

function PrintFrozenTeamlist($handDB, $link)
{ # similar to PrintActiveTeamList
  # get ordered list of teams from PATIENT table entries with
  #   frozen=1.

  $tms = array();
  $qry = "SELECT DISTINCT CONCAT(tName, ' (',TEAM.team,')'), studydayDate, tNote
    FROM TEAM,PATIENT,STUDYDAY
      WHERE PATIENT.Team = TEAM.team
       AND  TEAM.Studyday = STUDYDAY.studyday
       AND frozen=1";
  $tms = SQLManySQL($handDB, $qry, 'get team data');
  $tms = FixupTeamLink($handDB, $tms, $link);
  PrintDetailTable($tms, 4); # print table with 4 columns
}

# ? make following fx public (see below)
function FixupTeamLink($handDB, $data, $link)
{ # (see later usage too)
```

```
  $opt = array();
  $i = 0;
  foreach ($data as $p)
    { $p[3] = $link;
        preg_match ( '/\((\d+)\)/', $p[0], $match); # get (ID)
      $p[3] = str_replace ('TEAMID', $match[1], $p[3]);
      $opt[$i] = $p;
      $i += 1;
    };
  return($opt);
}
```

3.11.6 strict_participant_super.php

This form accepts a single GET data item, the team being assessed. Here the administrator selects a team member from the list of four participants. They choose one, and then move on to determine which of the four scenarios is being assessed (*strict_scenario_super.php*). We obtain the participants from the TEAMMEMBER table.

```
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_participant_super.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                               $MINUSERSTATUS, $MAXUSERSTATUS,
                               'Nurse/registrar?', 0);

 $TEAM = $_GET['teamval'];
 CheckCode($TEAM, 'Bad team selection');
 $qry = "SELECT tName FROM TEAM WHERE team = $TEAM";
 list ($TEAMNAME) = GetSQL($handDB, $qry, 'get tm name');

 # get a list of participants:
 $qry = "SELECT PERSDATA.person, pdForename, pdSurname
   FROM TEAMMEMBER, PERSDATA
     WHERE TEAMMEMBER.Person = PERSDATA.Person
       AND TEAMMEMBER.Team = $TEAM";
 # [the above assumes single entry per person in PERSDATA]

 $PARTICIP = array();
 $PARTICIP = SQLManySQL($handDB, $qry, 'get team membrs');
```

```
 $PARTABLE = '';
 foreach ($PARTICIP as $PROW)
   {
     $FORE = $PROW[1];
     $SUR  = $PROW[2];
     $ID   = $PROW[0];
     # also how many ratings for this participant: [ugh]
     $qry = "SELECT COUNT(rating) FROM RATING
             WHERE Assessor = $ID";
     list ($CNT) = GetSQL ($handDB, $qry, 'get num ratings');
     $DISABLED = '';
     if ($CNT > 3)
        { $DISABLED ='disabled';
        };

     $PARTABLE .= "<tr><td>$FORE $SUR ($CNT)</td> " .
 " <td><input type='radio' $DISABLED name='particip' value='$ID' " .
 " onClick=\"javascript:if(this.checked)" .
 "{document.strict_scenario_super.assessor.value='$ID';}\">" .
      "</td>";
   };

print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
  <script type='text/javascript'>
  <!--
  function CheckInput(myform)
    {
      if (myform.assessor.value < 1)
         { alert(
  'Please select the participant whose name is on the form!');
           return(false);
         };
      return (true);
    }
  //-->
  </script>
</head>
<body>
$MYHEADER
<h3>You chose Team $TEAMNAME</h3>
  <p>Select the person who completed the form:
HTML0;

print <<<HTML3
<FORM name="strict_scenario_super"
```

```
      ACTION="strict_scenario_super.php"
      METHOD="POST"
      onSubmit="return CheckInput(this)" >
  <input type=hidden name="assessor" value="0">
  <input type=hidden name="teamval" value="$TEAM">
  <input type=hidden name="teamname" value="$TEAMNAME">

  <div align='center'>
  <table width='40%'>
  <tr><td><i>Participant (No of ratings completed)</i></td>
      <td><i>Click here</i></td>
  </tr>
  $PARTABLE
  <tr><td colspan='2'><INPUT TYPE="submit" NAME="submit"
    VALUE="Confirm your selection...">
  </td></tr>
  </table></div>
</form>

<p><a href='strict_assess_super.php'>Choose another team</a>
<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML3;
```

In the above form we should almost certainly identify (and blank out) any participant for whom all forms have already been completed [FIX ME]!

3.11.7 strict_scenario_super.php

This script accepts a TEAM and a PERSON key value identifying the nurse or registrar who is performing the assessment, both as POST values. We POST from the previous form (rather than a single GET link) as we then require confirmation.

Finally, we select the relevant patient (if still available for data entry for this person), and POST values off to the form *strict_assess_track.php* Values are as follows:

assessor The ID of the assessor;

thisstudyday

radioval Both this and the preceding value are zero — 'stubs' which signal to the receiving page that it needs to look up the PATIENT entry value using the 'team' and 'whichpatient' values.

team

whichpatient These last two values can be used to uniquely identify the PATIENT table entry.

```
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_scenario_super.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                              'Choose patient scenario...', 0);

 $TEAM = $_POST['teamval'];
 $TEAMNAME = $_POST['teamname'];
 $ASSESSOR = $_POST['assessor'];
 CheckCode($TEAM, 'Bad team code');
 CheckCode($ASSESSOR, 'Invalid assessor');
 # print ("<p>Debug: Team is '$TEAM' Assessor '$ASSESSOR'");

 $FORENAME = FetchForename ($handDB, $ASSESSOR);
 $SURNAME = FetchSurname($handDB, $ASSESSOR);

# create a list of PATIENT scenarios for this team and 'assessor'
# we must disable those which have already been assessed
# first, find patients who HAVE been assessed by this assessor:
  $qry = "SELECT PATIENT.patient FROM PATIENT, RATING
   WHERE RATING.patient = PATIENT.patient
     AND PATIENT.Team = $TEAM
     AND Assessor = $ASSESSOR";
 $DONELIST = SQLManySQL($handDB, $qry, 'get rated pts');
  if ( count($DONELIST) > 0)
     { $DONELIST = DeepImplode($DONELIST); # turn into list
     } else
     { $DONELIST = '0';
     };

# next, find those who haven't!
 $qry = "SELECT WHICHPATIENT.Whichpatient, PatientName
   FROM PATIENT, WHICHPATIENT
     WHERE PATIENT.Whichpatient = WHICHPATIENT.whichpatient
       AND Team = $TEAM
       AND patient NOT IN ($DONELIST)";
 $PTLIST = SQLManySQL ($handDB, $qry, 'get unrated pts');
```

```
 $PARTABLE = '';
 foreach ($PTLIST as $PROW)
   {
     $PID = $PROW[0];
     $PNAM  = $PROW[1];
     $PARTABLE .= "<tr><td>$PNAM</td> " .
      " <td><input type='radio' name='particip' value='$PID' " .
      " onClick=\"javascript:if(this.checked)" .
      "{document.strict_assess_track.whichpatient.value='$PID';}\">" .
      "</td>";
   };
 if (count($PTLIST) < 1)
    { $PARTABLE = "<tr><td colspan='2'>
           (No unassessed patient scenarios identified!)</td></tr>";
    };

print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
  <script type='text/javascript'>
  <!--
  function CheckInput(myform)
    {
      if (myform.whichpatient.value < 1)
         { alert('Please select a patient scenario');
           return(false);
         };
      return (true);
    }
  //-->
  </script>
</head>
<body>
$MYHEADER
<h3>$FORENAME $SURNAME: Team $TEAMNAME</H3>
Please select the patient scenario the form refers to:
HTML0;

print <<<HTML3
<FORM name="strict_assess_track"
      ACTION="strict_assess_track.php"
      METHOD="POST"
      onSubmit="return CheckInput(this)" >
  <input type=hidden name="thisstudyday" value="0"><!-- hack -->
  <input type=hidden name="radioval" value="0"><!-- hack -->
  <input type=hidden name="assessor" value="$ASSESSOR">
  <input type=hidden name="team" value="$TEAM">
```

```
  <input type=hidden name="whichpatient" value="0">

  <div align='center'>
  <table width='40%'>

  <tr><td colspan='2'>
  </td></tr>
  $PARTABLE
  <tr><td colspan='2'>
    <INPUT TYPE="submit" NAME="submit"
       VALUE="Confirm your selection...">
  </td></tr>
  </table></div>
</form>

<p><a href='strict_participant_super.php?teamval=$TEAM'>
   Back to previous (TEAM) page</a>
<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML3;
```

3.12 Entering a person

Here we enter basic information about a person. The person might be an assessor, a participant (nurse or registrar), or indeed a teacher/debriefer or even a superuser. When the administrator clicks on the 'Add new' button, the data are submitted as POST data to the PHP script called *strict_admin_personadded.php*. Preliminary processing is first performed by the Javascript in the header of the page.

Note that we need to be able to leave the current specialty blank if dealing with somebody who's not a registrar, and we can leave current location and year of first qualification blank when dealing with an assessor, teacher or superuser.

3.12.1 strict_add_person.php

Full PHP/HTML code follows:

```
## Screen layout: ################################################
#                                                                #
#   Forename: [insert here]                                      #
#                                                                #
#   Surname: [insert]                                            #
#                                                                #
#   Gender: [M|F poplist]                                        #
#                                                                #
#   Role:  [poplist]                                             #
#                                                                #
#   Current location: [poplist]                                  #
#                                                                #
#   Current specialty: [poplist] (registrar only)                #
#                                                                #
#   Year of 1st qualification: [text]                            #
#                                                                #
#   [Add new person button]                  [Abort button]      #
#                                                                #
##################################################################
# Information about years of experience and previous simulation
# exposure are entered in a separate screen entered after
# the administrator clicks on [Add new person].
# Setting up a login name and password is yet another screen!

 header( 'Cache-control: no-cache' );
 $THISPAGE = 'strict_add_person.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
```

```
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                                          'Add a new person', 0);
$THISYEAR = WhatYearIsIt(); # :ancillary.php. INSERTED BELOW.
  print <<<HTML0
<html>
<head><title>Administration --- Enter details of a person</title>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
  <script type='text/javascript'>
  <!--
  // The following is clumsy and must be adjusted
  //   if new fields are added. (aagh).
  function CheckInput(myform)
    { errorcount = 0;
      if (! ValidForename(myform.forename, 0))
         { errorcount ++;
         };
      if (! ValidSurname(myform.surname, 0))
         { errorcount ++;
         };
      if (myform.gender.selectedIndex < 1)
         { errorcount ++;
         };
      if (myform.personrole.selectedIndex < 1)
         { errorcount ++;
         };
      // also check current specialty IF a registrar
      if (myform.personrole.selectedIndex == 2)
         { // note hard-coding of reg as #2 [ugh]
           if (myform.currentspecialty.selectedIndex < 1)
              { errorcount ++;
         };   }; // might give specific error message!
      // check current location, start year IF reg/nurse
      if (myform.personrole.selectedIndex <= 2)
         { if (! ValidStartYear(myform.startyear, 0))
              { errorcount ++;
              };
           if (myform.currentlocation.selectedIndex < 1)
              { errorcount ++;
         };   };
      //
      if (errorcount > 0)
         { if (errorcount == 1)
              { alert (
  "Relevant fields MUST be completed. There was an error!");
              } else
```

```
            { alert (
"Problem! Relevant fields MUST be completed. There were " +
                 errorcount + " errors.");
            };
          return (false);
        };
     return (true);
  }

// ValidStartYear(startyear, chirp):
// check if valid year. If not return false.
// If chirp is nonzero, then also give ALERT.
function ValidStartYear(startyear, chirp)
  { thisyear = $THISYEAR ;
    if (  (startyear.value.length != 4)
        ||(startyear.value < 1950)
        ||(startyear.value > thisyear)
       ) // what if insert alphas...???
       { if (chirp)
            { alert ("Please enter valid 4 digit year e.g. 2003");
            };
        // startyear.select();
        // startyear.focus();
         return (false);
       };
     return (true);
  }

function ValidSurname(surname, chirp)
  { if (  (surname.value.length < 2)
       ) // can add more to the above
       { if (chirp)
            { alert ("Please enter valid surname");
            };
        // surname.select();
        // surname.focus();
         return (false);
       };
     return (true);
  }

function ValidForename(forename, chirp)
  { if (  (forename.value.length < 2)
       ) // can add more to the above
       { if (chirp)
            { alert ("Please enter valid forename");
            };
         // forename.select();
```

```
          // forename.focus();
          return (false);
        };
      return (true);
    }

function ConfirmClear ()
  { if (confirm (
 'Are you sure you want to clear the form? (Click OK to clear it!)') )
       { return true;
       };
    return false;
  }
  //-->
  </script>
</head>
<body>
$MYHEADER
HTML0;


print <<<HTML2
<FORM name="strict_assess"
      ACTION="strict_admin_personadded.php"
      METHOD="POST"
      onSubmit="return CheckInput(this)" >
  <input type=hidden name="confirmation" value="0">
  <table width=65%>
    <tr><td colspan='2'>
      <b>PERSON'S DETAILS:-</b>
    </td></tr>
    <tr><td></td><td>Forename:</td><td>
     <input type="text" name="forename" size="32"
       onChange="ValidForename(this, 1)" >
    </td></tr>
    <tr><td></td><td>Surname:</td><td>
      <input type="text" name="surname" size="32"
       onChange="ValidSurname(this, 1)" >
    </td></tr>
    <tr><td></td><td>Year of 1st qualification:</td><td>
       <input type="text" name="startyear" size="5"
       onChange="ValidStartYear(this, 1)" >
    </td></tr>
    <tr><td width='5%'> </td><td width='14%'>
      Gender: </td><td width='81%'>
      <select name="gender" size="1">
               <option selected value="0">?</option>
               <option value="1">F</option>
```

```
                <option value="2">M</option>
      </select>
    </td></tr>
    <tr><td></td><td>Role: </td><td>
HTML2;

PrintPoplist ($handDB, "personrole",
  "SELECT personrole, PERSONROLE.rText from PERSONROLE
      WHERE personrole > 0
      ORDER BY personrole" );

print <<<HTML3
    </td></tr>
    <tr><td></td><td>Current location: </td><td>
HTML3;

PrintPoplist ($handDB, "currentlocation",
  "SELECT currentlocation, CURRENTLOCATION.LocationText
    from CURRENTLOCATION WHERE currentlocation > 0
      ORDER BY currentlocation" );

print <<<HTML4
    (registrar and nurse only)
    </td></tr>
    <tr><td></td><td>Current specialty: </td><td>
HTML4;

PrintPoplist ($handDB, "currentspecialty",
  "SELECT currentspecialty, CURRENTSPECIALTY.SpecialText
      from CURRENTSPECIALTY WHERE currentspecialty > 0
      ORDER BY currentspecialty" );

print <<<HTML5
    (registrar only)
    </td></tr>
    <tr><td></td><td>
      <INPUT TYPE="submit" NAME="submit" VALUE="Enter new person">
    </td><td align='right'>
      <INPUT TYPE="reset" VALUE="Clear Form"
        onClick="return ConfirmClear()">
    </td></tr>
  </table>
</FORM>
<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML5;
```

The most interesting feature of the above page is the hidden POST variable

called ‘confirmation’, forced to zero, so that strict_admin_personadded.php knows that incoming data is ‘raw’. The possibility exists that similar people might already exist in the database. The latter script checks for this and requires confirmation, then forcing resubmission of data to *itself*, but with the confirmation variable set to 1. If ‘confirmation’ is 1, then no further checking takes place!

3.12.2 strict_admin_personadded.php

We process the POST data from the preceding page. In addition, we [WILL] provide the option to add a user name and password (only for super-users and assessors), or to go straight to the page where further information can be added or edited (including years of experience and previous simulation exposure).

TO DO: (c) fix minor HTML frills in the following...

```
## Screen layout: ##############################################
#                                                             #
#   <Message about added person>                              #
#                                                             #
#   [Add user name/password]  (for assessors/superusers)      #
#                                                             #
#   [Add further information about simulation exposure]       #
#                                                             #
#   [Back to main page]                                       #
#                                                             #
###############################################################
#
# alternatively, if confirmation is required we get:
#
###############################################################
#   WARNING! Similar names exist in the database!             #
#                                                             #
#   LIST OF SIMILAR NAMES:                                    #
#                                                             #
#   [insert list of forename, surname, etc]                   #
#                                                             #
#   Are you sure you wish to add this person?                 #
#                                                             #
#   [Do NOT add person]               [YES, Add person]       #
###############################################################

 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_admin_personadded.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
```

```
 if (! $success)
    { exit();
    };

 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                                        'Adding new person', 0);
  print <<<HTML0
<html>
<head><title>Administration --- Adding new person</title>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
  <script type='text/javascript'>
  <!--
  function CheckInput ()
    { return (true);  // stub
    }
  //-->
  </script>
</head>
<body>
$MYHEADER
HTML0;

$FORENAME    = $_POST['forename'];
$SURNAME     = $_POST['surname'];
$STARTYEAR   = $_POST['startyear'];
$GENDER      = $_POST['gender'];
$PERSONROLE  = $_POST['personrole'];
$CONFIRM = $_POST['confirmation'];
$CURRENTSPECIALTY = $_POST['currentspecialty'];
$CURRENTLOCATION  = $_POST['currentlocation'];

Sanitise($FORENAME);
Sanitise($SURNAME);
CheckCode($GENDER, 'Bad gender value');
 # in the following NULL is returned on failure:
CheckCode($PERSONROLE, 'Bad role code');
CheckCodeNull($CURRENTSPECIALTY);
CheckCodeNull($CURRENTLOCATION);

if ($PERSONROLE < 10 ) # note hard-coded value
   { CheckCode($STARTYEAR, 'Bad year value');
   };
# [we might institute further checks of STARTYEAR]

$similarnames = array(); # null array
if ($CONFIRM == 1) # don't check for similar names!
```

```
   { # do nothing...
   }
elseif ($CONFIRM == 0) # check for similar names
   { $sur = $SURNAME;
     $sur = strtoupper($sur); # upper case
     $similarnames = SQLManySQL($handDB,
       "SELECT distinct Person FROM PERSDATA WHERE
         UPPER(pdSurname) = '$sur' OR
         UPPER(pdForename) = '$sur'", # check for switch!
   'get identical surnames');
      # might use more substantial algorithm (soundex or better)
   } else
   { readfile ('lostdata5.htm');
     exit();
   };

# ----------now insert OR confirm-------------#
if (count ($similarnames) < 1) # if ok, simply store:
   {
    $NEWID = InsertPerson($handDB, $FORENAME, $SURNAME,
                          $STARTYEAR, $GENDER, $PERSONROLE,
                          $CURRENTSPECIALTY, $CURRENTLOCATION);
    if ($NEWID > 0)
       { print <<<HTML2A
         <p>New user added!
         <p>$FORENAME $SURNAME has been added to the database.
         (Role code: $PERSONROLE)
HTML2A;
  if (strlen($STARTYEAR) > 3)
     { print "This person first qualified in $STARTYEAR.";
     };
  if ($PERSONROLE > 19) # [nasty hard-coding again]
     { print <<<HTML2X
       <p>It is now wise to add LOGIN details for this user.
       Click <a href='strict_do_editlogon.php?editlogon=$NEWID'>
       here</a> to do so!
HTML2X;
    } elseif($PERSONROLE < 10) # [hard-coding: exclude teacher!]
    { print <<<HTML2Y
      <p>You may wish to add further information about this
      individual.
      Please click <a href="strict_do_edit.php?editperson=$NEWID">
      here</a> to do so.
HTML2Y;
           };

       } else # at present should never happen...
       { print <<<HTML2D
```

```
         <p>Failed to add user :-(
         <p> Something bad happened while inserting
         user $FORENAME $SURNAME. Please share your grief
         with your friendly local super-administrator.
HTML2D;
       };
    print <<<HTML2E
<p><a href="strict_add_person.php">Add <i>another</i> person</a>
<p><a href="strict_add_team.php">Add people to a team!</a>
<p><a href='mainpage.php'>Return to main page</a>
HTML2E;

   } else  # ARE similar cases:
   { print <<<HTML2B
     <FORM name="strict_assess"
      ACTION="strict_admin_personadded.php" // myself!
      METHOD="POST"
      onSubmit="return CheckInput(this)" >
     <input type=hidden name="forename" value="$FORENAME">
     <input type=hidden name="surname" value="$SURNAME">
     <input type=hidden name="startyear" value="$STARTYEAR">
     <input type=hidden name="gender" value="$GENDER">
     <input type=hidden name="personrole" value="$PERSONROLE">
     <input type=hidden name="currentspecialty"
            value="$CURRENTSPECIALTY">
     <input type=hidden name="currentlocation"
            value="$CURRENTLOCATION">
     <input type=hidden name="confirmation" value="1">

   <h2>Please confirm ...</h2>
     The following users have names similar to the one
     you wish to add ($FORENAME $SURNAME). Please <i>confirm</i>
     that you wish to add this user by clicking on the
     CONFIRM button below. (Obviously if the person already
     exists in the database, Abort and do not confirm).
   <p><div align='center'><table width=65% border='2'>
   <tr><td align='center' colspan='5'>
        <b>Users with similar names</b></td></tr>
   <tr><td><i>Forename</i></td>
       <td><i>Surname</i></td>
       <td><i>Current role</i></td>
       <td><i>Year started</i></td>
       <td><i>(Database) ID No</i></td></tr>
HTML2B;

  # NOTE that because $similarnames was generated by SQLManySQL,
  # it is actually an array of arrays, each sub-array
  # having one element. This is nasty. So we flatten
```

```
  # the array to 1 dimension!
    $similarnames = Flatten($similarnames);
    $users = GetUserDetails($handDB, $similarnames);
  # get forename, surname, role, year started.
  # print ('<br>Debugging: ');
  # print_r ($users);  # debug
    PrintDetailTable($users, 5); # print table with 5 columns

  print <<<HTML2C
  <tr><td colspan='2'><a href='mainpage.php'>
   Abort. Do NOT add user $FORENAME $SURNAME!</a></td>
   <td colspan='3'><INPUT TYPE="submit" NAME="submit"
       VALUE="CONFIRM! Add this person."></td></tr>
  </table></div>
HTML2C;
   };

# ------------------------------------------------------ #
function InsertPerson($myODBC, $FORENAME, $SURNAME,
                      $STARTYEAR, $gender, $role,
                      $CURRENTSPECIALTY, $CURRENTLOCATION)
{
  if (strlen($STARTYEAR) < 4) # bad data: default to 1900 [ugh]
     { $STARTYEAR = '1900';
     };

  $now = date('Y-m-d h:i:s');
  $NEWID = FetchKey($myODBC, 'Person');   # new key for PERSON
  $qry = "INSERT INTO PERSON (person, pMade, FirstQualified)
     VALUES ($NEWID, TIMESTAMP '$now', DATE '$STARTYEAR-01-01')";
  DoSQL($myODBC, $qry, "insert new person");

  $iPERSDATA = FetchKey ($myODBC, 'Persdata');
  $qry = "INSERT INTO PERSDATA (persdata, Person,
          PersonRole, CurrentSpecialty, CurrentLocation,
          pdCreated, pdSurname, pdForename, pdGender)
      VALUES ($iPERSDATA, $NEWID,
          $role, $CURRENTSPECIALTY, $CURRENTLOCATION,
          TIMESTAMP '$now', '$SURNAME', '$FORENAME', $gender)";
  DoSQL ($myODBC, $qry, "insert person data");

  return($NEWID);  # hmm.
}

# ------------------------------------------------------ #
function GetUserDetails($handDB, $similarnames)
{ # $similarnames is array of IDs (key of PERSON table)
  # this fx is cumbersome and slow.
```

```
  # returns array of arrays, each subarray containing forename,
  # surname, role, date of 1st qualifn (as year-01-01) and ID.
  $opt = array();
  $i = 0;
  foreach ($similarnames as $p)
    { $detl = array();
      $detl[0] = FetchForename ($handDB, $p);
      $detl[1] = FetchSurname($handDB, $p);
      $detl[2] = FetchRole ($handDB, $p);
      list ($detl[3]) = GetSQL ($handDB,
        "SELECT FirstQualified
        FROM PERSON WHERE person = $p",
        'GET yr of 1st qualification');
      $detl[4] = $p;
      $opt[$i] = $detl; #
      $i += 1;
    };
  return ($opt);
}
```

3.13 Editing a person

Here we have the facility to edit a person's details, as well as adding more information such as their exposure to simulation and past work experience.

3.13.1 strict_edit_person.php

This introductory page allows us to select a particular person. When the administrator clicks on a link, control is transferred to the page *strict_do_edit.php*.

