* 5COM1053 Computer Science Development Exercise

****

* 2017 Women’s Euro Championship Access Network - WECAN
* GP3 Case Study Assignment Briefing: Implementation & Demonstration

This “GP3” assignment concerns the case study briefing documents, provided with GP1 & GP2, which outlined the background to the proposed prototype “Women’s Euro Championship Access Network Case Study” (WECAN).

Note: Part 2 of this document provides an update of some changes in the specification of the WECAN system. You MUST use this updated specification for this assignment. Where there are differences between GP1 GP2, this document has precedence over all previous ones

* 1.1 Aims

This assignment is designed to give you practice in software engineering techniques. You will use technologies and experience challenges often found in contemporary software development.

The **technologies** include the following design techniques: web-based architecture (including the Model-View-Controller [MVC] architectural pattern) and object-oriented design, and for implementation: a software development platform and framework, which include components written using PHP, SQL, HTML, CSS, and JavaScript.

The **challenges** include:

* teamwork, including leadership & negotiation (building on your experience in GP1& GP2)
* planning, time management, and fixed deadlines
* demonstrating software to a notional “business client”
* coping with a complex (possibly incomplete and inconsistent) brief
* reading technical documentation and obtaining technical support from a “software supplier”
* 1.2 Tasks [see also section 2.5]

Use an Agile system development methodology using 3 Sprint Cycles to:

* identify an overall design and produce a data model

For each sprint cycle:

* identify priorities & aims and produce sprint cycle plan
* refine prioritised requirements and design solutions,
* develop the GUI (graphical user interface)
* develop software using the supplied framework
* test the software
* evaluate the cycle & and plan the next one

By the end of the project you should

* demonstrate the software to the client
* evaluate your project and your project management.
* 1.3 Assessment [see also part 5]

This is a **team-work** exercise, worth 20% of the total for the module.

With team-work, all team members are expected to make equal contributions. These can take different forms: not just volume of work, but also specialist skills, knowledge, or leadership. You should plan for approximately equal contributions and we expect most teams will achieve this goal.

However, sometimes team members do not make equal contributions, and so there is an **individual assessment** component to mark which will reflect whether everyone has contributed, and to what extent. This will take the form of peer-group assessment, moderated by the module team (the demo examiners). For details, see Part 5.

* 1.4 : e-mail Briefing [Background]

From: **Ms P.Pincher, Business Development Director, WECAN Systems**Date: **27/01/2017**To: **your Team**   
Subject: **Phase 3: WECAN Prototype Project**

DoIT has won a contract to produce a prototype for the “Womens Euro Championship Access Network System” (WECAN) 2015. Thank you for the work that you have done producing Requirements *[GP1]* and Design Specifications *[GP2]*. You team must now implement part of the WECAN and to be responsible for demonstrating it to our client *[staff from the module team]*.

In our bid, we said that we would showcase the capability of our newly adopted standard development platform EasyPHP. You must use the CodeIgniter framework that has been implemented on the EasyPHP platform to produce your WECAN Prototype. I appreciate that these may be new to you but documentation has been provided and our CodeIgniter framework supplier, Hackitt Systems, has promised to provide technical support. The advantage of this approach is that you will only be required to edit existing code, not to create programs from scratch. The underlying technologies are industry standards: HTML, CSS, PHP, JavaScript and SQL.

The WECAN, which you are expected to produce for the client demonstration, is only part of the specification *[GP1]* and design *[GP2]* that you have already produced*.* You may find some of your earlier work useful, but should only use what is relevant to this task. In addition, the client has specifically asked for a data model, which takes into account the requirements and constraints imposed on a data model by the framework.

Our client has provided as documentation: **Client’s Sample Data & User Stories**, **Client Priorities and User Acceptance Test (UAT)** to be used in Agile planning*.* This includes sample data and user stories which the client will expect your WECAN to handle. At the demonstration, the client will use the **UAT** to test your WECAN and may add some unscripted tests. Before test your software thoroughly against the UAT.

The demonstration will consist of the following to convince the client of the quality of our development processes and management:

* live running of your software against the UAT script
* a short presentation by your team
* a Q&A session

I am very busy at present with other aspects of our bid, but I am prepared to respond to any further questions you might have **about the client’s brief**.

In addition, **technical support** for the EasyPHP platform and CodeIgniter framework is available, but from a different source: the supplier, Hackitt Systems.