```
## Screen layout: ###############################################
#   EDIT A PERSON'S DETAILS:                                    #
#                                                               #
#   Please select one of the following users:                   #
#                                                               #
#   [insert list of forename, surname, year started, id No]     #
#                                                               #
#################################################################
 header( 'Cache-control: no-cache' );
 $THISPAGE = 'strict_edit_person.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                               "Edit somebody's details", 0);
print <<<HTML1
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>
<body>
$MYHEADER
<p>[<a href='mainpage.php'>Return to main page</a>]

  <p>select one of the following users (sorted by surname):
HTML1;

$link = "<a href='strict_do_edit.php?editperson=USERID'>Go</a>";
PrintActiveUserlist($handDB, $link);

print <<<HTML3
  </table></div>
```

```
<p><a href='mainpage.php'>Return to main page</a>
  </body></html>
HTML3;
```

3.13.2 strict_do_edit.php

The following PHP script is deceptively short, because it uses GET and then simply invokes a much more complex script. A similar script is *strict_post_edit.php*, which uses POST. Both then invoke a common script called *subedit.php*, which follows shortly.

```
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_do_edit.php';
 require_once('ValidFx.php'); # our login validation script
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
  # here GET user ID
 $OTHERLOGONID = $_GET['editperson'];
 CheckCode($OTHERLOGONID, 'Bad log-on ID'); # or fail
 require ('subedit.php');
```

3.13.3 strict_post_edit.php

The POST variant *strict_post_edit.php* is a bit more complex, because it takes other data concerning work experience, and if valid, updates the relevant table:

```
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_post_edit.php';
 require_once('ValidFx.php'); # our login validation script
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
  # here GET user ID
$OTHERLOGONID = $_POST['editperson'];
CheckCode($OTHERLOGONID, 'Invalid logon id'); # or fail
$EROLE = $_POST['erole'];
$EYEARS= $_POST['eyears'];
$ETEXT = $_POST['etext'];
$SIMEXP =$_POST['simexp'];
$XTEXT = $_POST['xtext'];

 #first, deal with data on work experience
if (strlen($EROLE) > 0)
```

```
   { CheckCode($EROLE, 'Bad role code'); # or fail
     CheckCode($EYEARS, 'Bad year number'); # ok as numeric check
     if (strlen($ETEXT) > 0)
        { Sanitise($ETEXT);
        }
     elseif ($EROLE == 5) # note hard-coding
        { print ("Woops! Must have text comment for 'other'");
          exit();
        };
     # we actually permit general comments if not 'other'!
    $ekey = FetchKey($handDB, 'Experience'); # new key
    $qry = "INSERT INTO EXPERIENCE
       (experience, Person, eRole, eYears, eText)
       VALUES
       ($ekey, $OTHERLOGONID, $EROLE, $EYEARS, '$ETEXT')";
       # ETEXT might be null string.
     DoSQL ($handDB, $qry, 'insert work experience');
   };

 # next, handle simulation exposure
 # (note the two are mutually exclusive)
if (strlen($SIMEXP) > 0)
   { CheckCode($SIMEXP, 'Bad simulation exposure code'); # or fail
     if (strlen($XTEXT) > 0)
        { Sanitise($XTEXT);
        }
     elseif ($SIMEXP == 5) # note hard-coding
        { print ("Hmm! Must have text comment for 'other'");
          exit();
        };
     # here too permit general comments if not 'other'!
    $xkey = FetchKey($handDB, 'Simexposure'); # new key
    $qry = "INSERT INTO SIMEXPOSURE
       (simexposure, Person, Simnature, xText)
       VALUES
       ($xkey, $OTHERLOGONID, $SIMEXP, '$XTEXT')";
       # XTEXT might be null string.
     DoSQL ($handDB, $qry, 'insert sim exposure');
   };


 require ('subedit.php');
```

subedit.php

Here's the much more complicated *subedit.php*. The reason for the above shenanigans is to allow us to POST data from within *strict_do_edit.php* to the POST ver-

sion, which can then invoke itself. Why not simply use POST always? It's easier in some circumstances to submit the user ID using GET!

```
## Screen layout: ###############################################
#   ALTER USER PARTICULARS                                      #
#                                                               #
#   Edit log-on details for user: [Forename Surname]            #
#   2. Alter ONE of the following:                              #
#       2.1  User surname:   [current surname]                  #
#       2.2  User forename:  [current forename]                 #
#       2.3  User role: [poplist of roles]                      #
#       2.4  Location: [poplist]                                #
#       2.5  Specialty: [poplist]                               #
#                                                               #
#   3. Work experience                                          #
#       [table; option to add details]                          #
#                                                               #
#   4. Simulation exposure                                      #
#       [table; option to add details]                          #
#                                                               #
################################################################
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                                        "Edit details", 0);
print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
  <script type='text/javascript'>
  <!--
function CheckInputSurname (myform)
  {   valu = myform.newsurname.value;
      if (valu.length < 2)
         { alert('Minimum surname length is 2 characters');
           // might perform other checks?
           return (false);
         };
      return (true);
  }

function CheckInputForename (myform)
  {   valu = myform.newforename.value;
      if (valu.length < 1)
         { alert('Minimum forename length is 1 character');
           // might perform other checks?
```

```
          return (false);
        };
      return (true);
  }

function CheckAllInputs(myform)
  { // we will not check the poplists, and
    // unlike Agate, there is no ‘died’ or ‘hidden’.
    if (  (! CheckInputForename(myform))
        ||(! CheckInputSurname(myform))
        ) // clumsy
        { return (false);
        };
    return (true);
  };

function CheckWork(workform)
 {
   if (workform.erole.selectedIndex < 1)
      { alert(’Please select a work role’);
        return(false);
      };
   if (  (workform.eyears.value.length < 1)
       ||(workform.eyears.value < 1)
       ||(workform.eyears.value > 40)
      )
      { alert(’Please select a meaningful number of years!’);
        return(false);
      };
   if (workform.erole.selectedIndex == 5)
      // note hard-coding of 5=other!
      { if (workform.etext.value.length < 3)
           { alert(
   "You selected ’other’. Please add a descriptive note!");
             return(false);
           };
      };
   return(true);
 };

function CheckSimExposure(expform)
 {
   if (expform.simexp.selectedIndex < 1)
      { alert(’Please select nature of simulation exposure’);
        return(false);
      };
   if (expform.simexp.selectedIndex == 5)
      // note hard-coding of 5=other!
```

```
      { if (expform.xtext.value.length < 3)
           { alert(
   "You selected 'other'. Please add a descriptive note!");
             return(false);
           };
      };
   return(true);
 };
  //-->
  </script>
</head>
<body>
$MYHEADER
HTML0;

# check user ID or fail:
$OTHERSURNAME = FetchSurname($handDB, $OTHERLOGONID);
$OTHERFORENAME = FetchForename($handDB, $OTHERLOGONID);
$qry = "SELECT pLoginName FROM PERSON WHERE person = $OTHERLOGONID";
list ($OTHERLOGIN) = GetSQL($handDB, $qry, 'get user login');
# the value in OTHERLOGIN may be null.
# in the following we assume only one valid PERSDATA entry [ugh]:
$qry = "SELECT PersonRole FROM PERSDATA WHERE Person = $OTHERLOGONID";
list ($OTHERROLE) = GetSQL($handDB, $qry, 'get role');
$qry = "SELECT CurrentSpecialty FROM PERSDATA WHERE Person = $OTHERLOGONID";
list ($OTHERSPECIALTY) = GetSQL($handDB, $qry, 'get role');
$qry = "SELECT CurrentLocation FROM PERSDATA WHERE Person = $OTHERLOGONID";
list ($OTHERLOCATION) = GetSQL($handDB, $qry, 'get role');

  $qry =  "SELECT personrole, rText FROM PERSONROLE
            WHERE personrole > 0";
  $ROLEPOPLIST = TextPoplistSelected ($handDB,
    "newrole", $qry, $OTHERROLE );

  $qry =  "SELECT currentlocation, LocationText FROM CURRENTLOCATION
            WHERE currentlocation > 0";
  $LOCATIONPOPLIST = TextPoplistSelected ($handDB,
     "newlocation", $qry, $OTHERLOCATION);

  $qry =  "SELECT currentspecialty, SpecialText FROM CURRENTSPECIALTY
            WHERE currentspecialty > 0";
  $SPECIALTYPOPLIST = TextPoplistSelected ($handDB,
     "newspecialty", $qry, $OTHERSPECIALTY);

print <<<HTML3
<p><a href='mainpage.php'>Return to main page</a> or alter details
 for $OTHERFORENAME $OTHERSURNAME (Login name: '$OTHERLOGIN'). By default,
 all values are left at their current settings.
```

```
<div align='center'>
 <FORM name="strict_update"
   ACTION="strict_update.php"
   METHOD="POST"
   onSubmit="return CheckAllInputs(this)" >
  <input type=hidden name="editperson" value="$OTHERLOGONID">

  <table width=80%>
   <tr><td width='30%'>a. Alter surname: </td>
       <td width='70%'>
          <input type='text' name='newsurname'
          size='16' value='$OTHERSURNAME'></td>
   </tr>
   <tr><td width='30%'>b. Alter forename: </td>
       <td width='70%'>
          <input type='text' name='newforename'
          size='16' value='$OTHERFORENAME'></td>
     </tr>
   <tr><td width='30%'>c. Change role:</td>
       <td width='70%'> $ROLEPOPLIST</td>
   </tr>
   <tr><td width='30%'>d. Change location:</td>
       <td width='70%'> $LOCATIONPOPLIST (only for nurse, registrar)</td>
   </tr>
   <tr><td width='30%'>e. Change specialty:</td>
       <td width='70%'> $SPECIALTYPOPLIST (registrar only)</td>
   </tr>
   <tr><td colspan='2'>
     <INPUT TYPE="submit" NAME="submit" VALUE="Go!"></td>
   </tr>
  </table>

 </form>
</div>
HTML3;

if ($OTHERROLE < 10) # note hard-coding
   { # write extra tables for simulation exposure and work experience:
     # FIRST, work experience:
  $qry =  "SELECT erole, rText FROM EROLE
            WHERE erole > 0";
  $EXPERPOP = TextPoplist ($handDB, "erole", $qry);

print <<<HTML4
<div align='center'>
 <FORM name="strict_post_edit"
   ACTION="strict_post_edit.php"
```

```
   METHOD="POST"
   onSubmit="return CheckWork(this)" >
  <input type=hidden name="editperson" value="$OTHERLOGONID">

  <table width=80% border='1'>
   <tr><td colspan='3' align='center'>
     <b>Work Experience</b></td></tr>
   <tr>
    <td align='left'><i>Role</i></td>
    <td align='left'><i>Years</i></td>
    <td align='left'><i>Note</i></td>
   </tr>
HTML4;

  $exper = array();
  $qry = "SELECT EROLE.rText, eYears, eText
    FROM EXPERIENCE, EROLE
    WHERE EXPERIENCE.eRole = EROLE.erole
      AND Person = $OTHERLOGONID ORDER BY experience";
  $exper = SQLManySQL($handDB, $qry, 'get work experience');
  PrintDetailTable($exper, 3); # print table with 3 columns

print <<<HTML5
   <tr>
    <td>$EXPERPOP</td>  <!-- role pop list (erole) -->
    <td><input type='text' name='eyears'
          size='5' value=''></td>
    <td><input type='text' name='etext'
         size='30' value=''></td>
   </tr>
   <tr><td colspan='3'>
     <INPUT TYPE="submit" NAME="submit"
       VALUE="Add above entry!"></td>
   </td></tr>
  </table>
 </form>
</div>
HTML5;

     # NEXT, exposure to simulation:
  $qry =  "SELECT simnature, SimText FROM SIMNATURE
            WHERE simnature > 0";
  $SIMPOP = TextPoplist ($handDB, "simexp", $qry);

print <<<HTML6
<div align='center'>
 <FORM name="strict_sim_exp"
   ACTION="strict_post_edit.php"
```

```
   METHOD="POST"
   onSubmit="return CheckSimExposure(this)" >
  <input type=hidden name="editperson" value="$OTHERLOGONID">

  <table width=80% border='1'>
   <tr><td colspan='2' align='center'>
     <b>Simulation Exposure</b></td></tr>
   <tr>
    <td align='left'><i>Nature of exposure</i></td>
    <td align='left'><i>Note</i></td>
   </tr>
HTML6;

  $exper = array();
  $qry = "SELECT SIMNATURE.SimText, xText
    FROM SIMEXPOSURE, SIMNATURE
    WHERE SIMEXPOSURE.Simnature = SIMNATURE.simnature
      AND Person = $OTHERLOGONID ORDER BY simexposure";
  $exper = SQLManySQL($handDB, $qry, 'get sim exposure');
  PrintDetailTable($exper, 2); # print table with 2 columns

print <<<HTML7
   <tr>
    <td>$SIMPOP</td>  <!-- pop list (simexp) -->
    <td><input type='text' name='xtext'
         size='30' value=''></td>
   </tr>
   <tr><td colspan='3'>
     <INPUT TYPE="submit" NAME="submit"
       VALUE="Add above entry!"></td>
   </td></tr>
  </table>
 </form>
</div>
HTML7;
   };

print <<<HTML8

<p><a href='strict_edit_person.php'>Back to list of people</a>

<p><a href='mainpage.php'>Return to main page</a>
  </body></html>
HTML8;
```

3.13.4 strict_update.php

We accept personal details from the preceding page, and update the database accordingly. Submitted items are:

1. otheruserid
2. newrole
3. newlocation
4. newspecialty
5. newsurname
6. newforename

Each of these values should be valid, with a few exceptions:

- If the new role is not a registrar, then newspecialty can (and must) be null/zero;
- If the new role is neither registrar nor nurse, then newlocation can (and must) be null/zero.

```
## Screen layout: ##################################################
#                                                                  #
#   User details updated. [insert data here]                       #
#                                                                  #
#   [Return to previous menu]                                      #
#                                                                  #
#   [Return to main menu]                                          #
#                                                                  #
####################################################################
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_update.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                              "Details updated", 0);
print <<<HTML0
```

```
<html>
<head><title>Details updated</title>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>
<body>
$MYHEADER
HTML0;

$OTHERLOGONID = $_POST['editperson'];
$NEWROLE =      $_POST['newrole'];
$NEWLOCATION =  $_POST['newlocation'];
$NEWSPECIALTY = $_POST['newspecialty'];
$NEWSURNAME =   $_POST['newsurname'];
$NEWFORENAME =  $_POST['newforename'];

# we might check and forbid a new role of 'superuser'!

CheckCode($OTHERLOGONID, 'Bad logon id'); # or fail
CheckCode($NEWROLE, 'Bad role code');
Sanitise($NEWSURNAME);
Sanitise($NEWFORENAME);
if (  (strlen($NEWSPECIALTY) < 1)
    ||($NEWSPECIALTY == 0)
   ) { $NEWSPECIALTY = 'NULL';
     } else
     { CheckCode($NEWSPECIALTY, 'Bad specialty code');
     };
if (  (strlen($NEWLOCATION) < 1)
    ||($NEWLOCATION == 0)
   ) { $NEWLOCATION = 'NULL';
     } else
     { CheckCode($NEWLOCATION, 'Bad location code');
     };
 # further checks:
 if ($NEWROLE == 2) # hard-coding again
    { if ($NEWSPECIALTY == 'NULL')
         { print
      '<p>Oops! Specialty must be specified for registrar!';
           exit();
         };
    } else
    { $NEWSPECIALTY = 'NULL'; #only for registrar
    };
 if ($NEWROLE < 10) # hard-coding again
    { if ($NEWLOCATION == 'NULL')
         { print '<p>Darn, failed! Must specify location!';
           exit();
```

```
        };
    } else
    { $NEWLOCATION = 'NULL'; #force
    };

$OTHERSURNAME = FetchSurname($handDB, $OTHERLOGONID);
$OTHERFORENAME = FetchForename($handDB, $OTHERLOGONID);

$qry = "UPDATE PERSDATA SET
          pdForename = '$NEWFORENAME',
          pdSurname  = '$NEWSURNAME',
          PersonRole = $NEWROLE,
          CurrentSpecialty = $NEWSPECIALTY,
          CurrentLocation = $NEWLOCATION
        WHERE person = $OTHERLOGONID";
DoSQL($handDB, $qry, 'update user data');

list ($ROLETXT) = GetSQL ($handDB,
  "SELECT rText FROM PERSONROLE WHERE
    personrole = $NEWROLE", 'get role');

list ($SPECTXT) = GetSQL ($handDB,
  "SELECT LocationText FROM CURRENTLOCATION WHERE
    currentlocation = $NEWLOCATION", 'get role');

list ($LOCTXT) = GetSQL ($handDB,
  "SELECT SpecialText FROM CURRENTSPECIALTY WHERE
    currentspecialty = $NEWSPECIALTY", 'get role');

print <<<HTML3
<p>Data have been updated for $OTHERFORENAME $OTHERSURNAME.
<p>Name is now $NEWFORENAME $NEWSURNAME;  role, specialty
  and location are ($ROLETXT; $SPECTXT; $LOCTXT)

<p><a href='strict_do_edit.php?editperson=$OTHERLOGONID'>
   Back to previous page</a>

<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML3;
```

3.14 Editing log-on details

The following page presents a table of users who are permitted to have log-on privileges (Their role is either an assessor or an administrator). Clickable links are created which submit a GET to the page *strict_do_editlogon*. The user ID submitted to that page is associated with the name *editlogon*.

3.14.1 strict_edit_logon.php

```
## Screen layout: ###############################################
#                                                               #
#   ALTER USER LOG-ON                                           #
#                                                               #
#   Please select one of the following users:                   #
#                                                               #
#   [insert list of forename, surname, etc]                     #
#     each user has a clickable link ...                        #
#                                                               #
#################################################################
 header( 'Cache-control: no-cache' );
 $THISPAGE = 'strict_edit_logon.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                              "Edit somebody's log-in", 0);
print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>
<body>
$MYHEADER
<p><a href='mainpage.php'>Return to main page</a> or ...

  <p>select one of the following users (sorted by surname):
  <p><div align='center'>
  <table width='90%' border='2'>
  <tr><td><i>Forename</i></td>
      <td><i>Surname</i></td>
      <td><i>Role</i></td>
      <td><i>Started</i></td>
```

```
      <td><i>Click here</i></td></tr>
HTML0;

  $activeusers = array();
  $qry = 'SELECT distinct Person FROM PERSDATA
     WHERE PersonRole > 10';
  # note hard-coding of teacher as 10. [fix me]
  $activeusers = SQLManySQL($handDB, $qry, 'get active users');
  $activeusers = Flatten($activeusers); #
  $users = GetLinkedUserDetails($handDB, $activeusers,
    "<a href='strict_do_editlogon.php?editlogon=USERID'>Go</a>");
   MyDoubleSort($users, 1, 0); #sort by surname, then forename!
   PrintDetailTable($users, 5); # print table with 5 columns


print <<<HTML3
  </table></div>
<p><a href='mainpage.php'>Return to main page</a>
  </body></html>
HTML3;
```

3.14.2 strict_do_editlogon.php

Here's the page which accepts new log-on details for the selected user. Once the administrator has entered the new details, we must still POST the values to *strict_do_editlogon2.php*.

```
## Screen layout: ###############################################
#   ALTER USER LOG-ON                                           #
#   You have chosen to edit log-on for user: [Forename Surname]#
#                                                               #
#   1. Please FIRST enter YOUR password:     [password box]    #
#      (A security measure)                                    #
#   2. Next, alter one or both of the following:              #
#       2.1  User log-on :   [box, with current log-on]        #
#       2.2  User password: [pwd box]confirm new pwd [pwd box] #
#                                                               #
################################################################
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_do_editlogon.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
```

```
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                                $MINUSERSTATUS, $MAXUSERSTATUS,
                                "Edit person’s details", 0);
print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
  <script type=’text/javascript’>
  <!--
function ConfirmClear ()
  { if (confirm (
 ’Are you sure you want to clear the form? (Click OK to clear it!)’) )
       { return true;
       };
    return false;
  }

function CheckInput (myform)
  { errorcount = 0;
    errmsg = ’’;
    mypwd = myform.mypswd.value;
    ulogon = myform.userlogon.value;
    pwd1 = myform.newpassword1.value;
    pwd2 = myform.newpassword2.value;

    if (NastyString(mypwd))
       { errmsg = errmsg +
  "You password does NOT contain characters \\\\ | ‘ ’  ";
         errorcount ++;
       };
    if (NastyString(pwd1))
       { errmsg = errmsg +
  "Password CANNOT contain characters \\\\ | ‘ ’  ";
         errorcount ++;
       };
    if ( ulogon.length < 4)
       { errmsg = errmsg +
  ’Please supply user name (4 characters or more). ’;
         errorcount ++;
       };
    if ( mypwd.length < 6)
       { errmsg = errmsg +
  "Your password IS 6 or more characters long. ";
         errorcount ++;
       };
    if ( pwd1.length < 6)
       { errmsg = errmsg +
  "New user password: minimum length must be 6 characters. ";
```

```
          errorcount ++;
        };
    if ( pwd2.length < 6)
        { errmsg = errmsg +
  "User password minimum length (confirmation) must be 6 characters. ";
          errorcount ++;
        };
    if ( pwd1 != pwd2)
        { errmsg = errmsg + "Confirmatory password doesn't match! ";
          errorcount ++;
        };
    if (errorcount > 0)
        { alert (errmsg);
          return (false);
        };
    return (true);
  }

function NastyString(pwd)
{ if (pwd.indexOf('\\\\') > -1) // quadruplicate: PHP + Javascript!
     { return (true);
     };
  if (pwd.indexOf('|') > -1)
     { return (true);
     };
  if (pwd.indexOf("'") > -1)
     { return (true);
     };
  if (pwd.indexOf('`') > -1)
     { return (true);
     };
  return(false);
}
  //-->
  </script>
</head>
<body>
$MYHEADER
HTML0;

# here GET user ID or fail..
$OTHERLOGONID = $_GET['editlogon'];
CheckCode($OTHERLOGONID, 'Bad log-on'); # or fail

# hmm. We should also check that the user is entitled to log-on
# privileges:
$OTHERROLE = FetchRole($handDB, $OTHERLOGONID);
if (  ($OTHERROLE != 'superuser')
```

```
    &&($OTHERROLE != 'assessor')
    &&($OTHERROLE != 'teacher+superuser') # ugly coding
   )
   { print "<p>This user (ID $OTHERLOGONID: $OTHERROLE)
     doesn't have log-on privileges.";
     exit(); # fail
   };
$OTHERSURNAME = FetchSurname($handDB, $OTHERLOGONID);
$OTHERFORENAME = FetchForename($handDB, $OTHERLOGONID);
$qry = "SELECT pLoginName FROM PERSON WHERE person = $OTHERLOGONID";
list ($OTHERLOGIN) = GetSQL($handDB, $qry, 'get user login');

print <<<HTML2
<p><a href='mainpage.php'>Return to main page</a> or ...

<FORM name="strict_do_editlogon2"
      ACTION="strict_do_editlogon2.php"
      METHOD="POST"
      onSubmit="return CheckInput(this)" >
 <input type=hidden name="otheruserid" value="$OTHERLOGONID">
 <table width=80% cellpadding=1 cellspacing=1>
   <tr><td><b>1. FIRST please enter YOUR password: </b></td>
    <td> <input type="password" name="mypswd" size="16" /> <br>
      (This is a security measure)</td></tr>
   <tr><td colspan='2'> </td></tr>
   <tr><td><b>2. Enter user log-on/password for...</b> </td>
    <td><b>$OTHERFORENAME $OTHERSURNAME</b></td>
   </tr>
   <tr><td>a. User log-on name: </td>
    <td>
      <input type='text' name='userlogon' size='16'
         value='$OTHERLOGIN'><br>
    (If exists, can be left the same)</td>
   </tr>
   <tr><td rowspan='2'>b. New password: </td>
    <td><input type='password' name='newpassword1' size='16' ><br>
      (Enter new password)</td>
   </tr>
   <tr>
    <td><input type='password' name='newpassword2' size='16' <br>
      (Re-type <i>new</i> password)</td>
   </tr>
   <tr><td><INPUT TYPE="submit" NAME="submit" VALUE="Go!"></td>
    <td><INPUT TYPE="reset" VALUE="Clear Form"
      onClick="return ConfirmClear()"></td></tr>
 </table>
</form>
  </body></html>
```

```
HTML2;
```

3.14.3 strict_do_editlogon2.php

The page which takes the new log-on data and writes them to the database.

```
## Screen layout: ###############################################
#                                                               #
#   User id/password updated. New log-on name [name here]       #
#                                                               #
#   [Return to main menu]                                       #
#                                                               #
#################################################################
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_do_editlogon2.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                              "Edit person's details", 0);
print <<<HTML0
<html>
<head><title>Edit person's details</title>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>
<body>
$MYHEADER
HTML0;

$OTHERLOGONID = $_POST['otheruserid'];
$MYPSWD  = $_POST['mypswd'];
$USERLOGON=$_POST['userlogon'];
$NEWPWD1  =$_POST['newpassword1'];
$NEWPWD2  =$_POST['newpassword2'];

CheckCode($OTHERLOGONID, 'Error in logon'); # or fail
if (!ValidatePassword($USERLOGON, 4))
   { readfile('badlogon.htm');
     exit();
   };
if (! ValidatePassword($MYPSWD, 6))
  # nasty, should md5 in PREVIOUS menu!
```

```
   { readfile('badpassword.htm');
     exit();
   };
if ( (!ValidatePassword($NEWPWD1, 6))
   ||($NEWPWD1 != $NEWPWD2)
   )
   { readfile('badpassword.htm');
     exit();
   };
$qry = "SELECT pPassword FROM PERSON WHERE person = $USERKEY";
list ($mypwd) = GetSQL($handDB, $qry, 'get my pwd');
if ($mypwd != $MYPSWD)
   { readfile('badpassword.htm');
     exit();
   };
$OTHERSURNAME = FetchSurname($handDB, $OTHERLOGONID);
$OTHERFORENAME = FetchForename($handDB, $OTHERLOGONID);
$qry = "UPDATE PERSON SET
          pLoginName = '$USERLOGON',
          pPassword = '$NEWPWD1'
        WHERE person = $OTHERLOGONID";
DoSQL($handDB, $qry, 'update user id/password');

print <<<HTML3
<p>Log-on/password have been updated for user
  $OTHERFORENAME $OTHERSURNAME.
Log-on name is '$USERLOGON'.
<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML3;
```

3.15 Rostering

3.15.1 strict_roster.php

Here we select the study day for which we wish to create a roster.

```
## Screen layout: ##################################################
#   CREATE A ROSTER:                                              #
#                                                                 #
#   Enter the date of the evaluation: [__________]                #
#                                                                 #
#   [Main menu]                                                   #
###################################################################
 header( 'Cache-control: no-cache' );
 $THISPAGE = 'strict_roster.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
```

```
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                                    'Create a roster', 0);

 $qry = "SELECT studyday, CONCAT( dName, ': ', studydayDate)
          FROM STUDYDAY";
  # note idiosyncratic mySQL syntax CONCAT ipo ||
 $STUDYPOP = TextPoplist ($handDB, "studyday", $qry );

print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
  <script type='text/javascript'>
  <!--

function CheckAllInputs(myform)
  { if ( myform.studyday.selectedIndex < 1)
       { // alert ( 'Please select a day!' );
         return (false);
       }
    return (true);
  };

  //-->
  </script>
</head>
<body>
$MYHEADER
<p>[<a href='mainpage.php'>Return to main page</a>]

  <p>Select a study day:
 <FORM name="strict_show_roster"
   ACTION="strict_show_roster.php"
   METHOD="POST"
   onSubmit="return CheckAllInputs(this)" >
HTML0;

print <<<HTML8

    <p>Study day: $STUDYPOP

    <p><INPUT TYPE="submit" NAME="submit"
```

```
        VALUE="Display roster">
    </form>

<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML8;
```

3.15.2 strict_show_roster.php

We display the roster before printing it. All this entails is loading the relevant TXT file, and turning it into an HTML table populated with the required values.

We will read in one of four roster template files, called *Timetable1.txt*, *Timetable2.txt*, *Timetable3.txt* and *Timetable4.txt*. These are stored on the web in a *txt* subdirectory. If the study day associated with this roster has an even number (in the STUDYDAY key field) then we load the second; if odd, the first. If however the ISB scenario is respiratory (ISBnature value in the database for that team is 1), then we add 2 to the timetable number. We replace the following variables within a template with appropriate values:

TEAM1–2 The two teams of the day

DATE The date on which the study will occur

PT1A The first case for the first group of the day;

PT1B The first case for the second group;

PT2A The second case for the first group;

... and so forth.

TEACHER1A ... and likewise for teachers.

To identify each variable to be replaced within the .TXT files, we precede each with a dollar sign. When the substitution has been made, we allow the user to save the result as a file called *roster.csv*. (Although the source data file names are .TXT, these are actually CSV files we're altering). Because there are 11 teachers, we will have to substitute in reverse order.

We can only populate the template if two teams exist for the specified day, and they have been finalised. Otherwise we produce a warning, and perhaps a list of all teams with their dates, in order.

```
## Screen layout: ##################################################
#   SHOW A ROSTER:                                                 #
#                                                                  #
#   Display roster here                                            #
#                                                                  #
#   [Main menu]                                                    #
####################################################################
 header( 'Cache-control: no-cache' );
 $THISPAGE = 'strict_show_roster.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                                    'Create a roster', 0);
  $STUDYDAY = $_POST['studyday'];
 if (strlen($STUDYDAY) > 0)
    { Sanitise($STUDYDAY);
    } else
    { print "<p>Invalid day parameter ($STUDYDAY)";
      exit(); # stub. perhaps read HTML here.
    };
 $qry = "SELECT studydayDate FROM STUDYDAY
           WHERE studyday = $STUDYDAY";
 list ($TEAMDATE) = GetSQL($handDB, $qry, 'get study date');

  $ERROR = 0;
  # Get a list of patients by team:
  $qry = "SELECT PatientName
           FROM PATIENT, WHICHPATIENT, TEAM
          WHERE PATIENT.WhichPatient = WHICHPATIENT.whichpatient
           AND PATIENT.Team = TEAM.team
           AND TEAM.Studyday = $STUDYDAY
           AND PatientNature < 3
           AND frozen = 1
          ORDER BY PATIENT.Team, Timing";
  # v0.62 PatientNature<3 excludes fancy additions
  $PTNTS = Flatten(SQLManySQL($handDB, $qry, 'get pt/tm order'));
  $ODDAY = $STUDYDAY % 2; # modulo 2 (odd/even)
  $C = count($PTNTS);
  if ($C != 8)
     { $MODEL = "<p>Error! There are $C finalised cases.
       There should be 8 (four per team).";
       $ERROR = 3;
```

```
    };

  if (! $ERROR) # next, try teachers:
    { $qry = "SELECT CONCAT(pdForename, ' ', pdSurname)
        FROM TEACHING, PERSDATA, TEAM
          WHERE TEACHING.Teacher = PERSDATA.person
            AND TEACHING.Team = TEAM.team
            AND TEAM.Studyday = $STUDYDAY
          ORDER BY TEAM.Team, Teachrole";
       # [above assumes 1 entry/person in PERSDATA]
      $TEACH = Flatten(SQLManySQL($handDB, $qry, 'team teachers'));
      $T = count($TEACH);
      if ($T != 22)
         { $MODEL = "<p>Error! There are $T teacher roles!
           There should be 22 (11 per team, including coordinators).";
           $ERROR = 4;
         };
    };

  $INPUTS = '';
  if (! $ERROR)
    { $qry = "SELECT tName, ISBnature FROM TEAM
                WHERE Studyday = $STUDYDAY";
      $TMS = SQLManySQL($handDB, $qry, 'get teams');
      $TMS = Flatten ($TMS);
      $TEAMA = $TMS[0];
      $ISBTYPE = $TMS[1];
      $TEAMB = $TMS[2]; # ugh.

      $TEMPLATENUMBER = 1;
      if ($ODDAY == 1)
         { $TEMPLATENUMBER += 1;
         };
      if ($ISBTYPE == 1)
         { $TEMPLATENUMBER += 2; # value is now 1..4
         };
      $templatename = "txt/Timetable$TEMPLATENUMBER" . ".txt";
      $TTBL = file($templatename); # slurp
      # first, strip out comments:
      foreach ($TTBL as $LN)
        { if (! ( preg_match ( '/^%%/', $LN) ) # not comment
              && (preg_match ( '/,/', $LN) )   # is something!
             )
             { $LN = str_replace(",",'</td><td>', $LN);
               $LN = "<tr><td>$LN</td></tr>\n"; # make html
               $MODEL .= $LN;
             };
        };
```

```
        $PTA = array();
        $PTB = array();
        $j = 0;
        while ($j < 4)
          { $j += 1;
            $pa = $PTNTS[$j-1];
            $pb = $PTNTS[4+$j-1];
            $PTA[$j] = $pa;
            $PTB[$j] = $pb;
            $INPUTS .=  "<input type=hidden name='PT" .
                        $j . "A' value='$pa'>";
            $INPUTS .=  "<input type=hidden name='PT" .
                        $j . "B' value='$pb'>";
          };
        $TCHA = array();
        $TCHB = array();
        $k = 0;
        while ($k < 11)
          { $k += 1;
            $ta = $TEACH[$k-1];
            $tb = $TEACH[11+$k-1];
            $TCHA[$k] = $ta;
            $TCHB[$k] = $tb;
            $INPUTS .=  "<input type=hidden name='TEACHER" .
                        $k . "A' value='$ta'>";
            $INPUTS .=  "<input type=hidden name='TEACHER" .
                        $k . "B' value='$tb'>";
          };
        $MODEL = FixTemplate($MODEL, $TEAMDATE,
                    $PTA, $PTB,
                    $TCHA, $TCHB,
                    $TEAMA, $TEAMB);
      };

  if ($ERROR) # will embed message in table
      { $MODEL = "<tr><td>$MODEL</td></tr>";
      };

print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>
<body>
$MYHEADER
<p>[<a href='mainpage.php'>Return to main page</a>]

<p><div align='center'>
```

```
 <table width='95%' border='2'>
  $MODEL
 </table>
</div>

  <p>
 <FORM name="strict_csv_roster"
   ACTION="strict_csv_roster.php"
   METHOD="POST"
   onSubmit="return(true)" >
   <input type=hidden name="teamdate" value="$TEAMDATE">
   <input type=hidden name="teama" value="$TEAMA">
   <input type=hidden name="teamb" value="$TEAMB">
   <input type=hidden name="odday" value="$ODDAY">
   <input type=hidden name="isbtype" value="$ISBTYPE">
   $INPUTS
HTML0;

if (! $ERROR)
   { print <<<HTML7
    <p><INPUT TYPE="submit" NAME="submit"
        VALUE="Download as CSV file">
HTML7;
   };

print <<<HTML8
    </form>
<p>(The downloaded CSV file can be prettied up in Excel
    prior to printing it)!
<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML8;

function FixTemplate ($LN, $TEAMDATE,
          $PTA, $PTB,
          $TCHA, $TCHB,
          $TEAMA, $TEAMB)
{ # cumbersome:
  $LN = str_replace('$TEAMA', $TEAMA, $LN);
  $LN = str_replace('$TEAMB', $TEAMB, $LN);
  $LN = str_replace('$DATE',   $TEAMDATE, $LN);
  $j = 0;
  while ($j < 4)
    { $j += 1;
      $LN = str_replace('$PT' . $j . 'A', $PTA[$j], $LN);
      $LN = str_replace('$PT' . $j . 'B', $PTB[$j], $LN);
    };
  $k = 11;
```

```
  while ($k > 0) # descend to accommodate 'TEACHER10', etc.
    { $LN = str_replace('$TEACHER' . $k . 'A', $TCHA[$k], $LN);
      $LN = str_replace('$TEACHER' . $k . 'B', $TCHB[$k], $LN);
      $k -= 1;
    };
  return ($LN);
}
```

3.15.3 strict_csv_roster.php

This is similar to the preceding, but uses parameters derived there to print to a csv file.

```
# PHP to generate a CSV file for download:
header("Content-type: application/octet-stream");
header("Content-Disposition: attachment; filename=\"roster.csv\"");
header( 'Cache-control: no-cache' );

# $THISPAGE = 'strict_csv_roster.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };

 $TEAMDATE=  $_POST['teamdate'];
 $TEAMA   =   $_POST['teama'];
 $TEAMB   =   $_POST['teamb'];
 $ODDAY   =   $_POST['odday'];
 $ISBTYPE =   $_POST['isbtype'];
 if (strlen($TEAMDATE) < 1)
    { print "<p>Invalid date parameter ($TEAMDATE)";
      exit(); # stub. perhaps read HTML here.
    };
 CheckCode($ODDAY, 'very odd day');
 CheckCode($ISBTYPE, 'bad isb type');
 Sanitise($TEAMDATE);
 Sanitise($TEAMA);
 Sanitise($TEAMB);

       $PTA = array();
       $PTB = array();
       $j = 0;
       while ($j < 4)
         { $j += 1;
           $PTA[$j] = $_POST["PT$j" . 'A'];
```

```
            $PTB[$j] = $_POST["PT$j" . 'B'];
          };
        $TCHA = array();
        $TCHB = array();
        $k = 0;
        while ($k < 11)
          { $k += 1;
            $TCHA[$k] = $_POST["TEACHER$k" . 'A'];
            $TCHB[$k] = $_POST["TEACHER$k" . 'B'];
          };
    # might even sanitise the above!

 $TEMPLATENUMBER = 1;
 if ($ODDAY == 1)
    { $TEMPLATENUMBER += 1;
    };
 if ($ISBTYPE == 0)
    { $TEMPLATENUMBER += 2; # value is now 1..4
    };
 $templatename = "txt/Timetable$TEMPLATENUMBER" . ".txt";

$TTBL = file($templatename); # slurp
foreach ($TTBL as $LN)
  { if (! ( preg_match ( '/^%%/', $LN) ) # not comment
        && (preg_match ( '/,/', $LN) )   # is something!
        )
        { $MODEL .= $LN;
        };
  };
 $MODEL = FixTemplate($MODEL, $TEAMDATE,
             $PTA, $PTB,
             $TCHA, $TCHB,
             $TEAMA, $TEAMB);

print $MODEL;

# put the following in a library [fix me]

function FixTemplate ($LN, $TEAMDATE,
          $PTA, $PTB,
          $TCHA, $TCHB,
          $TEAMA, $TEAMB)
{ # cumbersome:
  $LN = str_replace('$TEAMA', $TEAMA, $LN);
  $LN = str_replace('$TEAMB', $TEAMB, $LN);
  $LN = str_replace('$DATE',   $TEAMDATE, $LN);
  $j = 0;
  while ($j < 4)
```

```
    { $j += 1;
      $LN = str_replace('$PT' . $j . 'A', $PTA[$j], $LN);
      $LN = str_replace('$PT' . $j . 'B', $PTB[$j], $LN);
    };
  $k = 11;
  while ($k > 0) # descend..
    { $LN = str_replace('$TEACHER' . $k . 'A', $TCHA[$k], $LN);
      $LN = str_replace('$TEACHER' . $k . 'B', $TCHB[$k], $LN);
      $k -= 1;
    };
  return ($LN);
}
```

3.15.4 strict_do_session.php

This script accepts a GET argument — a team ID. Similar is the POST variant. Both invoke a longer script, *subsession.php*.

Important details are when during the teaching a particular scenario took place (Timing), and who was the associated teacher/debriefer.