* 1.5 Support for the Software Implementation

Support is available from the MakeIT Business Development Director to clarify the **client’s business requirements** and from Hackitt Systems support team for **technical information** as follows.

*To contact “Ms Pincher” to answer questions about* ***the client’s brief*** *you should use the* ***Class Discussion Board*** *on the Studynet site for the module.*

***You must post to a thread prefixed “GP3: ask Ms P - ...”*** *for example “GP3: ask Ms P - target screen size”. You can start a new thread if you like but be sure to check all existing threads first. Ms Making does not like answering questions she has already answered.*

*To contact the software supplier to obtain* ***technical support*** *for EasyPHP or CodeIgniter you should use the* ***Class Discussion Board*** *on the Studynet site for the module.*

***You must post to a thread prefixed “GP3: ask Hackitt - ...”.*** *You can start a new thread if you like but first be sure to check all existing threads and the documentation already posted on the website. Hackitt Systems are entitled to charge for information they have already provided.*

These are the ONLY ways to get support. We will not accept requests by email for new information nor will guidance be given in the labs or in person - since, to be fair to everybody, we wish to make such information visible to all. We may also post updated information from time to time.

2: System Design

* 1. System Outline

The ONLY **users** of this WECAN system are the staff at Competitor Id and Access Office (CIAO).

The client has agreed that your WECAN must implement the following functionality:

* manage the authorisations which allow competitors to enter tournament venues for matches
* respond to and log results of requests to enter venues by competitors.

At the demo, your WECAN must demonstrate that it can handle data about the following:

* national teams with their competitors and their id cards
* venues and matches
* authorisation of cards to enter venues for matches
* responses to requests by a competitor’s card to enter a venue
* the log of actual entries to venues by competitor’s cards

Before the tournament, all competitors identified by their NFAs must be registered, issued with cards and given authorisations to enter all venues where their teams will play their first round (group stage) matches. However, an individual competitor may need to leave, or may need to join during the tournament. The system should allow appropriate operations for both teams and for individual competitors.

A card can have the following state: “valid”, “expired” or “cancelled”. All newly created cards are set to “valid” from the start date to end date of the Women’s Euro tournament 2017. When a team is eliminated, the end date for all of their competitors’ cards is changed and their cards “expired”. The “cancelled” state is only used when a new card is issued to replace a card which is lost/ damaged/stolen. The new card has a different card number and must be given the authorisations of the card which it replaces. The old card must be “cancelled”. No competitor should have more than one valid card at a time.

Authorization for competitors to enter a venue assumes that a card-reader system to read the card idwill control entry to the venue. However, for the purposes of the prototype and demo, the card id will be input manually, rather than detected by card reader software. When a competitor wants to enter a venue for a match, the system should check whether their card is “valid” AND has authorisation to access the venue for that match. Although matches have a specified time, the authorisation should be valid for the whole of the match day (not limited to match times). The system should log whether an entry has been allowed or not, and add a new record in the entry log.

For information on functional and non-functional requirements of WECAN see detailed specification *[Section 2.4]*

* 2.2 Data Model and Description

The data model for your WECAN must be capable of storing all of the data required by the system.

Your data model should include:

* “Main tables” (i.e. those tables that map to classes of objects identified in this application): competitor, team, card, venue, match. Your WECAN should allow CRUD operations on these tables
* At least one “join table” to support the many-to-many relationship between cards and matches reflecting “authorisation” to enter a match venue for a competitor, and a table implementing an entry log.
* Other tables (expected by the Code Igniter framework) holding supporting information. For example, a “state” table, i.e. containing states of a card, or competitors and people titles, such as “Mr”, “Ms”, “Dr”, etc. These tables should appear in your data model, but WECAN need NOT have CRUD operations on them.

You will need to design and implement the data names, structures, and relationships. Table and field (column) names should be self-explanatory. Base your design on “**Client’s Sample Data & Scenarios”.** However, remember that it shows the client’s view of the data, and not the requirements/ constraints imposed by DB design and the Code Igniter framework. Note:

* A team will have many competitors , but a competitor will belong to only one team
* A competitor can have more than one card, even though only one card can be valid at a time.
* A venue can have many matches, but a match only happens in one venue.
* There is a many-many relationship between Card and Match (one match will have many cards authorised and one card will be authorised for many matches), you will need a "link" table – Authorisation. Authorisation is for one Card and one Match. Card can then be linked back to Competitor/Team, and Match can link back to Venue (using foreign keys).
* [CS ONLY] You must also provide a log of all attempted entries. If you want to create a table for this, you will have many to many relationship between cards and matches, so will need a “link table (say “Entry”.