```
header( 'Cache-control: no-cache' );
 # $THISPAGE = 'strict_do_session.php';
require_once('ValidFx.php'); # our login validation script
$success = validate_login(SHOW_USER);
if (! $success)
   { exit();
   };
 # here GET session id
$TEAMKEY = $_GET['teamkey'];
CheckCode($TEAMKEY, 'Bad team id'); # or fail

require ('subsession.php');
```

3.15.5 strict_post_session.php

The POST variant *strict_post_session.php* is a bit more complex. The following page should allow updating of each/all four of the patients, and nine teachers, which requires 12+9+1 POST variables.

It accepts POST data which will be used to update the four existing simulated patient sessions for this team. POST data are the TEAM ID, the Timing for each patient, and nine associated *teacher* entries. In addition we have a 'freeze' code (are we to freeze PATIENT data?), and an 'isfrozen' code (are they already frozen)!

On 6/1/2008 we also added the option to update the tNote text associated with the TEAM.

```
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_post_session.php';
 require_once('ValidFx.php'); # our login validation script
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };

  # Obtain team ID
 $TEAMKEY = $_POST['teamkey'];
 CheckCode($TEAMKEY, 'Bad team code'); # or fail
 $FREEZE = $_POST['freeze'];
 if (strlen($FREEZE) > 0)
    {  CheckCode($FREEZE, 'Absent freeze code'); # should be 1
    } else
    { $FREEZE = 0;
    };
 $ISFROZEN = $_POST['isfrozen'];
 # also check whether invoker detected 'prior freeze' (clumsy)

 # Update patient scenario entries
 $i = 0;
 if ($ISFROZEN != 1)
    { while ($i < 4)
        { $i += 1;
          Upd1 ($handDB, $i, $TEAMKEY, $FREEZE);
        };
    };
 # and teachers:
 $j = 0;
 while ($j < 11)
   { $j += 1;
     Upd2 ($handDB, $j, $TEAMKEY, $FREEZE);
   };
 # a note:
 $NEWNOTE = $_POST['newnote'];
 Sanitise($NEWNOTE);
 $NEWNAME = $_POST['newname'];
 Sanitise($NEWNAME);
 $qry = "UPDATE TEAM SET tNote = '$NEWNOTE',
                         tName = '$NEWNAME'
                     WHERE team = $TEAMKEY";
 DoSQL($handDB, $qry, 'new team note');

 require ('subsession.php');

 function Upd1 ($handDB, $i, $TEAMKEY, $FREEZE )
 { $TIMING      = $_POST["timing$i"];
```

```
   CheckCodeNull($TIMING, 'Bad AFTER code');
   $qry = "UPDATE PATIENT
            SET Timing  = $TIMING,
                frozen   = $FREEZE
            WHERE WhichPatient = $i
              AND Team = $TEAMKEY
              AND frozen <> 1";
   DoSQL ($handDB, $qry, 'update PATIENT data');
 }

 function Upd2 ($handDB, $i, $TEAMKEY, $FREEZE )
 { $TEACHER = $_POST["teacher$i"];
   CheckCode($TEACHER, "Incorrect teacher($i) code");
   # first, check whether entry exists for this team/ROLE:
   $qry = "SELECT teaching FROM TEACHING
     WHERE Team = $TEAMKEY AND Teachrole = $i";
   list ($teaching) = GetSQL($handDB, $qry, 'get teaching');
   if (strlen($teaching) > 0) # does exist:
      { $qry = "UPDATE TEACHING
                 SET Teacher = $TEACHER
                 WHERE teaching = $teaching";
      } else
      { $teaching = FetchKey($handDB, 'Teaching');
        $qry = "INSERT INTO TEACHING
        (teaching, Team, Teacher, Teachrole)
        VALUES
        ($teaching, $TEAMKEY, $TEACHER, $i)";
      };
   DoSQL ($handDB, $qry, 'update TEACHING data');
 }
```

In the above function (Upd1) we never permit updating of data in the PATIENT table once the frozen flag has been set to 1. These values are then 'cast in stone'.

subsession.php

This script is similar to *subedit.php*. It can POST to the preceding script.

```
## Screen layout: ###############################################
#   VIEW/EDIT SIMULATED PATIENT SESSIONS for group X:          #
#                                                              #
#   Here we list all sessions for a given group:               #
#   (The four columns on the right can all be edited)          #
#    Patient    Timing DVD track ID                            #
#    -------    ------ ------------                            #
#    Helen      fourth     AQX                                 #
#    Alan       second     JXE                                 #
```

```
#    Donald     third      VCX                                  #
#    Georgina   first      AJZ                                  #
#                                                              #
#    Teaching role:               Teacher:                     #
#    Debrief + assessment 1       Bloggs                       #
#    Debrief + assessment 2       ...                          #
#    Defibrillator skill station                               #
#    CRM lecture                                               #
#    Airway skill station                                      #
#    Immersive intervention                                    #
#    Problem-centred learning                                  #
#    Debrief + assessment 3                                    #
#    Debrief + assessment 4                                    #
#                                                              #
################################################################
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                                        "Edit team sessions", 0);
 CheckCode($TEAMKEY, 'Aagh. bad team');
```

In the following code (amended in v0.61) we determine not only the study day, name and note associated with a team, but also the type of ISB scenario: airway (0) or cardiovascular (1).

```
 $qry = "SELECT Studyday, tNote, ISBnature, tName FROM TEAM WHERE team = $TEAMKEY'
list ($THISSTUDYDAY, $TEAMNOTE, $ISBTYPE, $TEAMNAME) =
       GetSQL($handDB, $qry, 'get study day..');
if ($ISBTYPE == 1)
   { $ISBTYPE = 'airway';
   }
elseif ($ISBTYPE == 0)
  { $ISBTYPE = 'cardiovascular';
  } else
  { $ISBTYPE = '?';
  };

  $STUFF = '';
  $i =0;
  $FROZEN = 0;
  while ($i < 4)
    { $i += 1;
      list ($FRZ, $PT, $ISA, $TLC) = GetSQL($handDB,
       "SELECT frozen, PatientName, Timing, pTLC
         FROM PATIENT, WHICHPATIENT
           WHERE PATIENT.Whichpatient = WHICHPATIENT.whichpatient
            AND Team = $TEAMKEY
```

```
            AND PATIENT.WhichPatient = $i", "get teacher $i");
       if ($FRZ == 1)
          { $FROZEN = 1;
          }; # if any frozen, all are!
       if (strlen($ISA) < 1)
          { $ISA = 0;
          };
       # note we ignore Teacher value retrieved in the above.
      $ISAPOP = "<select name='timing$i' size='1'>
            <option value='0'>?</option>
            <option value='1'>first</option>
            <option value='2'>second</option>
            <option value='3'>third</option>
            <option value='4'>fourth</option>
          </select>";
      $ISAPOP = str_replace ("value='$ISA'",
          "selected value='$ISA'", $ISAPOP);
      if ($FROZEN == 1)
         { $ISAPOP = str_replace ('<select',
            '<select disabled', $ISAPOP);
         };
     $STUFF .= "<tr><td>$PT $TEAMKEY</td>
     <td>$ISAPOP</td>
     <td>$TLC</td>
     </td></tr>";
    };  # end 'while $i < 4'

print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
  <script type='text/javascript'>
  <!--

function CheckInput(myform)
  {
    var i1, i2, i3, i4;
    i1 = myform.timing1.selectedIndex;
    i2 = myform.timing2.selectedIndex;
    i3 = myform.timing3.selectedIndex;
    i4 = myform.timing4.selectedIndex;

    // ensure that Alan/Georgina don't both have same timing:
    if (  ((i1 > 0) && (i3 > 0))
        &&(  ((i1 < 3) && (i3 < 3))
           ||((i1 > 2) && (i3 > 2))
         )
       ) { alert (
```

```
        "Alan and Georgina can't BOTH be 1st/2nd; or both 3rd/4th!");
          return (false);
        }; // note hard-coding.
    // and likewise for the other two
    if (  ((i2 > 0) && (i4 > 0))
        &&(  ((i2 < 3) && (i4 < 3))
           ||((i2 > 2) && (i4 > 2))
         )
       ) { alert (
        "Donald and Helen can't BOTH be 1st/2nd; or both 3rd/4th!");
          return (false);
        }; // note hard-coding.
    // also ensure (5/1/2008) no 2 share same nonzero timing:
    if (  (  (i1 > 0)
           &&((i1 == i2) || (i1 == i4)
         )  )
        ||((i2 > 0) && (i2 == i3)
         )
        ||((i3 > 0) && (i3 == i4)
         )
       ) { alert ("Two scenarios cannot share the same Timing");
          return (false);
        };

    var froz = $FROZEN;
    if (! froz)
       { if (myform.freeze.checked)
            { if (  (i1 <1)
                  ||(i2 <1)
                  ||(i3 <1)
                  ||(i4 <1)
                ) { alert ("Can't commit! Incomplete!!");
                    return (false);
                  };
              if (! confirm(
"Are you sure you wish to IRREVOCABLY commit the above?"))
                { return(false);
                };
              if (! confirm(
"Are you REALLY sure? Click CANCEL if you have the slightest doubt."))
                { return(false);
                };
            };
       };

    return (true);
};
//-->
```

```
  </script>
</head>
<body>
$MYHEADER
<p>Session data for Team $TEAMNAME: (ISB randomisation is to <i>$ISBTYPE</i> scena

<p>
<div align='center'>
 <FORM name="strict_post_session"
      ACTION="strict_post_session.php"
      METHOD="POST"
      onSubmit="return CheckInput(this)" >
  <input type=hidden name="teamkey" value="$TEAMKEY">
  <table width='60%'>
  <tr><td><i>Patient</i></td>
      <td><i>Timing</i></td>
      <td><i>DVD Track ID</i></td>
  </tr>
HTML0;

print $STUFF;

if (! $FROZEN)
   { print <<<HTML5A
   <tr><td colspan='3'>
     Irrevocably commit timing:
      <input type='checkbox' name='freeze' value='1'>
     <br>(Don't check this box unless you're certain)
   </td></tr>
HTML5A;
   };

  # also POST whether we are already frozen: (clumsy)
  print "<input type=hidden name='isfrozen' value='$FROZEN'>";

  # Is the ISB scenario airway or cardiovascular?
  # print "<tr><td colspan='3'>ISB type: $ISBTYPE</td></tr>";

# next, the teachers, as a separate list:
  print "<tr><td colspan='3'>
  <b>Teachers associated with this session</b></td></tr>";
  print "<tr><td colspan='2'><i>Teaching role</i></td>
            <td><i>Teacher</i></td></tr>";
  # list roles:
  $qry = "SELECT trText FROM TEACHROLE ORDER BY teachrole";
  $TROLES = array();
  $TROLES = SQLManySQL($handDB, $qry, 'list teacher roles');
  $TROLES = Flatten($TROLES);
```

```
  # create table rows:
  # NB. the following ASSUMES that the role codes start at 1 (Coordinator)
  # and there are no omissions, for purposes of later processing!
  $j = 0;
  foreach ($TROLES as $tr)
    {
      $j += 1;
      # get current Teacher, if exists:
      $qry = "SELECT Teacher FROM TEACHING WHERE
         Team = $TEAMKEY AND Teachrole = $j";
      list ($TCH) = GetSQL($handDB, $qry, 'get current teacher');
      if (strlen($TCH)<1)
         { $TCH = 0; # default to nobody!
         };
      $qry =  "SELECT Person, CONCAT(pdForename, ' ', pdSurname)
          FROM PERSDATA
           WHERE PersonRole = 10 OR PersonRole = 202"; # hard-coded
      # NOTE idiosyncratic mySQL CONCAT.
      $TEACHPOP = TextPoplistSelected ($handDB,
        "teacher$j", $qry, $TCH);
      # as of version 0.5, we don't freeze these entries:
      # if ($FROZEN == 1)
      #   { $TEACHPOP = str_replace ('<select',
      #   '<select disabled', $TEACHPOP);
      #   };
      print "<tr><td colspan='2'>$tr</td>
                  <td>$TEACHPOP</td></tr>\n";
    };

 print <<<HTML7A
   <tr>
    <td colspan='3'>
     Comment:
     <input type='text' size='60' name='newnote' value='$TEAMNOTE'>
    </td>
    <td colspan='3'>
     Team <i>name</i>:
     <input type='text' size='12' name='newname' value='$TEAMNAME'>
    </td>
   </tr>
   <tr><td colspan='3'>
     <INPUT TYPE="submit" NAME="submit" VALUE="Update database."></td>
   </tr>
HTML7A;

print <<<HTML8
  </table>
```

```
 </form>
</div>
<p>[<a href="strict_edit_session.php">edit another session</a>]
<p><a href='mainpage.php'>Return to main page</a>
  </body></html>
HTML8;
```

3.16 Pulling out data

[Describe analysis menus, and CSV export]

```
 header( 'Cache-control: no-cache' );
 $THISPAGE = 'strict_view_results.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                            'Backup; View study results', 0);
print <<<HTML1
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>
<body>
$MYHEADER
<h3>Back-up</h3>
<p><a href='backup_all.php'>Back-up all data</a> (CSV export)
<h3>Data extraction</h3>
<p>[Here we will pull out data in a format
    specified by the study authors]
HTML1;

print <<<HTML3
<p>[<a href='mainpage.php'>Return to main page</a>]
</body></html>
HTML3;
```

3.16.1 CSV back-up: backup_all.php

This page is clumsy. We print data for all SQL tables to a single CSV file, hoping that we never have to use these data. [We still need to spruce this up, extracting

SQL meta-data for each table].

We constrain our export to rows where the key is >= 1000, except for the STUDYDAY and TEAM tables, where we begin at 1.

```
# PHP to generate a CSV file for download:
header("Content-type: application/octet-stream");
header("Content-Disposition: attachment; filename=\"FULLBACKUP.csv\"");
header( 'Cache-control: no-cache' );

 $THISPAGE = 'backup_all.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);  # provides $handDB
 if (! $success)
    { exit();
    };

$date = date("F j, Y");
print "%%DatabaseBackupName=\"STrICT\",DatePerformed=\"$date\"";
print "\n" . "%% If you wish to restore the database, you must";
print "\n" . "%%  split this file into many CSV files, one per table,";
print "\n" . "%%   and re-create a fresh database, importing the CSV files";
 # Later we will write a Perl script to automate this recovery process

 # get list of tables:
 $qry = "SHOW TABLES"; # good in mySQL
 $tbllist = SQLManySQL($handDB, $qry, 'show tables');
 $tbllist = Flatten($tbllist);
 foreach ($tbllist as $t)
   { $keymin = 1000;
     if (  ($t == 'STUDYDAY')
        || ($t == 'TEAM')
        ) { $keymin = 1;
          }; # clumsy hack (minimum key):
     PrintTableCsvData ($handDB, $t, $keymin);
   };
 print "\n\n" . "%% ---------END OF DATA------------ ";

function PrintTableCsvData ($handDB, $tname, $keymin)
{
  $HDR = "\n\n" . "%%TableName=\"$tname\""; # record table name
  $keyname = strtolower($tname); # lower case key: our rule!

  # Include column meta-data here:
    $HDR .= "\n" . "%%ColumnData";
  # get the very first row, so we can get field attribs:
    $qry = "SELECT * FROM $tname WHERE $keyname = $keymin";
```

```
    $RSLT = mysql_query($qry);
    $FLDS = mysql_num_fields($RSLT);
    $j = 0;
    $HDR .= "($FLDS)=";
    while ($j < $FLDS) # for each field
      { $fnam = mysql_field_name($RSLT, $j);
        $ftype = mysql_field_type($RSLT, $j);
        $HDR .= "$fnam($ftype),";
        $j += 1;
      };
    if (mysql_fetch_array($RSLT)) # if any rows:
       { print $HDR;  # print header;
                      # next, get and print all data:
         $qry = "SELECT * FROM $tname WHERE $keyname >= $keymin";
         $ALLDATA = SQLManySQL($handDB, $qry, 'get all table data');
         foreach ($ALLDATA as $DROW)
            { print "\n"; # new row
              foreach ($DROW as $DI)
                { # replace CR within data with \n:
                  $DI = str_replace("\n", '\n', $DI);
                  # replace comma with \,
                  $DI = str_replace(",", '\,', $DI);
                  print "\"$DI\",";
                };
            };
       };
};
```

3.17 Viewing and deleting data

We need to be able to view data in a simple fashion, and even delete them. We will follow a similar process to the one we used in *strict_assess_super.php*, first selecting a team, then a *patient scenario* (not who completed the form), and finally one of the completed assessments (if any exist). The first script is *strict_view.php*.

3.17.1 strict_view.php

```
 header( 'Cache-control: no-cache' );
 $THISPAGE = 'strict_view.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                                        'View/delete data', 0);

print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>
<body>
$MYHEADER
<p><a href='mainpage.php'>Return to main page</a> or ...

  <p>select one of the following teams:

  <p><div align='center'>
  <table width='90%' border='2'>
  <tr><td><i>Team No. (ID)</i></td>
      <td><i>Date</i></td>
      <td><i>Note</i></td>
      <td><i>Click here</i></td>
  </tr>
HTML0;

$link = "<a href='strict_view_patient.php?teamval=TEAMID'>Go</a>";
 PrintFrozenTeamlist($handDB, $link);

print <<<HTML8
  </table>
```

```
 </div>
<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML8;

function PrintFrozenTeamlist($handDB, $link)
{ # similar to PrintActiveTeamList
  # get ordered list of teams from PATIENT table entries with
  #   frozen=1.

  $tms = array();
  $qry = "SELECT DISTINCT CONCAT(tName, ' (',TEAM.team,')'), studydayDate, tNote
    FROM TEAM,PATIENT,STUDYDAY
      WHERE PATIENT.Team = TEAM.team
       AND  TEAM.Studyday = STUDYDAY.studyday
       AND frozen=1";
  $tms = SQLManySQL($handDB, $qry, 'get team data');
  $tms = FixupTeamLink($handDB, $tms, $link);
  PrintDetailTable($tms, 4); # print table with 4 columns
}

# the following fx might be made public (it recurs)
function FixupTeamLink($handDB, $data, $link)
{ # (see later usage too)
  $opt = array();
  $i = 0;
  foreach ($data as $p)
    { $p[3] = $link;
      preg_match ( '/\((\d+)\)/', $p[0], $match); # get (ID)
      $p[3] = str_replace ('TEAMID', $match[1], $p[3]);
      $opt[$i] = $p;
      $i += 1;
    };
  return($opt);
}
```

3.17.2 strict_view_patient.php

Here, given a particular team (GET, not POST), we display all 4 patient scenarios for that team, and the administrator chooses one of them. We receive POST data describing the team ID as *teamval*.

```
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_view_patient.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
```

```
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                              'Choose patient scenario...', 0);

 $TEAM = $_GET['teamval'];
 CheckCode($TEAM, 'Bad team code');

# create a list of PATIENT scenarios for this team.
# We should disable access to scenarios for which no assessments
# exist.

  $ERR = 0;
  $qry = "SELECT DISTINCT WHICHPATIENT.Whichpatient, PatientName
   FROM PATIENT, RATING, WHICHPATIENT
   WHERE RATING.patient = PATIENT.patient
     AND PATIENT.Whichpatient = WHICHPATIENT.whichpatient
     AND PATIENT.Team = $TEAM";
 $PTLIST = SQLManySQL($handDB, $qry, 'get rated pts');
  if ( count($PTLIST) < 1)
     { $ERR = 1;
     } else
     { $PARTABLE = '';
       foreach ($PTLIST as $PROW)
         {
           $PID = $PROW[0];
           $PNAM  = $PROW[1];
           $PARTABLE .= "<tr><td>$PNAM</td> " .
      " <td><input type='radio' name='particip' value='$PID' " .
      " onClick=\"javascript:if(this.checked)" .
      "{document.strict_view_assessor.whichpatient.value='$PID';}\">" .
      "</td>";
         };
     };

  if ($ERR)
     { $PARTABLE = "<tr><td colspan='2'>
        (No rating data exist for this team's scenarios!)</td></tr>";
     };

print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
  <script type='text/javascript'>
  <!--
```

```
  function CheckInput(myform)
    {
      if (myform.whichpatient.value < 1)
         { alert('Please select a patient scenario');
           return(false);
         };
      return (true);
    }
  //-->
  </script>
</head>
<body>
$MYHEADER
HTML0;

  if (! $ERR)
     { print "<p>Please select a patient scenario:";
     };

print <<<HTML3
<FORM name="strict_view_assessor"
      ACTION="strict_view_assessor.php"
      METHOD="POST"
      onSubmit="return CheckInput(this)" >
  <input type=hidden name="team" value="$TEAM">
  <input type=hidden name="whichpatient" value="0">

  <div align='center'>
  <table width='40%'>
  $PARTABLE
HTML3;

if (! $ERR)
   { print <<<HTML5
  <tr><td colspan='2'>
    <INPUT TYPE="submit" NAME="submit"
       VALUE="Confirm your selection...">
  </td></tr>
HTML5;
   };

print <<<HTML7
  </table></div>
</form>
<p>[<a href='strict_view.php'>
   Select another team</a>]
<p><a href='mainpage.php'>Return to main page</a>
</body></html>
```

```
HTML7;
```

3.17.3 strict_view_assessor.php

Finally, given a team and patient scenario, we list everyone who has assessed the patient scenario (registrars, nurses and assessors), with the option to select someone and then view their assessment. We submit the Team and Whichpatient values as hidden POST variables to the next script (*strict_view_rating.php*) so that we can return from there to here!

```
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_view_assessor.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                              'Select an assessment', 0);
 $TEAM = $_POST['team'];
 CheckCode($TEAM, 'Bad team selection (view)');
 $qry = "SELECT tName FROM TEAM WHERE team = $TEAM";
 list($TEAMNAME) = GetSQL($handDB, $qry, 'get team name.');
 $WHICHPATIENT = $_POST['whichpatient'];
 CheckCode($WHICHPATIENT, 'Bad pt(view)');
 # get a list of assessors, but associate with RATING!!
 # (there is only one rating per assessor)
 $qry = "SELECT RATING.rating, pdForename, pdSurname
   FROM PERSDATA, PATIENT, RATING
     WHERE RATING.Patient = PATIENT.patient
       AND RATING.Assessor = PERSDATA.Person
     AND Team = $TEAM
     AND WhichPatient = $WHICHPATIENT";
 $PARTICIP = array();
 $PARTICIP = SQLManySQL($handDB, $qry, 'get assessors');
 $PARTABLE = '';
 foreach ($PARTICIP as $PROW)
   { $FORE = $PROW[1];
     $SUR  = $PROW[2];
     $RATING = $PROW[0];
     $PARTABLE .= "<tr><td>$FORE $SUR </td> " .
      " <td><input type='radio' name='rater' value='$RATING' " .
      " onClick=\"javascript:if(this.checked)" .
      "{document.strict_view_rating.rating.value='$RATING';}\">" .
```

```
      "</td>";
   };

print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
  <script type='text/javascript'>
  <!--
  function CheckInput(myform)
    {
      if (myform.rating.value < 1)
         { alert('Please select an assessor!');
           return(false);
         };
      return (true);
    }
  //-->
  </script>
</head>
<body>
$MYHEADER
<h3>You chose Team $TEAMNAME</h3>
  <p>Select an assessor:
HTML0;

print <<<HTML3
<FORM name="strict_view_rating"
      ACTION="strict_view_rating.php"
      METHOD="POST"
      onSubmit="return CheckInput(this)" >
  <input type=hidden name="rating" value="0">
  <input type=hidden name="team" value="$TEAM">
  <input type=hidden name="teamname" value="$TEAMNAME">
  <input type=hidden name="whichpatient" value="$WHICHPATIENT">
  <div align='center'>
  <table width='40%'>
  <tr><td><i>Assessor</i></td>
      <td><i>Click here</i></td>
  </tr>
  $PARTABLE
  <tr><td colspan='2'>
   <INPUT TYPE="submit" NAME="submit" VALUE="Confirm your selection">
  </td></tr>
  </table></div>
</form>
<p>[<a href='strict_view_patient.php?teamval=$TEAM'>
  Choose another patient</a>]
```

```
<p><a href='mainpage.php'>Return to main page</a>
</body></html>
HTML3;
```

3.17.4 strict_view_rating.php

Here we display the selected assessment, given the ID of the rating. We also accept the POST variables *team* and *whichpatient*.

```
## Screen layout: ################################################
#   Assess a patient scenario:                                   #
#                                                                #
#  (Long table full of radio buttons goes here)                  #
#                                                                #
#   [submit]                                                     #
#                                                                #
#   [return to assessment menu]                                  #
#   [return to main menu]                                        #
################################################################
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_view_rating.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = 1;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER); # provides $USERKEY
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                             $MINUSERSTATUS, $MAXUSERSTATUS,
                          'Assessment data:', 0);

 $RATING = $_POST['rating'];
 CheckCode($RATING, 'Bad rating id');
 $TEAM = $_POST['team'];
 CheckCode ($TEAM, 'Hmm bad team');
 $TEAMNAME = $_POST['teamname'];
 Sanitise ($TEAMNAME);
 $WHICHPATIENT = $_POST['whichpatient'];
 CheckCode ($WHICHPATIENT, '?? patient');

 $qry = "SELECT PatientName,
          CONCAT(pdForename, ' ', pdSurname)
         FROM PERSDATA, PATIENT, RATING, WHICHPATIENT
          WHERE RATING.Assessor = PERSDATA.Person
           AND RATING.Patient = PATIENT.patient
           AND PATIENT.Whichpatient = WHICHPATIENT.whichpatient
           AND rating = $RATING";
   # note mySQL concatenation idiosyncrasy
 list ($SCENARIO, $RATER) =
    GetSQL($handDB, $qry, 'get scenario, rater');

 $qry = "SELECT ratText, rValue
```

```
          FROM ONERATING, RORDER
        WHERE ONERATING.Rorder = RORDER.rorder
         AND Rating = $RATING
        ORDER BY ONERATING.Order";
 $R = SQLManySQL ($handDB, $qry, 'get ratings');

 $BIGTBL = '';
 foreach ($R as $LN)
    { $txt = $LN[0];
      $vlu = $LN[1];
      $BIGTBL .= "<tr><td>$txt</td><td>$vlu</td></tr>\n";
      # here might intercalate 'dressing up'
    };

print <<<HTML0
<html> <head>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
  <script type='text/javascript'>
  <!--
  function CheckDeletion (myform)
   { if (myform.deletionpassword.value.length < 6)
        { alert('Invalid password');
          return(false);
        };
      if (! confirm('DELETE ENTIRE RECORD? Are you sure?'))
         { return (false);
         };
     // ideally should md5 the password.
     return(true); // stub.
   };
  //-->
  </script>
</head>
<body>
$MYHEADER
<h3>Team $TEAMNAME</3>

<h4>Rating of scenario $SCENARIO $TEAMNAME by '$RATER'</4>

<p><div align='center'>
 <table width='80%' border='3' cellpadding='3'> <!-- ugh -->
  <tr><td width='90%'><i>Item rated</i></td>
     <td width='10%'><i>Score</i></td>
  </tr>
  $BIGTBL
  <tr><td colspan='2'>
    <table width=100%>
```

```
     <tr>
      <td width='50%'>
       <FORM name="strict_view_assessor"
         ACTION="strict_view_assessor.php"
         METHOD="POST"
         onSubmit="return(true);" >
        <input type='hidden' name='team' value='$TEAM'>
        <input type='hidden' name='whichpatient'
               value='$WHICHPATIENT'>
        <INPUT TYPE="submit" NAME="submit"
               VALUE="Back to previous page..">
       </FORM>
      </td>
      <td width='50%'>
       <FORM name="strict_aagh"
         ACTION="strict_aagh.php"
         METHOD="POST"
         onSubmit="return CheckDeletion(this)" >
        Enter DELETION PASSWORD:
        <input type='password'
            name='deletionpassword' size='16' value=''><br>
        <INPUT TYPE="submit"
            NAME="submit" VALUE="DELETE these data!">
        <input type='hidden'
            name='rating2delete' value='$RATING'>
       </FORM>
      </td>
     </tr>
    </table>
   </td>
  </tr>
 </table>
</div>
HTML0;

print <<<HTML3
  </table></div>
<p>[<a href='mainpage.php'>Return to main page</a>]
</body></html>
HTML3;
```

3.17.5 strict_aagh.php

Here we delete an entire RATING entry (with all associated ONERATING entries). The password must be correct.

```
## Screen layout: ##############################################
#                                                              #
#   Rating deleted!                                            #
#                                                              #
#   [Return to main menu]                                      #
#                                                              #
################################################################
 header( 'Cache-control: no-cache' );
# $THISPAGE = 'strict_aagh.php';
 require_once('ValidFx.php'); # our login validation script
 $MINUSERSTATUS = strict_ADMINISTRATOR_MIN;
 $MAXUSERSTATUS = EVERYONE;
 $success = validate_login(SHOW_USER);
 if (! $success)
    { exit();
    };
 list ($MYHEADER, $USERSTATUS) = strictHeaderGreet($success,
                              $MINUSERSTATUS, $MAXUSERSTATUS,
                              "Delete rating!", 0);
print <<<HTML0
<html>
<head><title>Delete rating</title>
<META http-equiv=content-type content=text/html;charset=utf-8>
<LINK href="css/strictstyle.css" type=text/css rel=stylesheet>
</head>
<body>
$MYHEADER
HTML0;

$rating = $_POST['rating2delete'];
CheckCode ($rating, 'Aagh. bad rating code');
$password = $_POST['deletionpassword'];

# print "<p>Debug: password is '$password'";

if ($password == 'Deletion Password') # [later must md5 me!]
   { $qry = "DELETE FROM ONERATING WHERE
               Rating = $rating";
     DoSQL($handDB, $qry, 'delete onerating');
     $qry = "DELETE FROM RATING WHERE
               rating = $rating";
     DoSQL($handDB, $qry, 'delete rating');
     # mySQL auto-commits [ugh]
     # might check for success!
```

```
    print "<p>Entire record deleted as requested.";
  } else
  { print "<p>Nice try, but I couldn't comply with your request!";
  };

print <<<HTML3
<p>[<a href='mainpage.php'>Return to main page</a>]
</body></html>
HTML3;
```

3.18 Ancillary code

The following ancillary functions are used throughout the database.

3.18.1 PrintPoplist

PrintPoplist obtains the desired primary key (the first item specified in the $SQL SELECT statement) and $elems extra items. ($elems is not an explicit parameter). The $elems extra items are concatenated into a single string! Each (key; $elems concatenation) pair is printed as an *option* within an HTML ¡SELECT¿ statement.

```
function PrintPoplist ($handDB, $listname, $query )
{
  print( "<select name='$listname' >
         <option selected value='0'>?</option>\n");
  $handQ = mysql_query($query, $handDB);
  if (! is_resource($handQ))
    { # debug:
      print ( "<br> SQL QUERY error: " . mysql_error() );
    };
  $count = 0;
  while ($row = mysql_fetch_array($handQ, MYSQL_NUM))
    { $mykey = $row[0];
      $myvalue = ConcatenateArray( array_slice($row, 1) );
        # concatenate remaining elements
      print ( "<option value='$mykey'>$myvalue</option>\n" );
      $count += 1;
    };
  print ('</select>');
  return($count);
}
```

3.18.2 TextPoplist

TextPoplist is like PrintPoplist but returns a string for later insertion into HTML text!

```
function TextPoplist ($handDB, $listname, $query )
{
  $opt = '';
  $opt = $opt . sprintf( "<select name='$listname' >
         <option selected value='0'>?</option>\n"); # or use %s :-)

  $handQ = mysql_query($query, $handDB);
  if (! is_resource($handQ))
     { # debug:
       print ( "<br> SQL QUERY error: " . mysql_error() );
     };
  $count = 0;
  while ($row = mysql_fetch_array($handQ, MYSQL_NUM))
    { $mykey = $row[0];
      $myvalue = ConcatenateArray( array_slice($row, 1) );
        # concatenate remaining elements
      $opt = $opt . sprintf (
        "<option value='%d'>%s</option>\n", $mykey, $myvalue );
      $count += 1;
    };
  $opt = $opt . sprintf ('</select>');
  return($opt);
}
```

3.18.3 TextPoplistSelected

This function is identical to TextPoplist but it takes an additional parameter — the key of the *selected* item which is then flagged as selected.

```
function TextPoplistSelected ($handDB, $listname, $query, $k)
{ $opt = '';
  $s = '';
  if ($k < 1)
     { $s = 'selected';
     };
  $opt = $opt . sprintf( "<select name='$listname' >
  <option $s value='0'>?</option>\n");
  $handQ = mysql_query($query, $handDB);
  $count = 0;
  while ($row = mysql_fetch_array($handQ, MYSQL_NUM))
    { $mykey = $row[0];
      $myvalue = ConcatenateArray( array_slice($row, 1) );
        # concatenate remaining elements
```

```
      $s = '';
      if ($k == $mykey)
         { $s = 'selected';
         };
      $opt = $opt . sprintf (
        "<option $s value='%d'>%s</option>\n", $mykey, $myvalue );
      $count += 1;
    };
  $opt = $opt . sprintf ('</select>');
  return($opt);
}
```

3.18.4 ConcatenateArray

Join all array elements into a single string, separating elements with blanks. There is a terminal blank.

```
function ConcatenateArray ($ary)
{ $opt = '';
  foreach ($ary as $itm)
    { $opt .= "$itm ";
    };
  return($opt);
};
```

3.18.5 strictHeaderGreet

Return text string, *and* add in header with image!

```
function strictHeaderGreet($success,
                           $MINUSERSTATUS,
                           $MAXUSERSTATUS,
                           $title,
                           $previous)  # $previous is 0 or 1.
{ $matches = '';
  list($USER, $USERSTATUS, $OLDPAGE) = PullOutUserInfo($success);
if (  ($USERSTATUS < $MINUSERSTATUS)
    ||($USERSTATUS > $MAXUSERSTATUS)
   )
   { readfile ('failedaccess.htm');
     exit();  # fail
   };
# print ("<br>Debug: user status is $USERSTATUS");
$OLD='';
if ($previous)
   { $OLD = "(previously at <a href='$OLDPAGE'>$OLDPAGE</a>)";
```

```
   };
$txt[0] = "<p><div align=center>
<table width=95%>
<tr><td width=150 align='center'>
<img src='images/tinylogo.png'></td>
<td width=4><img width=4 height=64 src='images/vbar.png'></td>
<td><h2>$title</h2>
  <span class='info'>User: $USER $OLD</span></td>
</tr></table>
<hr>
</div>";

$txt[1] = $USERSTATUS;
 return ($txt);
}
```

3.18.6 PullOutUserInfo

Pull out user and associated details. String format is as shown above (Section **??**). Returns a list of the three items in that order.

```
function PullOutUserInfo ($userstring)
{
  $matches = '';
  if (! preg_match ( '/USER="(.+)"\|STATUS="(.+)"\|OLDPAGE="(.*)"/',
                     $userstring, $matches) )
     { print (
 "<br>Error. Bad user data. Terminated. String was &lt;$userstring&gt;" );
       exit();
     };
  array_shift($matches);  # remove first element (is whole match)
  return ($matches);
}
```

3.18.7 WhatYearIsIt

Simply retrieve the current year!

```
function WhatYearIsIt()
{ return ( date("Y") );
};
```

3.18.8 CheckCode

Check that item is numeric.

```
function CheckCode(&$code, $msg)
{  if (! preg_match ( '/^\s*(\d+)\s*$/', $code, $vals) )
      { print ( "<br>Bad data item &lt;$code&gt;.
                 <p> $msg. ");
        readfile('badcode.htm');
        exit();
      };
  $code = $vals[1]; # trim whitespace, just in case.
};
```

Similar is CheckCodeNull, which however doesn't exit, merely returning NULL if the match fails:

```
function CheckCodeNull(&$code)
{  if (! preg_match ( '/^\s*(\d+)\s*$/', $code, $vals) )
      { $code = 'NULL';
      };
  $code = $vals[1]; # trim whitespace
};
```

3.18.9 FetchKey

The following should be atomic AND block other threads. We must fix this up [fix me!]

```
function FetchKey ($handDB, $ky)
{
  # start atomic:
  list($keyval) = GetSQL($handDB,
                    "SELECT u$ky FROM UIDS WHERE uids = 1",
                    "get key value");
  $keyval ++;
  DoSQL($handDB,
         "UPDATE UIDS SET u$ky = $keyval WHERE uids = 1",
         "set new key value");
  #end atomic.

  $keyval --;
  return ($keyval);
}
```

3.18.10 GetSQL

Obtain a single row from the database as an array.

```
function GetSQL ($handDB, $qry, $message)
{
```

```
  if ($DEBUGGING)
     { print ("<br>DEBUGGING: Query was &lt;$qry&gt; ==&gt; ");
     };
  $handQ = mysql_query($qry, $handDB);
  if (! is_resource($handQ))
     { print ( "<br>SQL error ($message):" . mysql_error() );
       exit(); # this is serious: FAIL!
     };
  # problem: SELECT MAX(mysession), sTrainee FROM MYSESSION
  #    WHERE sAssessor = 3 GROUP BY sTrainee
  # IF there are no entries in MYSESSION this will not fail,
  # but also will not return a valid handle in $handQ. Thus:
  if (   (! is_resource($handQ)) # is a resource!
      || (! mysql_num_rows($handQ))
     )
     { return('');  # NULL
     };
  $row = mysql_fetch_array($handQ, MYSQL_NUM);
  if ($DEBUGGING)
     {
      print (" ==&gt;");
      print_r ($row);  # DEBUGGING
     };
  return ($row); # return array!
}
```

3.18.11 DoSQL

```
function DoSQL ($handDB, $qry, $message)
{
  $handQ = mysql_query($qry, $handDB);
#  if (! is_resource($handQ)) # noooooo!
#     { print ( "<br>SQL error ($message):" . mysql_error() );
#       exit(); # this is serious: FAIL!
#     };
  if ($DEBUGGING)
     { print ("<br>DEBUGGING: Stmt to execute was &lt;$qry&gt; ");
     };
  # simply return.
 }
```

3.18.12 SQLManySQL

Obtain array of arrays (all data).

```
function SQLManySQL ($handDB, $qry, $msg)
{
```

```
  $handQ = mysql_query($qry, $handDB);
  if (! is_resource($handQ))
    { # debug:
      print ( "<br> SQL QUERY error: ($msg) " . mysql_error() );
      exit(); # serious failure. [???]
    };
  $outAry = array();
  if (! is_resource($handQ))
     {
       print ( "<br> SQL QUERY error: ($msg) " . mysql_error() );
       return ($outAry);
     };
  $i = 0;
  while ($row = mysql_fetch_array($handQ, MYSQL_NUM))
    { $outAry[$i] = $row; # array of arrays
      $i ++;
    };
  return ($outAry);
}
```

3.18.13 FetchSurname

Given a person's ID, obtain their surname.

```
function FetchSurname ($handDB, $id)
{
  $qry = "SELECT MAX(persdata) FROM PERSDATA
           WHERE Person = $id AND
             pdSurname IS NOT NULL";
  list($pd) = GetSQL ($handDB, $qry, "fetch key for surname");
  $qry = "SELECT pdSurname FROM PERSDATA WHERE persdata = $pd";
  list($surname) = GetSQL($handDB, $qry, "fetch surname");
  return ($surname);
}
```

3.18.14 FetchForename

In a similar fasion to **FetchSurname**, obtain a person's forename.

```
function FetchForename ($handDB, $id)
{
  $qry = "SELECT MAX(persdata) FROM PERSDATA
           WHERE Person = $id AND
             pdForename IS NOT NULL";
  list($pd) = GetSQL ($handDB, $qry, "fetch key for forename");
  $qry = "SELECT pdForename FROM PERSDATA WHERE persdata = $pd";
  list($forename) = GetSQL($handDB, $qry, "fetch forename");
```

```
  return ($forename);
}
```

3.18.15 FetchRole

The following can be made far simpler if we accept that in this database a person will only ever have one role.

```
function FetchRole ($handDB, $id)
{
  $qry = "SELECT MAX(persdata) FROM PERSDATA
           WHERE Person = $id AND
             PersonRole IS NOT NULL";
  list($pd) = GetSQL ($handDB, $qry, "fetch key for role");
  $qry = "SELECT PERSONROLE.rText FROM PERSDATA,PERSONROLE
            WHERE PERSDATA.PersonRole = PERSONROLE.personrole AND
             PERSDATA.persdata = $pd";
  list($PersonRole) = GetSQL($handDB, $qry, "fetch role");
  return ($PersonRole);
}
```

3.18.16 Sanitise

Clean up data input:

```
function Sanitise (&$txt)
{
  # first remove whitespace at start, end:
  $matches = '';
  preg_match ( '/^\s*(.*)\s*$/', $txt, $matches);
  $txt = $matches[1]; # get $1; or use ltrim, rtrim.
  $txt = preg_replace( '/\|/', '', $txt); # get rid of pipes
  $txt = preg_replace( '/\`/', '', $txt); # and backticks
  $txt = preg_replace( '/"/', "'", $txt); #double quote -> single
  $txt = preg_replace( "/'/", "''", $txt);#duplicate any quote
  if ($DEBUGGING)
     {
       print "<br><b>Sanitised:</b> &lt;$txt&gt;";
     };
}  # return with altered $txt (It's by reference)
```

3.18.17 Flatten

Flatten: See http://nz.php.net/array ; Given an array of many dimensions, flatten to a unidimensional array, which is returned as a new array. The problem with the following is the deprecated Call-time pass-by-reference!

```
function Flatten($myarray)
{
  $simflat = array();
  array_walk($myarray, 'flatten_array', &$simflat);
  # see flatten_array fx below!
  return ($simflat);
}

function flatten_array($value, $key, &$array)
{
  if (!is_array($value))
     { array_push($array,$value);
     } else
     { array_walk($value, 'flatten_array', &$array);
     };
}

function Flatten($myarray)
{
  $Aout = array();
  foreach ($myarray as $a)
    { if (is_array($a))
         { $J = Flatten($a);
           foreach ($J as $i)
             { array_push ($Aout, $i);
             };
         } else
         { array_push ($Aout, $a);
         }
    };
  return ($Aout);
}
```

3.18.18 PrintDetailTable

```
function PrintDetailTable($data, $cols)
{ # cols is number of colums, $data = array of data.
  foreach ($data as $d)
    {
      # print ('<br>Deeper debug: ');
      # print_r ($d);
      print ('<tr>');
      $i = 0;
      while ($i < $cols)
        { print ('<td>');
          if (strlen ($d[$i]) < 1)
             { print '?';  # should work if NULL
             } else
```

```
            { print $d[$i];
            };
          print ('</td>');
          $i += 1;
        };
      print ('</tr>');
    };
}
```

3.18.19 ValidatePassword

```
function ValidatePassword($pswd, $minlen)
{
  if (strlen ($pswd) < $minlen)  # minimum length
     { return (0);
     };

  if (preg_match ( "/[\\\'\'\|]/", $pswd))  # check for illegals
     { return (0);
     };
  return (1); # valid
}
```

3.18.20 PrintActiveUserlist

```
function PrintActiveUserlist($handDB, $link)
{
  print <<<HTMLpau
  <p><div align='center'>
  <table width='90%' border='2'>
  <tr><td><i>Forename</i></td>
      <td><i>Surname</i></td>
      <td><i>Role</i></td>
      <td><i>Started</i></td>
      <td><i>Click here</i></td></tr>
HTMLpau;

  $activeusers = array();
  $qry = 'SELECT distinct Person FROM PERSDATA
     WHERE PersonRole > 0';
  $activeusers = SQLManySQL($handDB, $qry, 'get active users');
  $activeusers = Flatten($activeusers); #
  $users = GetLinkedUserDetails($handDB, $activeusers, $link);
   MyDoubleSort($users, 1, 0); #sort by surname, then forename!
   PrintDetailTable($users, 5); # print table with 5 columns
}
```

3.18.21 MyDoubleSort

```
# Given 2-D array of rows, sort, first by the first column,
#  then by the second (if first column entries are equal):
# similar to MySort.

function MyDoubleSort(&$arry, $col1, $col2)
{
  GLOBAL $CLUMSY_INDEX;  #
  $CLUMSY_INDEX = $col1;
  GLOBAL $CLUMSY_INDEX2;  #
  $CLUMSY_INDEX2 = $col2;
  usort ($arry, 'my_dblsort');
}

function my_dblsort ($first, $second)
{
  GLOBAL $CLUMSY_INDEX;
  GLOBAL $CLUMSY_INDEX2;
  $r = strcmp ($first[$CLUMSY_INDEX], $second[$CLUMSY_INDEX]);
  if ($r)  # if nonzero (not the same):
     { return ($r);
     };
  # two items are equal, so compare sub-items:
  return ( strcmp ($first[$CLUMSY_INDEX2], $second[$CLUMSY_INDEX2]) );
}
```

3.18.22 GetLinkedUserDetails

The following is like GetUserDetails from strict_admin_personadded.php *but* the added parameter $link is something along the lines of:

```
 "<a href='strict_do_editlogon.php?editlogon=USERID'>Go</a>";
```

A required component is the text string USERID, replaced with the relevant key contained in the list $link. The URL represents a GET (not POST) to the page specified e.g. strict_do_editlogon

```
function GetLinkedUserDetails($handDB, $unames, $link)
{ # $unames is array of IDs (key of PERSON table)
  # this fx is cumbersome and slow.
  # returns array of arrays, each subarray containing
  # forename, surname, role, date started practice
  # (as year-01-01) and link URL (contains ID).

  $opt = array();
```

```
  $i = 0;
  foreach ($unames as $p)
    { $detl = array();
      $detl[0] = FetchForename ($handDB, $p);
      $detl[1] = FetchSurname($handDB, $p);
      $detl[2] = FetchRole ($handDB, $p);
      list ($detl[3]) = GetSQL ($handDB, "SELECT FirstQualified
        FROM PERSON WHERE person = $p",
        'GET year of 1st qualification');
      $detl[4] = $link;
      # use str_replace rather than regex:
      $detl[4] = str_replace ('USERID', $p, $detl[4]);
      # might check for success.
      $opt[$i] = $detl;
      $i += 1;
    };
  return ($opt);
}
```

3.18.23 Study day-related functions

```
function PrintStudydayList($handDB, $link)
{
  print <<<HTMLpdl
  <p><div align='center'>
  <table width='90%' border='2'>
  <tr><td><i>Study day (ID)</i></td>
      <td><i>Date</i></td>
      <td><i>Comment</i></td>
      <td><i>Click here</i></td></tr>
HTMLpdl;

  $studydays = array();
  $qry = "SELECT CONCAT(dName, ' (', studyday, ')'), studydayDate, dText
            FROM STUDYDAY WHERE studyday > 0";
  $studydays = SQLManySQL($handDB, $qry, 'get studydays');
  $studydays = FixupStudydayLink($handDB, $studydays, $link);
  PrintDetailTable($studydays, 4); # print table with 4 columns
}

function FixupStudydayLink($handDB, $data, $link)
{ # Very similar to FixupTeamLink. (Share rtn??)
  $opt = array();
  $i = 0;
  foreach ($data as $p)
    { $p[3] = $link;
      preg_match ( '/\((\d+)\)/', $p[0], $match); # get (ID)
      $p[3] = str_replace ('STUDYDAYID', $match[1], $p[3]);
```

```
      $opt[$i] = $p;
      $i += 1;
    };
  return($opt);
}
```

3.18.24 Other PHP functions

```
# unfortunately implode will not implode nested arrays, thus:
function DeepImplode ($ary)
{ $opt = '';
  foreach ($ary as $itm)
    { $val = $itm[0];
      $opt = "$opt$val,";
    };
  $opt = rtrim ($opt, ","); # get rid of terminal comma
  return ($opt);
}
```

Get Julian date (floating point) from year, month, day etc.

```
function Julian($fy, $fm, $fd, $fh, $fmi, $fs, $ff)
{ $LOCALTIMEOFFSET = 0;
  $DAYLIGHTSAVING = 0;
  $f= 367*$fy - (int)(7*($fy+(int)(($fm+9)/12))/4)
              - (int)(3*((int)(($fy+($fm-9)/7)/100)+1)/4)
              + (int)(275*$fm/9)+$fd+1721028.5
              - ($LOCALTIMEOFFSET+$DAYLIGHTSAVING)/24
              + ($fh + ($fmi + ($fs+ "0.$ff")/60)/60)/24;
  return ($f);
}
```

Conversely, given Julian date, return Gregorian year, month and day.

```
function Gregorian ($jd)
{
 $EPSILON = 0.000001;
 $LOCALTIMEOFFSET = 0;
 $DAYLIGHTSAVING = 0; # hack

 $jd += ($LOCALTIMEOFFSET+$DAYLIGHTSAVING)/24;
 $jd += $EPSILON; # 1 != 0.99999999...99
$Z = floor($jd - 1721118.5);
$R = $jd - 1721118.5 - $Z;
$G = $Z - 0.25;
$A = floor($G / 36524.25);
$B = $A - floor($A / 4);
$year = floor(($B+$G) / 365.25);
```

```
$C = $B + $Z - floor(365.25 * $year);
$month = (int)((5 * $C + 456) / 153);
$day = $C - (int)((153 * $month - 457) / 5) + $R;
if ($month > 12)
   { $year = $year + 1;
     $month = $month - 12;
   };
$ar = array();
$ar[0] = $year;
$ar[1] = $month;
$ar[2] = (int)($day);
return ($ar);
}
```

3.18.25 Timetable templates

The templates shown in Tables 1 and 2 are boringly similar to one another. They also act as CSV files. We will add the frill (when we import these files) of stripping off the comments generated by DogWagger. In Version 0.61 we added two more templates (See tables 3 and 4 to cater for the case where the ISB scenarios are swapped around, in keeping with the value of ISBnature in the TEAM table.

Table 1: The first CSV Timetable template file

```
   ,DATE $DATE,Co-ordinator $TEACHER1A,,DATE $DATE,Co-ordinator $TEACHER1B,
,$TEAMA,,,$TEAMB,,
Time,Activity,Location,Instructors,Activity,Location,Instructors
0745-0815,Simulation centre and ICU orientation,Advanced Clinical Skills Centre,,Simulation centre and ICU orientation,Advanced Clinical Skills Centre,
0815-0830,Paperwork completion,Advanced Clinical Skills Centre,$TEACHER1A,Paperwork completion,Advanced Clinical Skills Centre,$TEACHER1B
0830-0845,Paperwork completion,Breakout room 1,,$PT1B, Intensive Care,
0845-0900,$PT1A, Intensive Care,,Debrief and assessment,Breakout room 2,$TEACHER2B
0900-0920,Debrief and assessment,Breakout room 1,$TEACHER2A,$PT2B, Intensive Care,
0925-0940,$PT2A, Intensive Care,,Debrief and assessment,Breakout room 2,$TEACHER3B
0940-1000,Debrief and assessment,Breakout room 1,$TEACHER3A,Morning tea,Upstairs,
1000-1015,Morning tea,Upstairs,,Morning tea,Upstairs,
1020-1100,Defibrillator skill station ,Breakout room 2,$TEACHER4A,Airway skill station,Airway Laboratory,$TEACHER6B
1100-1140,Lecture,Lecture Theatre,$TEACHER5A,Lecture,Lecture Theatre,$TEACHER5B
1140-1220,Airway skill station,Airway Laboratory,$TEACHER6A,Defibrillator skill station ,Breakout room 2,$TEACHER4B
1220-1300,Lunch,Upstairs,,Lunch,Upstairs,
1300-1315,Peter Smeaton, Intensive Care,$TEACHER7A,Problem Centred Learning 1,Breakout room 2,$TEACHER9B
1315-1330,Debrief,Breakout room 1,$TEACHER8A,Problem Centred Learning 1,Breakout room 2,
1330-1345,Joan Digger, Intensive Care,,Problem Centred Learning 2,Breakout room 2,$TEACHER9B
1345-1400,Debrief,Breakout room 1,,Problem Centred Learning 2,Breakout room 2,
1400-1415,Julia Robertson, Intensive Care,,Problem Centred Learning 3,Breakout room 2,$TEACHER9B
1415-1430,Debrief,Breakout room 1,,Problem Centred Learning 3,Breakout room 2,
1430-1445,Afternoon tea,Upstairs,,Afternoon tea,Upstairs,
1445-1500,Problem Centred Learning 1,Breakout room 1,$TEACHER9A,Fredrica Fetherstone, Intensive Care,$TEACHER7B
1500-1515,Problem Centred Learning 1,Breakout room 1,,Debrief,Breakout room 2,$TEACHER8B
1515-1530,Problem Centred Learning 2,Breakout room 1,$TEACHER9A,Stephan Irwin, Intensive Care,
1530-1545,Problem Centred Learning 2,Breakout room 1,,Debrief,Breakout room 2,
1545-1600,Problem Centred Learning 3,Breakout room 1,$TEACHER9A,Victor Hemingway, Intensive Care,
1600-1615,Problem Centred Learning 3,Breakout room 1,,Debrief,Breakout room 2,
1615-1630,Mid afternoon break,Upstairs,,Mid afternoon break,Upstairs,
1630-1645,$PT3A, Intensive Care,,Mid afternoon break,Upstairs,
1645-1705,Debrief and assessment, Breakout room 1,$TEACHER10A,$PT3B, Intensive Care,
1705-1720,$PT4A, Intensive Care,,Debrief and assessment,Breakout room 2,$TEACHER10B
1720-1740,Debrief and assessment,Breakout room 1,$TEACHER11A,$PT4B, Intensive Care,
1740-1800,Evaluation and Close,Breakout room 1,,Debrief and assessment,Breakout room 2,$TEACHER11B
1800-1820,,,,Evaluation and Close,Breakout room 2,
```

Table 2: The second CSV Timetable template file

```
   ,DATE $DATE,Co-ordinator $TEACHER1A,,DATE $DATE,Co-ordinator $TEACHER1B,
,$TEAMA,,,$TEAMB,,
Time,Activity,Location,Instructors,Activity,Location,Instructors
0745-0815,Simulation centre and ICU orientation,Advanced Clinical Skills Centre,,Simulation centre and ICU orientation,Advanced Clinical Skills Centre,
0815-0830,Paperwork completion,Advanced Clinical Skills Centre,$TEACHER1A,Paperwork completion,Advanced Clinical Skills Centre,$TEACHER1B
0830-0845,Paperwork completion,Breakout room 1,,$PT1B, Intensive Care,
0845-0900,$PT1A, Intensive Care,,Debrief and assessment,Breakout room 2,$TEACHER2B
0900-0920,Debrief and assessment,Breakout room 1,$TEACHER2A,$PT2B, Intensive Care,
0925-0940,$PT2A, Intensive Care,,Debrief and assessment,Breakout room 2,$TEACHER3B
0940-1000,Debrief and assessment,Breakout room 1,$TEACHER3A,Morning tea,Upstairs,
1000-1015,Morning tea,Upstairs,,Morning tea,Upstairs,
1020-1100,Defibrillator skill station ,Breakout room 2,$TEACHER4A,Airway skill station,Airway Laboratory,$TEACHER6B
1100-1140,Lecture,Lecture Theatre,$TEACHER5A,Lecture,Lecture Theatre,$TEACHER5B
1140-1220,Airway skill station,Airway Laboratory,$TEACHER6A,Defibrillator skill station ,Breakout room 2,$TEACHER4B
1220-1300,Lunch,Upstairs,,Lunch,Upstairs,
1300-1315,Problem Centred Learning 1,Breakout room 1,$TEACHER9A,Fredrica Fetherstone, Intensive Care,$TEACHER7B
1315-1330,Problem Centred Learning 1,Breakout room 1,,Debrief,Breakout room 2,$TEACHER8B
1330-1345,Problem Centred Learning 2,Breakout room 1,$TEACHER9A,Stephan Irwin, Intensive Care,
1345-1400,Problem Centred Learning 2,Breakout room 1,,Debrief,Breakout room 2,
1400-1415,Problem Centred Learning 3,Breakout room 1,$TEACHER9A,Victor Hemingway, Intensive Care,
1415-1430,Problem Centred Learning 3,Breakout room 1,,Debrief,Breakout room 2,
1430-1445,Afternoon tea,Upstairs,,Afternoon tea,Upstairs,
1445-1500,Peter Smeaton, Intensive Care,$TEACHER7A,Problem Centred Learning 1,Breakout room 2,$TEACHER9B
1500-1515,Debrief,Breakout room 1,$TEACHER8A,Problem Centred Learning 1,Breakout room 2,
1515-1530,Joan Digger, Intensive Care,,Problem Centred Learning 2,Breakout room 2,$TEACHER9B
1530-1545,Debrief,Breakout room 1,,Problem Centred Learning 2,Breakout room 2,
1545-1600,Julia Robertson, Intensive Care,,Problem Centred Learning 3,Breakout room 2,$TEACHER9B
1600-1615,Debrief,Breakout room 1,,Problem Centred Learning 3,Breakout room 2,
1615-1630,Mid afternoon break,Upstairs,,Mid afternoon break,Upstairs,
1630-1645,$PT3A, Intensive Care,,Mid afternoon break,Upstairs,
1645-1705,Debrief and assessment, Breakout room 1,$TEACHER10A,$PT3B, Intensive Care,
1705-1720,$PT4A, Intensive Care,,Debrief and assessment,Breakout room 2,$TEACHER10B
1720-1740,Debrief and assessment,Breakout room 1,$TEACHER11A,$PT4B, Intensive Care,
1740-1800,Evaluation and Close,Breakout room 1,,Debrief and assessment,Breakout room 2,$TEACHER11B
1800-1820,,,,Evaluation and Close,Breakout room 2,
```

Table 3: The 3rd CSV Timetable template file (swopped ISB)

```
   ,DATE $DATE,Co-ordinator $TEACHER1A,,DATE $DATE,Co-ordinator $TEACHER1B,
,$TEAMA,,,$TEAMB,,
Time,Activity,Location,Instructors,Activity,Location,Instructors
0745-0815,Simulation centre and ICU orientation,Advanced Clinical Skills Centre,,Simulation centre and ICU orientation,Advanced Clinical Skills Centre,
0815-0830,Paperwork completion,Advanced Clinical Skills Centre,$TEACHER1A,Paperwork completion,Advanced Clinical Skills Centre,$TEACHER1B
0830-0845,Paperwork completion,Breakout room 1,,$PT1B, Intensive Care,
0845-0900,$PT1A, Intensive Care,,Debrief and assessment,Breakout room 2,$TEACHER2B
0900-0920,Debrief and assessment,Breakout room 1,$TEACHER2A,$PT2B, Intensive Care,
0925-0940,$PT2A, Intensive Care,,Debrief and assessment,Breakout room 2,$TEACHER3B
0940-1000,Debrief and assessment,Breakout room 1,$TEACHER3A,Morning tea,Upstairs,
1000-1015,Morning tea,Upstairs,,Morning tea,Upstairs,
1020-1100,Defibrillator skill station ,Breakout room 2,$TEACHER4A,Airway skill station,Airway Laboratory,$TEACHER6B
1100-1140,Lecture,Lecture Theatre,$TEACHER5A,Lecture,Lecture Theatre,$TEACHER5B
1140-1220,Airway skill station,Airway Laboratory,$TEACHER6A,Defibrillator skill station ,Breakout room 2,$TEACHER4B
1220-1300,Lunch,Upstairs,,Lunch,Upstairs,
1300-1315,Fredrica Fetherstone, Intensive Care,$TEACHER7A,Problem Centred Learning 1,Breakout room 2,$TEACHER9B
1315-1330,Debrief,Breakout room 1,$TEACHER8A,Problem Centred Learning 1,Breakout room 2,
1330-1345,Stephan Irwin, Intensive Care,,Problem Centred Learning 2,Breakout room 2,$TEACHER9B
1345-1400,Debrief,Breakout room 1,,Problem Centred Learning 2,Breakout room 2,
1400-1415,Victor Hemingway, Intensive Care,,Problem Centred Learning 3,Breakout room 2,$TEACHER9B
1415-1430,Debrief,Breakout room 1,,Problem Centred Learning 3,Breakout room 2,
1430-1445,Afternoon tea,Upstairs,,Afternoon tea,Upstairs,
1445-1500,Problem Centred Learning 1,Breakout room 1,$TEACHER9A,Peter Smeaton, Intensive Care,$TEACHER7B
1500-1515,Problem Centred Learning 1,Breakout room 1,,Debrief,Breakout room 2,$TEACHER8B
1515-1530,Problem Centred Learning 2,Breakout room 1,$TEACHER9A,Joan Digger, Intensive Care,
1530-1545,Problem Centred Learning 2,Breakout room 1,,Debrief,Breakout room 2,
1545-1600,Problem Centred Learning 3,Breakout room 1,$TEACHER9A,Julia Robertson, Intensive Care,
1600-1615,Problem Centred Learning 3,Breakout room 1,,Debrief,Breakout room 2,
1615-1630,Mid afternoon break,Upstairs,,Mid afternoon break,Upstairs,
1630-1645,$PT3A, Intensive Care,,Mid afternoon break,Upstairs,
1645-1705,Debrief and assessment, Breakout room 1,$TEACHER10A,$PT3B, Intensive Care,
1705-1720,$PT4A, Intensive Care,,Debrief and assessment,Breakout room 2,$TEACHER10B
1720-1740,Debrief and assessment,Breakout room 1,$TEACHER11A,$PT4B, Intensive Care,
1740-1800,Evaluation and Close,Breakout room 1,,Debrief and assessment,Breakout room 2,$TEACHER11B
1800-1820,,,,Evaluation and Close,Breakout room 2,
```

Table 4: The 4th CSV Timetable template file (swopped ISB)

```
   ,DATE $DATE,Co-ordinator $TEACHER1A,,DATE $DATE,Co-ordinator $TEACHER1B,
,$TEAMA,,,$TEAMB,,
Time,Activity,Location,Instructors,Activity,Location,Instructors
0745-0815,Simulation centre and ICU orientation,Advanced Clinical Skills Centre,,Simulation centre and ICU orientation,Advanced Clinical Skills Centre,
0815-0830,Paperwork completion,Advanced Clinical Skills Centre,$TEACHER1A,Paperwork completion,Advanced Clinical Skills Centre,$TEACHER1B
0830-0845,Paperwork completion,Breakout room 1,,$PT1B, Intensive Care,
0845-0900,$PT1A, Intensive Care,,Debrief and assessment,Breakout room 2,$TEACHER2B
0900-0920,Debrief and assessment,Breakout room 1,$TEACHER2A,$PT2B, Intensive Care,
0925-0940,$PT2A, Intensive Care,,Debrief and assessment,Breakout room 2,$TEACHER3B
0940-1000,Debrief and assessment,Breakout room 1,$TEACHER3A,Morning tea,Upstairs,
1000-1015,Morning tea,Upstairs,,Morning tea,Upstairs,
1020-1100,Defibrillator skill station ,Breakout room 2,$TEACHER4A,Airway skill station,Airway Laboratory,$TEACHER6B
1100-1140,Lecture,Lecture Theatre,$TEACHER5A,Lecture,Lecture Theatre,$TEACHER5B
1140-1220,Airway skill station,Airway Laboratory,$TEACHER6A,Defibrillator skill station ,Breakout room 2,$TEACHER4B
1220-1300,Lunch,Upstairs,,Lunch,Upstairs,
1300-1315,Problem Centred Learning 1,Breakout rm 1,$TEACHER9A,Peter Smeaton,Intensive Care,$TEACHER7B
1315-1330,Problem Centred Learning 1,Breakout room 1,,Debrief,Breakout room 2,$TEACHER8B
1330-1345,Problem Centred Learning 2,Breakout rm 1,$TEACHER9A,Joan Digger,Intensive Care,
1345-1400,Problem Centred Learning 2,Breakout room 1,,Debrief,Breakout room 2,
1400-1415,Problem Centred Learning 3,Breakout rm 1,$TEACHER9A,Julia Robertson,Intensive Care,
1415-1430,Problem Centred Learning 3,Breakout room 1,,Debrief,Breakout room 2,
1430-1445,Afternoon tea,Upstairs,,Afternoon tea,Upstairs,
1445-1500,Peter Smeaton,Intensive Care,$TEACHER7A,Problem Centred Learning 1,Breakout rm 2,$TEACHER9B
1500-1515,Debrief,Breakout room 1,$TEACHER8A,Problem Centred Learning 1,Breakout room 2,
1515-1530,Joan Digger,Intensive Care,,Problem Centred Learning 2,Breakout rm 2,$TEACHER9B
1530-1545,Debrief,Breakout room 1,,Problem Centred Learning 2,Breakout room 2,
1545-1600,Julia Robertson,Intensive Care,,Problem Centred Learning 3,Breakout rm 2,$TEACHER9B
1600-1615,Debrief,Breakout room 1,,Problem Centred Learning 3,Breakout room 2,
1615-1630,Mid afternoon break,Upstairs,,Mid afternoon break,Upstairs,
1630-1645,$PT3A, Intensive Care,,Mid afternoon break,Upstairs,
1645-1705,Debrief and assessment, Breakout room 1,$TEACHER10A,$PT3B, Intensive Care,
1705-1720,$PT4A, Intensive Care,,Debrief and assessment,Breakout room 2,$TEACHER10B
1720-1740,Debrief and assessment,Breakout room 1,$TEACHER11A,$PT4B, Intensive Care,
1740-1800,Evaluation and Close,Breakout room 1,,Debrief and assessment,Breakout room 2,$TEACHER11B
1800-1820,,,,Evaluation and Close,Breakout room 2,
```

3.19 Subsidiary HTML files

3.19.1 HTML file: *badlogon.htm*

```
<head><title>Invalid log-on</title>
</head>
<body>
<h2>Invalid log-on</h2>
<p>Log-on name must be at least 4 characters,
 and cannot contain the characters ’ ‘ | or \
<p>Click <a href=’logout.php’>here</a> to continue</a>.
<!-- better is to forward to that page -->
</body>
```

3.19.2 HTML file: *badpassword.htm*

```
<head><title>Invalid password</title>
</head>
<body>
<h2>Invalid password</h2>
<p>Click <a href=’logout.php’>here</a> to continue</a>.
<!-- better is to forward to that page -->
</body>
```

3.19.3 HTML file: *failedaccess.htm*

```
<head><title>Access failed</title>
</head>
<body>
<h2>Failed access!</h2>
<p>Unfortunately you are not authorised to view this page!
<p>Sorry about that. Click
<a href="http://www.intensivist.com/strict/mainpage.php">here</a>
to return to the main page of AGATE.
<p><hr>
</body>
```

3.19.4 HTML file: *rescue.htm*

This page is used if all else fails (database connection failed):

```
<head><title>Something bad happened?</title></head>
<body>
Woops! Database connection failed.
Click here: <a href="http://www.intensivist.com/">
 Return to main page</a>
</body>
```

3.19.5 HTML file: *lostdata5.htm*

If odd or missing data are submitted to a PHP script which processes POST data, the following tiny page will be displayed:

```
<head><title>Lost data!</title></head>
<body>
<h2>Lost information?</h2>
<p>The POST data submitted to this form do not contain the expected
information! This suggests that you've randomly accessed this page,
or inserted really weird text in the previous one.
<p>Click
<a href='http://www.intensivist.com/strict/mainpage.php'>
here</a> to return to the main page.
<p><hr>
</body>
```

3.19.6 HTML file: *badcode.htm*

This file should rarely if ever be displayed. It indicates that bad data have been posted to a form, and in consequence display of the page failed due to the Check-Code function failing on a particular datum.

```
<h2>(Lost data)</h2>
Bad numeric data were sent to a PHP script. Checking of a code
failed.
<p>Click
<a href='http://www.intensivist.com/strict/mainpage.php'>
here</a> to return to the main page.
<p><hr>
</body>
```

3.20 CSS, CSV and TXT files

3.20.1 strictstyle.css

```
body {font-family : "Calisto MT", "Times New Roman", serif;
      font-size    : 11pt;
             color : #281D04;
     margin-left  : 2.5%;
      margin-right : 2.5%;
      background-color : #F0F8FF;
      }
.margins { margin-left : 5%;
           margin-right : 5%;
         }
.normaltext {font-family : "Calisto MT", "Times New Roman", serif;
```

```
             font-size    : 11pt;
             color : #281D04;
            }
.heavybox { border-width : thick;
            border-style : solid;
            border-color : #2F1D1D;
            margin-top : 1 ;
            margin-right : 1px ;
            margin-bottom : 1 ;
            margin-left : 1 ;
          }
.slim { border-width : thin;
        border-style : solid;
        border-color : #2F1D1D;
     }
hr  { color: #2F1D1D;
      background-color: #2F1D1D;
      height: 5px;
      width: 100%;
    }
    /* See: http://www.sovavsiti.cz/css/hr.html */
.emphatic { background-color : #FFFF00;
            color : #CC0000;
            margin-right : 1 ;
            margin-left : 1 ;
          }
.middling { vertical-align : middle;
           }
.stressed { background-color : #FFFFFF;
          }
ul,
ol {  margin-top: 0.6em;
   }
li { margin-top: 0.15em;
   }
a:link { text-decoration : none;
                  color : #0000CC;
       }
a:visited { text-decoration : none;
               color        : #000066;
          }
h1, h2, h3, h4, h5 { font-family : "Arial Rounded MT Bold", Helvetica;
                     color : #2F1D1D;
                   }
h2 { font-size : 18pt;
     margin-top: 0em;
     margin-bottom: 0.4em;
     margin-left: 0.5em;
```

```
     }
table { font-family : "Calisto MT", "Times New Roman", serif;
        font-size    : 9pt;
        BORDER-COLLAPSE: collapse;
      }
caption { BORDER-RIGHT: white 1px outset;
          BORDER-TOP: white 1px outset;
          BORDER-LEFT: white 1px outset;
          BORDER-BOTTOM: white 0px outset;
        }
th { BORDER-RIGHT: 1px outset;
     PADDING-RIGHT: 2px;
     BORDER-TOP: 1px outset;
     PADDING-LEFT: 2px;
     FONT-WEIGHT: bold;
     BACKGROUND: buttonface;
     PADDING-BOTTOM: 5px;
     FONT: message-box;
     OVERFLOW: hidden;
     BORDER-LEFT: 1px outset;
     CURSOR: default;
     PADDING-TOP: 2px;
     BORDER-BOTTOM: 1px outset;
     TEXT-ALIGN: left;
     vertical-align: top;
  }
td { PADDING-RIGHT: 2px;
     PADDING-LEFT: 2px;
     PADDING-BOTTOM: 5px;
     VERTICAL-ALIGN: top;
     PADDING-TOP: 2px
   }
```

3.20.2 CSV initialisation file: RORDER.csv

This file contains data used to populate the RORDER table. It is compatible with our CSS import, described previously.

```
rorder,'ratText',
1,Preparation of Intubation Equipment
2,Preparation of Equipment for a Failed Intubation
3,Preparation and Administration of Induction Drugs
4,Preparation and Application of Monitoring
5,Pre-oxygenation
6,Application and maintenance of Cricoid Pressure
7,"Intubation\, Checking\, and Securing Endotracheal Tube"
8,Timeliness of Intubation
9,Commencement of Ventilation
10,Initiation of Cardiovascular support
11,"Maintaining Anaesthesia\, Analgesia\, and Paralysis"
12,Please rate how you perceived your teams overall TECHNICAL performance
51,Control of Airway and Ventilation
52,Cardiac Compressions
53,Managing the Defibrillator
54,Safety of managing the defibrillator
55,Timeliness to first defibrillation/cardioversion
56,Preparation and administration of Resuscitation Medications
57,Preparation and administration of Adjuvant Medications
58,Preparing and Administering Fluids
59,Considering and Correcting Reversible Causes
60,Time Keeping and Record Keeping
61,Timeliness of Arrhythmia Resolution
62,Please rate how you perceived your teams overall TECHNICAL performance
101,A Leader was clearly established
102,The leader's plan for treatment was communicated to the team
103,Priorities and orders of actions were communicated to the team
104,The team leader showed an appropriate balance between authority and openness to suggestion
105,The team leader was able to stand back and maintain an overview of the situation
106,Plans were adapted when the situation changed
107,Each team member had a clear role
108,Instructions and verbal communication were explicit and directed
109,Team members repeated back or paraphrased instructions and clarifications
110,When directions were unclear team members asked for repetition and clarification
111,Team members shared situation assessment information
112,Team members asked each other for assistance prior to or during periods of task overload
113,Team members offered assistance when other team members became task overloaded
114,Team members verbalised important clinical interventions (e.g. I am giving adrenaline)
115,Task implementation was well co-ordinated
116,Team members refered to written aids appropriately
117,The team sourced external assistance when appropriate
118,Team members called attention to potentially hazardous actions or omissions
119,Individual team members reacted appropriately when other team members pointed out their potential
120,"When statements directed at avoiding or containing potential hazards\, did not elicit a response\
121,Disagreements or conflicts impaired team performance
122,The team became fixated on an isolated indicator or occurrence to the exclusion of other important
123,Team members made inappropriate assumptions about the capabilities or actions of other team member
124,Please rate how you perceived your teams overall behavioural performance
125,Please rate your teams OVERALL performance (Technical and Non Technical)
```

3.21 Image files

Here are the three images, which will be extracted to the local directory (not the images subdirectory) and must be manually moved to the *images* subdirectory.

Vertical bar: vbar.png

```
begin 644 vbar.png
MB5!.1PT*&@H‘‘‘‘-24A$4@‘‘‘‘P‘‘‘!‘"‘(‘‘‘"AC0&G‘‘‘‘%71%6’1#<F5A
M=&EO;B!4:6UE‘‘?7!QH6$Q3X]$2!‘‘‘‘!W1)344’UP<:%A,I=9WKX‘‘‘‘‘EP
M2%ES‘‘‘N(P‘‘+B,!>*4_=@‘‘‘"5)1$%4>-ICU)>592‘$F‘BJ&%4TJFA4T:BB
@446CBD85C2H:,8H‘)!H‘Z7M%$[P‘‘‘‘‘245.1*Y"8((‘
‘
end
```

The small logo: tinylogo.png

```
begin 644 tinylogo.png
MB5!.1PT*&@H‘‘‘‘-24A$4@‘‘‘$8‘‘‘!""‘,‘‘‘#=85"@‘‘‘‘%71%6’1#<F5A
M=&EO;B!4:6UE‘‘?8‘0$3&!F:=+3W‘‘‘‘!W1)344’V‘$""#0,&5\[O0‘‘‘‘EP
M2%ES‘‘‘7#‘‘‘%PP!B@*.9@‘‘‘P!03%1%‘‘‘‘‘@$‘‘@(!‘@("!‘,"!‘0$!@8&
M#0L&#@P%"0D)#‘P,#@X.%Q,*&10*’!<+$!‘0$A(2%!04%A86&!@8&AH:’!P<
M’AX>(Q\5)R‘/+"01+"44,2<3,BH7.2X6/C(8(2$A)"0D)B8F*2DI*RLK+"PL
M+BXN,3$Q-S<W.#@X.CHZ/#P\/CXˆ0#0913D?23L<3#T=4T,@4T0D5D8B4D<O
M6$<B64@C6DDD7$LF74PG7DTH8$\I85‘K8E$L9%(M950N85,T9U4P854[8E<]
M:%<R:5@R;%HU;5PW;ETX<%\Z<6‘Z<F$\=F4_0$!‘0D)"1$1$1D9&3DQ’24E)
M34U-4%!04E)25%145E966%937%A06%A86EI:7%Q<7EY>8%Y::V!)<F)‘=F5!
M>&=">6A">VI$?&M&?6Q&?6U*<VE3?G-;?W1;86%A965E:FIJ;&QL;FYN>W1F
M<’!P<G)R=’1T=W=W>GIZ?7U]@&])@7!+@G%,A7-.A71.@W-2AG51B7A2BGE4
MC’I5C7Q6B’M>CWU8D’Y9C7]AD8!:DH%<E()=E81>CX%DEX5@F(=AFHACFXED
MGXYHGI%UGY1\H(]IH9!JHI%LI9-NII5OII5PK9QVKYUXL:![LZ-__\YC@("‘
MA(2$BHJ*C(R,DY.3E965F9>3GYˆ?J)V%KJ*’JJ")KJ*(L:*‘MJ6‘LJ:-O*N&
MO:R’OZˆ,N;";O+*<H*"@JJJK:VML;&QN+BXNKJZOKZˆP;"+PK.0Q[:0R+B6
MRKN:SKV8Q+VMU<.>U<2?R\*NR,*WT\2DV,>BVLRMWLVHW,ˆSW="SX,ZIY-.N
MY]:QY-F_Z]JU[=RW[MVX\>"[]..ˆPL’‘RLK*U]#!V]3’W-C/T-#0TM+2U-34
MUM;6V=G9W=W=Z]_’X]W1Zˆ’,[>/,ˆ.?"ˆ>C#ˆNG$_.O&_NW(\NK6]ˆ_>Y.3D
M[.GBZ>GI[.SL]>_A]/’Kˆ/#Aˆ_7I_/;H_?CM\/#P]/3T]O;Vˆ_CRˆ?CU_?KQ
M_OOU__SWˆ?GY_/OY_OWY_/S\_OW\_O[\_O[ˆ(ZFDEP‘‘’)QT4DY3_________
M_____________________________________________________________
M_____________________________________________________________
M_____________________________________________________________
M____________________\‘OV=?&P‘‘!2])1$%4>-KMEWE45544QL\3GO6P‘IP:
M%3(@0<JF@T!A9=F@‘LGPZERIM)$>2J-D)5H8YY:9E98-9#:7C9;YFK/)YD&;
MS&C46Q:-VFBIVˆˆ\<ˆ&]ˆR9XK%K]XUZ+M2Z+<W_GV]\Y>ˆ\+:_E7@FW%_(ˆ8
M,L98N?V<R5B2MWN8),8R1’W@<3*(NXMN83(8<U<)_6X*8YY*Pˆ@.!@)RA&’X
M\.C%\V!AV-(2PD!‘J@!&R7$QEN8U]’-BF#H(&‘(!‘+7D,.8:*Q!&3:(8".@/
M2EF%,,[<3GEMB)SJV’)B8’+A+P14IV2(XYL/=GG@]1B6+6)R8F"04I82P%RE
M9YA-N\!KD<[<%4),3@33E[&4:B$F>AC;44IYVK%"ˆ$#N%U-.5(SR-U<8XUHG
```

```
M,=?>)TM3GB#JVK)@]!AA3.PZ!B+Z8./;:&WZ3GQ8DY2-QZV@]]R,]1)"=!F3
M%SA@HVX=T>W[<’X4Y%R^@:A*7\AQ7<6X5‘$9XC$BNI,C&J6<\SW1VK20\N@<
M‘W][PM_S-A)]6*@PAT#.4YN(5(%F1’<Y$J,6[XG%*XDV7\H#<:HT+_N<:‘.N
M,RL3T4HK$M-3%9/RE^A9)04_Q7!YX9]$BY%NNC>:G‘A,)1I4&:KG.Z+?QW->
MV’08.*.0UBO@"JVTIG,,T\6T&"_=’<AG>@’G!8W2O/8GHH]Q%3Q54>2$8P;!
MWTK<WU^)OAW.>4FS*<<’7);RZ<U$\[’+H"@NLT@QJ@(_@K^70,54:<Y\ZQ1P
M&J0YZVNBMG[(&35[=GQ,*OSU"G$SCO=])’,D/’G2>KT8+C=+>=-?1(]BG]Z1
M=]F)4?67)PR?\O=$SHMP0M=\:5ES(6<T#OU-’’JNKC=?/$RR]G<1_’T8KTZ‘
MF&6692T_‘OE=+.7<7X@^42Y[P^4X,-@I&>VN7OE[$.?#D<@-EHH’P!P)YHMP
M697_’B)L3#@PMK]+(.8JO’B^-,UW‘I@U4_#K%&E>\0W1.A2+JR),3B@&I^!!
MOYT/?S^P_7W<TK&L0!_^@_\0W8K=^H8=>@C&’@;&*J*_+PI<.3EKM8VQ9D/.
M25#W*=’&W,!"1VF%8.!O.C:Y‘RD]I%^22]LIUKOPJG"ZE-?_0>3’RE[.,1’$
MY.F+Y6LC^@$G4SS#E-=9P;C?=OEE[.*+:&!!C-U,VOV%H>8;(9@UHY#GA=*<
M_2-1*VZI<TQT8)2_:&WGH%M^5JQ;U4(K-%Y2+J.TGH#+M]@%’(FQ)W\KT2;E
M[S1I7KG<@;&F02/&Q,POB-:C@EVE(6.B’>-1I6*(!4CI>6Q[-,0\XZ18;Y?‘
MY1E2WH@&YF>!+P41AC$TW8=A\%O‘7RGGK0[#6/.XYK\*Q15.EVT,NF,FQ#P"
M,7=I[?*U<(KUU6C\:2I*ZV>,";=N_‘X,>G523E[>V"5^_PM%G!\.2O-S2R-B
MCFI@#0T-]Z[P^\N9WCD4PSHB>5\LQ;DV’\CCQ*[VZG+,H2‘F+8C9S6[@$^)1
M^’[;ZM4=I:4PU4’*-L/TG6\JBHOA^?9ZU&!=.V9P$)-O^SN2=Q([Z/5H8#4=
M27G:*=L?@)N*8FKHC,*’]M!OY‘23:JFQ*3V&8L$%\+>D4PS?6;_BKHOL?L<,
M&#!@_]K:VJOOBQWW[#5P8/ZA(T:<’KN)#@&]4AB3UE.<J-&M-GXOSK(_;&)’
M6Q\]6&-B^J.IX6-<S;IXL2"\2S@Q]6H@9F=G+UH5/U9FZOM[5G2,FR40J=X8
MO;@T$4K@/YI8O3B1<+K<;0P^<\2Y43!9"6)<948T-0G+Z1WK^G4_MF+^<\P6
4&HS)Z\L>5M(‘‘‘‘‘245.1*Y"8((‘
‘
end
```


The big logo: biglogo.png


```
begin 644 biglogo.png
MB5!.1PT*&@H‘‘‘‘-24A$4@‘‘‘9‘‘‘‘%\"‘,‘‘‘#6YQT’‘‘‘‘%71%6’1#<F5A
M=&EO;B!4:6UE‘‘?8‘0$3&!F:=+3W‘‘‘‘!W1)344’V‘$&%Q<(.(/[WP‘‘‘‘EP
M2%ES‘‘‘7#‘‘‘%PP!B@*.9@‘‘‘P!03%1%‘‘‘‘,#‘P64@C65E9>VE$:&AHBGE4
ME()=_\YCF)B8N:B#PK&+YM6O_NW(‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
```

```
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
M‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘S]/6$@‘‘‘ET4DY3________
M__\‘4T]X$@‘‘#\M)1$%4>-KMG>M"Z[H.A’,IM‘<V[_^Z!UBW4N3$8TG6V&A^
M+EBIK"\CV8X;EEN*2DMT‘*FO2B!D2B!D2B!D2B!D2B!D2B!D2B!D2B!D2B!D
M2B!D2B!D2B!D2B!D2B!D2B!D2B!D2B!D2B!D2B!D2B!D2B!D2B!D2B!D2B!D
M2B!D2B!D2B!D2B!D2B!DFAG((FI]BH[K..CH‘(J!R=G<=^45EAVX1,2XHP,H
M:"WQV*MO\"+2!-(25RF9U>G<BP:A)D(*Y""9M>E<]$Q#1AX=@*C]()F5Z5ST
MEP@1)Y#C;-:D\^G08\1$!@*R(ME<])<(&GIT‘&)0QW=W13H7_26BQAX=@!33
MR=U=D<[S2T0/LAAY=‘!23*<\SM(IKV(6Y!)A@X\.0‘BI‘LBNOT3T,$NA1P?P
M36L-C^-T5B$E)<(’I"Z;1^FL*5BT1.B‘/-7Q.,IF)=($4A=0W=U]D,Y%?XG(
M\4<’\!A/Y=U]D,[Z2T0/5HP^.H#’>*IY[,@5L$N$)B‘Z@(=P:JM-.9T)Q%!U
M4][#="(\&(EP‘8$,(J830TI(A‘H(FDWA:2YFD‘1R$@QF$"&=B_X2T3F(#N‘^
M%O#N%M*)7R)ZT-]&$!W‘/]6NT0_2V8"4C0@1$+C:?$]G"Q"R<W/L0%8$2‘L/
M-HOP‘&DRR)=TMO$@(T(#I#6;3_I+1‘_]:QZB‘_@;2%LR[](I\SCW&!<1%B"M
M=_==.MLO$3WX+XF(#N"7T#6Z4+042)F(D‘#19/-7.L^.*HY"A!E()8^]?‘7L
M$A3B‘**ZNS_3V=[1R8A0‘-’=W9_IU%XB.@7_<A$=P&<02AZ[’BD-$08@^FP^
M;PG$3D??/?LF.<?73<N#A@@!$,0@J_COUZMH$8@’"Y%X(#O‘XQV(D.;KNRY;
M[24^+I)‘CB)‘"I:8YP\@@D6*5EB9#P6%‘X$ZNICHZU6TR(%!P*.0??,1’@!6
ML(0?7G\3J>WH9?M$YX(!"%*P=O&GOWD\%JU#@Q1^S/‘T-Q@(M,N[BC_^‘^2K
M14YX\%HD&‘C:T;]G^R^/+Q8Y*5@+[V’?6""0041BUSO=]?53@]#V]5@@;0:Y
M3^8]D#N+%‘UR‘B2>2"@0F<>Y0?XE_’H5+5)A$%:+1‘)IZNA??^GZ0‘3@04HD
M$@A2L’81VP.//T5+‘R2:2"‘0J&"MTJ]=O^ERR..1:@)I!K**O_<=R*=%BCS6
MJL_ZJ4#:._J?V_LJZ+VO5Q:LXJ?%KM?)@%06K%^_*@%YMTAMP>*T2!@0K4&6
MY?DJ$ZDW".-$*PH(].T<>7Y\$8&\]_5J@RR$?3T*R%*?G@*0;9,M<MT*A0BX
M!7X>$,@@(KYM*Q$I6‘2Y!WX>$,@@XN]_‘"D4K6VIO0BA16*‘0)LF(L!M*UM$
M[.O8I_XT("B/[__C%Y!ZBY0_EJQHA0!9<"‘WB4>92-5%*"T2‘:3V%7Q?,B,#
MV2JGOH=W‘I5%(H"T&.3K>;J_/"JGOG\N@DSN?@X0N*-_P[AM9T0NFWP5X%[X
M.4"0@O4D_K][(#5]_>S#F2S2’TA3P?K0*O&H*5KW5(&3Q#\$"’2V^B%6$<CE
M=+TN7H2UKW<’@A0L^64_VX,N)Q;Y>@W@A@AY,-(;2’/!*@,I3’V?ER.JM!89
M"LA-Y’’2U^M"H.GKG8’H>’ST]4W085^7+D)<M!B‘5//X^/\2D".+Z(*8’8C2
M(.]+[6U#+%*XQ^NC^)%‘‘!ZWFPRD;)’Z,$CZ>E<@0"(*J;AN&!$D$(ZBU1.(
MNF"5#%($<H4B22‘PD,L&$H%"H;!(1R"-N[SWVLJ2^_H+‘H2BKW<$XLK#Q"(,
M1:L?$,<.XENT)@6BV.6MXE$"\HH‘(;!(-R!‘‘@HI>#T&4EH=%N)AW?7M!<2[
M8&W%!R.#6206B,Q#OB//>$PR]>T$I(-!X*+%N>L;"<2:QQ06Z0-$W]&K@/@1
MZ6>1+D!\U^CWDH&,M%[O‘D1OD&LE$*R+,!:M’D#Z=/0C(@-9)‘Z("X_Q^WH’
M(’J#O-0#\5NO3P[$IV‘9621RO>X/!"E8\NP2,<CP1<L=B%]’OUY+_XX08;/(
M‘$‘N12?,:!%O(’Z/03Z2+‘,9>M?7&X@KCQ*1D2WB#$1?L([K4N>B-3P0\‘T:
MD@X-XFB1J*+E"P08:N/940N+4#T8<06B+UAG!IEOZAL‘1.8!/;=]/@,R;E_W
M!.+W&.0^R:?([C3‘KJ\G$,0@$(_G<R##6L01B%L’>4AV9R+#‘NE3L-"B1;]>
M]P."%"SH,<BE#@AF$62+9TP@;@5+F-+.9)’.0&0>K5/>&?NZ%Q"]05Z!-",6
M(5^O=P5B4;#D+,]C$2<@^H*%&*1[T7*TB‘^0GAW]4\_(;U.OUWL"V0$@S]‘]
M/U%?=P’2W2">0#I/?3L"<>OH[WI[F\4B’D#T!L%N^$\@)E-?@M?(=@/B6K#>
MWCPMTK5H.0#IWT%>/X#\U[EH.4U][8$$=/2WM[)%3)[F=ISZ]@*R‘T‘NT+W^
MAT=WB_@0,0<29I"B1<;:]>T$Q(!’<<K[W]LQD;$L8@U$;Q!TC?[RE\<;MO_%
M2<08B._9ZL."-<G4UQ@(4K‘L=GGO"U9‘T7*8^MH"\>OH!YLFYT!,=GT[6:0’
M$,0@.AX3]’53(’X&J>H@’[H@0$L6B7R:VP%(OX(U@T4L@2‘&@0I6L:._?@<R
M^I:6(9!^1Q4/##+\>MT0"%*P;!Z#O+W5$\$L$O=@Q‘X(04=WW_6=‘$C?CCY!
M7S<#HC=([=GJ4R#/$-C">)"..‘Z0@((UND6L@+AUD)I=WD?UWO4E!$+3T0W7
```

```
MZ\"81@&R‘T‘:G]M.5[1L@)‘9!%P=&MQE8P"1>?CL\DYC$1,@>H.@:_27,R‘O
M$.#"N"+6ZQ9‘_%\Q‘QMDW*>Y%D‘0@Q0V1T$@K^=‘1MWU-0#BM\O;W$$.^GJ!
M,’2O>5K$‘(A;1V^<\H[=U_5‘]!T=+5BG’;UAZHM9Q’%UZ‘4DNF‘-:Q$U$+U!
MS’9Y’]7[Z^LF1+1‘2)[;0A:A7J]K@>@+%FJ0_^J!C%BTE$#Z=W3‘(*X’&;WZ
MN@Z(?+::HJ-[6\2K:.F‘Z‘WR"MW$,)#Q#C(Z‘‘E?HP_=151‘]‘8QW^5]U’"[
MOAH@Y!W]L*_3KM?M@2‘\;(XJ3E6T%$"&,,AP7U\W!R+SZ+G+^Z#!OK[>#D1O
M$+=-K*\RV=+J]AI98R!T!<NWBSBL#IN!Z‘V"GL2J>&[+4;14J\-6(-2[O)46
MH=SU;06"=’3Y"K9’%:.*EGD7:02B+UA.SVT1(&!?[V,14R‘&/,PVL4:U2!L0
MO4&<=WD?-<ZN;Q.0X%?,<%G$N&@U‘4$*EL<K9KB(V%JD!4C_CJXVR#B[O@U‘
MY*<#?)M88Q:M!B#]#=*\1K\7U-<-+-)8M,R‘&/!PFO+Z6\1RZHL#T1NDTR[O
MH\;8]86!#+6)56,1LM?2P$#T!0M];JN?\HY4M%‘@_3NZF4&*[]R0/S;H(*,-
M$)F’/,_HN<L[H$5‘(&X=O6@0NX+U!KZ6QF!+JX&(!1"JHXJC6P0#HC>(XD7B
M%GJ%;H<((A"0‘7=Y^UG$:.H+‘4$ZNGR%R(X^1M%"@.@+5KA!^‘\R‘D"@75Z(
M1T>#T!<M‘,@4!J’?]:T’0OF*F?DL4@\$,8C;B\0MQ’WZNAK(P+N\=18A>=U<
M-1!]1X_;Y:TCPF&16B#].[KQ&KW9(IVGOCH@!CPZ;6+ULHBZ:%4"T1L$W>5U
M!4*\ZUL’9**.SFZ1.B!N!2N@HWL345JD"DC_CNYLD/[K]1Y‘$![QN[RC6*0&
M2’^#.*W1FRUB,/6MW=)J!V+‘(V3*2VZ1"B!Z@Z!K]"Y‘3%Y+8W^0\1S(-+N\
M_2RBF?J>‘YFS8/D245CD%$C_CMX-"’3ZNM=!QD8@,@^>;^>,:Y$S(/T-XKC+
M6P<D]L%($Y"!CBH.9Y$3(/J"A:[1.QJ$\FGN,9#I=GG[6:2U:!T#03J(_*\7
M$$BQDCD)(7*Y/HDR71TN.‘]‘ZPX6K(U;X."M@>S0YP,\BAW]&4H/.Y$FBQP!
M41MD1S>Q+EAZR($T[?IZ‘IG/(!B1IBZR^/%84![\!NE‘9/’C‘7?T$7BX=Q$_
M("O:09Z’‘.)MD<6+Q[*C;]‘8@X=W7R\!6:’/%4,I#&C@CMY‘!+9("8C>(---
M>9N(P$7+"PC<T8<Q2‘P0+QYC3WE[$%E<>$PZY6T!@FYI^0!9=^P<VD@%"R4"
M6F3QX+’L_Y,’,H=!7"WB‘J18L"8QB.?4=W’@L>SH*V8&,PA(!"I:‘A#]FA‘M
M6,,9Q-$B‘A‘UCZFGO"U$D‘T4!X>L3_(0)BI8)D3V6B!JBQ1&,$U’[P[D"?FL
M:AZ3&40_]:V?92DM4AC‘9‘;16P0‘HK((R&-4@ZB)U,^RE!:1@Y^M8*%’%AT0
M!9%"\!,:1&>1&P:D_8R<’/J,!E$1N8%‘FBU2B’RZCHX#64*‘%‘*?U"#M4]\;
M#*212(’’I‘9IKUD-0)J(%,*>LJ-KB-QB@<Q:L%‘@_X@T‘6D@4@AZVH*%$EG/
M>1P!P7=]Y9!G-DB319I.+K98I!#R\\P\6J:^MU8@H$4*‘<]M$)#(>L;#\CN&
MA7A+/*;H("B010<$VO4MA%N<\D;G,8K(30$$4B’:$H^K_A-=57]SO>@_[$YF
M0‘Z*)G+H@D9[_6!,/]<9R+‘\Y‘’)8VG^8_;BQSJ&7QS!$$!N]?>7Z:?:7*8P
M0Q[8(/(]AORYE,9/]0O^@,<00&(L8@-DPH)U0RQB."!/((,;!.GKAI_I%?CX
M!I$7QMX6<00R=$<O#\QY4!9‘9C7(+:*O&P‘I\)C‘(!%]W0W(\!V].#AV(!,7
MK%N‘1;R‘%‘UBNO’30?4WF]’G>00\CT%NW8N6$Y‘I.GIY@(Z#TP*9NJ,7A^AH
M$260&7=YOZ6H>H‘6_5$)Y‘<8I’-?UP&9OJ,71^E6M#R‘3#/E/1JFUPA50’Z&
M0>1.Z641#9#"]]ZFZNB_LU1_VZD_RCC,Z3IZ<:A.%E$‘65‘@3^.J7U^W!_)C
MQ‘8D.A_Q<B&20!3R6*\W‘XE.!H,\+-(*1/D*ITE$!"0Z%1QRL$@CD.A,L,B>
M2‘+1B01(=!IX9&Z1!*+4;DRD"4AT$IAD;9$6(.I77T\E&4CSZK‘%2’0*N&1L
MD08@T1E@DRV1!*)7,)#HX?/)U"(PD-S$^B[+J2\,)’KPC+*T2‘*QD"&1!&*A
M2"!)1!+XQPH3B+OJ_QZ%.9‘D(FDUXI%‘K!0()(E(6FUX-‘%I_UL6,VLUX9’/
M0\P4"22)2%HM>#0"R4=4DNK^’H4+D+2(I-6‘1P*QU%\>BN.]>7+14#5_’\0+
M2!*1M*IY)!!;:0M6?C_$6*O6(‘H@N5Z7M(9]82<M(FJ-^TI;‘A$5^"W<E(<2
M")D2")D2")D2")D2")D2")D2")D2")D2")D2")D2")D2")D2")D2")D2")D2
M")D2")D2")D2")D2")D2")D2")D2")D2")D2")D2")D2")D2")D2")D2")D2
8")G^#_U;>4>HE[PH‘‘‘‘$E%3D2N0F""
‘
end
```

4 Backups and some technical details

4.1 The importance of backups

Backups are vital. It is the sole responsibility of the primary study author using this software to make regular backups. This backup option is available from the main web menu. The authors of this software will in no way be held liable or responsible for failure to back up all study data.

Backup files are downloaded as CSV scripts. Table entries generated automatically at the time of creation of the database aren't backed up, as re-creating the database will automatically generate these data. The distinction is fairly simple — all data rows with a primary key under 1000 are generated; all other data were entered online *apart* from data in the STUDYDAY and TEAM tables.[14]

4.2 Extracting PHP and SQL code

To modify our code (under the GPL), generate documentation, and create final PHP and HTML files, you need the following:

1. A simple text editor (NOT MS Word), for example, MS DOS edit or Vim. Better still is WinEdt, about the only software we would consider paying money for without being forced to do so!

2. Perl version 5.6; (You may have to tweak things slightly for version 5.8), and our Perl program *Dogwagger21.pl*;

3. Time and patience.

We've provided the source code for both program and documentation as a single convenient portmanteau — the file *AnOnlineDb*.tex* (where * is the version number, for example, *AnOnlineDb_0_50.tex*.

Standard LaTeX-handling utilities (such as *pdflatex*) can be used to convert the .TEX file into a PDF document.

To extract the various PHP files and so forth, you need our small Perl program, *DogWagger*, version 2.1. For use under Windows,[15] we have conveniently provided DOS batch files to allow easy calling of this program, which should be placed in the same directory as the .TEX file and the batch (.BAT) files.

[14] This is rather unfortunate. Because we clumsily hard code the study day number as equal to the primary key, and there is a fixed relationship between TEAM and STUDYDAY entries, we make this exception. Smarter would be to have added entries in the TEAM and STUDYDAY tables, loosening up this association and permitting keys over 1000, with associated SQL changes throughout the PHP scripts. Hmm.

[15] We used Windows 2000 professional.

4.2.1 DOS batch files

For the sake of convenience, here are the simple DOS batch files I used to manipulate the source code. Note the catch described under the header *make.bat* below!

cleanup.bat

Delete all unwanted files, prior to re-generating them.

```
del *.aux
del *.bak
del *.toc
del *.php
del *.htm
del *.sql
@echo off
cls
echo (Deleted .aux, .bak, .toc, PHP, HTM and SQL files)!
menu
```

menu.bat

A simple DOS menu which describes the batch file options:

```
rem Should clear screen in CALLING script
echo                                ...
echo ======================================================
echo :       MAIN MENU                                     :
echo ======================================================
echo :       Command        Meaning                        :
echo :       -------        -------                        :
echo :                                                     :
echo :       help           This menu                      :
echo :       cleanup        delete all generated files     :
echo :       make           Create all files               :
echo :       final          copy relevant files to FINAL dir :
echo :       update         cleanup+make+final             :
echo :                                                     :
echo ======================================================
```

Here's *help.bat* which simply displays the menu:

```
@echo off
cls
menu
```

make.bat

Invoke Dogwagger, version 2.1, to create all files! Here's the batch file:

```
@echo off
cls
perl dogwagger21.pl AnOnlineDb_0_63.tex
cls
echo (Made all files from AnOnlineDb_0_63.tex. See WAGLOG.LOG!)
menu
```

Make sure you have the correct version number in the line:

```
perl dogwagger21.pl AnOnlineDb_0_63.tex
```

There's a catch in the above. If you change to a new version, then you must manually edit the batch file, or first change the version number in the above make file, and then rerun make after changing the name of the actual .tex file! (Or simply edit the name presented in the DogWagger menu, of course).

final.bat

Create the final directory image for export to the web. Relies on the presence of all relevant files in the current directory, and its subdirectories *images*, and *csv*. The database_connect script should *not* be exported to the *strict* directory on the web, so that image is deleted from the *final* directory.

This script will try to create the following sub-directories within *final*:

- sql
- css
- images
- csv
- txt

It won't delete the contents of these sub-directories if they exist already.

```
@echo off
cls
rem move generated png files to images
copy *.png images
rem copy over php and html files
rem NB ALSO must fix passwords manually
```

```
rem NB also must move database_connect.php to correct location
md final
del /q final\*.*
copy *.php final
copy *.htm final
rem Delete database_connect script: NOT HERE!!
del final\STrICT_database_connect.php
md final\sql
del /q final\sql\*.*
copy *.sql final\sql
md final\css
del /q final\css\*.*
copy *.css final\css
md final\images
del /q final\images\*.*
copy images\*.jpg final\images
copy images\*.png final\images
md final\csv
del /q final\csv\*.*
copy csv\*.csv final\csv
md final\txt
copy txt\*.txt final\txt
echo off
cls
echo Files copied to \lara\agate\strict\final\
menu
```

update.bat

'Update' does the whole schmeer — cleans up, makes, and invokes *final.bat*.

```
echo off
cls
call cleanup.bat
call make.bat
call final.bat
cls
echo ---**UPDATED v 0.63**--
menu
```

4.3 Creating a mySQL database

Convenient tools such as CPanel easily allow one to administer a MySQL database. The one inconvenience of this tool is that it is remarkably easy (with a single click and no confirmation) to delete an entire database. We do the following:

1. Create a database called 'strict';

2. Create a user and a hard-to-guess password;
3. Make sure you add this user to the new ‘strict’ database!
4. Manually enter the user name and the password in the file *strict_database_connect.php* and upload this file as described in the next section.

Alternatively, phpMyAdmin is particularly useful for administering a database, including setting it up. Depending on your administrative rights, however, you may or may not be able to create a database on your server with this tool.

4.4 Uploading PHP scripts

Use any good FTP program or a CPanel utility like File Manager to move files onto your web server. Remember to:

1. Insert the relevant user name and password within the file *strict_database_connect.php* and store this file above the *public_html* directory;
2. Insert a non-trivial password within the *strict.sql* file, replacing the stub ‘PasswordGoesHere’ near the end of this file![16]

4.4.1 Directories required on web

The database should be installed in a directory accessible from the web called *strict*. Ideally this directory should be in the main html directory (usually *public_html*); if you don’t do this then some paths will have to be altered (e.g. in *database_create.php*). You will also need the following:

- an *sql* directory in which you should place the SQL setup file *strict.sql*;
- an *images* directory where all of the PNG and JPG files go;
- Put *strict_database_connect.php* in the directory which contains you web HTML directory i.e. *above* the web directory, so that it cannot be accessed directly from the web! This is a security measure;
- Insert strictstyle.css file in a css subdirectory;
- Template files for the roster go into the *txt* subdirectory.

[16] Or change this password immediately after you’ve created the database.

4.4.2 What next

You then need to:

1. Point your web browser to the file *database_create.php*, for example at the URL http://www.intensivist.com/strict/database_create.php. If all goes well you will see a listing of the SQL statements you submitted (debugging is on) and a statement that you have successfully created the database.

2. Test out the various web pages. A good place to start is of course the file *login.htm*, using the logon you created in your initial SQL setup.

3. Delete the *sql* directory and the *database_create.php* file.

5 Change log and future objectives

5.1 Change log

[From version 0.70 onwards, alterations to the STrICT database will be logged here.]

5.2 Future objectives

The following are desirable improvements to the database:

1. Build in randomisation of teams (team 1 or team 2 for each day) to either scenario-based or immersive teaching for *either scenario* (CVS or airway), and modify the roster printing accordingly (v 0.61):

   These changes were made to the TEAM table (added field ISBnature), to *strict_add_studyday.php* (with randomisation to one of the two options), and to the *subsession.php* script which now permits display of the ISB type for a given TEAM. Clearly now two extra timetable templates must be available — if the ISB type is 1, then we swop the ISB sessions. For consistency, we also renamed the template file *Timetable.txt* to *Timetable1.txt*.

   The above changes necessitated changes to *strict_show_roster.php* and *strict_csv_roster.php*. We now determine which of the *four* timetable files needs to be loaded.

2. Modify code to permit addition of assessment of several extra scenarios (v 0.62):

   This involved additional rows in the WHICHPATIENT table, and further alterations to *strict_add_studyday.php* to permit association of more PATIENT entries with the first three pilot 'study days'.

   In addition, we need to 'de-couple' the name of a study day and it's database table ID (primary key), and likewise for a TEAM. We add the columns dName and tName respectively. This change is to allow pilot studies to be included in our database, but still start with 'study day 1', and so forth. Small changes were required to the following PHP scripts:

   - ancillary;
   - strict_view_studyday;
   - strict_add_studyday;
   - strict_edit_session;

- strict_assess_super;
- strict_view;
- strict_post_session;
- subsession;
- strict_participant_super;
- strict_scenario_super;
- strict_view_assessor;
- strict_view_rating;
- strict_add_team.

We've also amended strict_show_roster.php to still display if more than 8 frozen patient entries are associated with a study day, inserted a 'Coordinator' role into the TEACHROLE table, (and now also a second immersion scenario debriefer!) and modified .TXT templates substantially.

3. We must add an assessor comment (v 0.63, raText field added to RATING); In version 0.63 (after some debugging) we also:

   - Permit viewing and deletion of established allocation of participants to teams (*strict_add_team.php*) — the function ShowExistingTeams;
   - Check that teams participants are all from the same hospital;
   - Allow clearing of an assessment form, and omission of entries in"n/a" fields;
   - Remove a debugging statement from *strict_aagh.php*;
   - In subsession.php, alter 'Go' to 'Update database'.
   - Updated references from 'anaesthetist.com' to 'intensivist.com'.
   - Amendments to *strict_add_studyday.php*. Fix Javascript error with teamdate, also create 5 initial days with seven patients per team.
   - clean up CSS a bit
   - introduce the *strict_GLOBALS.php* variable $CONNECTIONPATH within *database_create.php* and *ValidFx.php*.

   Thought: Should we check whether a team participant has performed *any* RATING, and if so, not allow alteration to any of the participants in a team?

4. It's desirable for the administrator to be able to alter the date of first qualification of a participant (in case of error) (v 0.64);

5. The HTML needs to be tidied up extensively to make it compliant with HTML 4.0 (v 0.65);

6. The CSS is shameful, and needs to be rewritten (v 0.66);

7. Certain PHP functions could be moved to a common library (v 0.67);

8. We might add in a warning if attempts are made to create a group from people in different ICUs (v 0.68) [done in v0.63]!

9. We need to write a Perl script which will take the backup CSV data, order the tables, and then insert the data back into a re-created 'clean' database (v 0.69);

10. It would be very desirable to use Javascript md5 encoding of all passwords prior to submission. Perhaps start with the 'deletion-page' password (v 0.70);

11. We need to consider the possibility of 'not applicable' ratings for some items, permitting these items to be left blank (ugh), and adding a 'Clear Form' button;[17] [done in v 0.63]

12. We still must examine the possibility of conflict between multiple concurrent users of the FetchKey function;

13. Check all 'stub' and 'fix me' annotations throughout the source (.TEX) document for other ideas.

[17] Having an extra 'n/a' button in each column is ugly and wasteful.

6 The GNU GENERAL PUBLIC LICENSE Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law:

that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term ”modification”.) Each licensee is addressed as ”you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2. You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.) These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it. Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

   In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.) The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable. If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by

court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

9. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

10. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and ”any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

11. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

    NO WARRANTY

12. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM ”AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND

FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

13. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

(END OF TERMS AND CONDITIONS)