However, you could simply write the entry data(who/where/when) to a text file (since at this stage there is no need for any searching or report facilities on this log)

* 2.3 Detailed System Requirements

2.3.1 Data Model

Design relationships between tables so that your software can implement facilities to:

1. perform all the CRUD operations on records of the main tables
2. add authorisations to access a venue for a match for individual competitors
3. register, issue a card and authorise an individual competitor who arrives during the tournament
4. de-register, expire card and cancel authorisations for a competitor leaving during the tournament
5. retrieve authorisations for access to a specified match/venue
6. respond to requests for all entries to a venue for a match
7. add authorisations to access a venue for a match for a team
8. change authorisations to access a venue for a match for a team, if the venue is changed
9. expire the cards of all competitors of a team after it is eliminated.
10. replace lost/stolen/destroyed cards and ensure appropriate authorisations for new card

Data provided by **Client’s Sample Data & Scenarios** should be loaded into the DB before the demo.

2.3.2 Functional Requirements

The system should support basic CRUD operations:

* input, change, retrieve teams, competitors, cards, authorisations, matches, venues and entries

You should then provide:

* actions for individual competitors:
* register/ de-register individual competitor (issue card/appropriate authorisations, or cancel card)
* replace lost/stolen/damaged cards (cancel old ones)
* automation of some common actions, such as:
* expire all cards for all competitors of a team (after they have been eliminated).
* amend authorisations for all competitors of a team(if an match/venue is changed during the tournament)
* cancel all cards at the end of the tournament
* some exception handling, such as:
* not allowing the registration of an competitor who is already registered
* not allowing the addition of an authorisation which has already been entered, or for a match for which the competitor is not authorised
* ensuring competitors are not given authorisation to access venues not holding matches for their team
* other features you consider desirable, but not explicitly stated in this specification(see Part 5 of UAT)

2.3.3 Non-Functional (Usability) Requirements

* The user interface should satisfy established HCI usability good practice.
* The system design should consider sensible "workflow" and provide functionality which distinguishes between actions for both groups of competitors, and single competitor (see **User Stories**).
* Navigation for input or amendment should be coherent and contain a linear sequence of actions: an efficient and effective workflow for the target users. For example:
* a form to enter an competitor’s details should confirm the data has been stored and issue a card
* once a team and its matches exists on the system, it should be possible to authorise access to a venues for matches by appropriate navigation.
* System functionality should be as flexible as possible. Searching, selecting and amending individual records should be possible in different ways.
* an example of flexibility in retrieval: searching for an individual competitor's authorisations could be done on either competitor name or card number.
* user tasks could be more efficient if actions can be combined
* The user interface should be both easy and efficient to use (by its target users). For example:
* the interface provides alternative views of the database, from displaying an existing single record, to displaying all appropriate records already in the database.
* such views of the data should be easy to display and update with minimum effort by the end-user, for example in a “one click”.
* users should not have to retrieve data by visually scanning all records
* the user should be supported by error prevention/handling and meaningful/helpful comments.
* Security may be considered by providing a login procedure, but this is NOT a basic requirement
* 2.4 Agile approach to system development

We will expect you to use an Agile approach to system development using Sprint Cycles

* Look at the Module Guide to identify the dates/deadlines of the 3 Sprint Cycles
* Based on **User Stories &** **Client Priorities** and your analysis of how to develop the system, at the start of each Sprint Cycle, you will be expected to produce:
* A sprint cycle plan, using the provided documentation
* the plan should include a list of objectives for this sprint cycle, who will undertake them and an estimate of how long they might take
* this must be shown to the Sprint Master and his/her signature obtained

(sprint plans without the signature will not be awarded marks)

* your aim must be to produce a working testable system at the end of each sprint cycle
* At the end of each Sprint Cycle, the plan should be completed to show:
* which tasks have been completed
* bugs that still exist in the system
* the uncompleted tasks and bug-fixes should become the priority for the next sprint cycle
* Unfortunately, the client will not be available to test the system until the system is submitted, so you should designate a member/s of your team to act as "testers" (these should not be the "developers"). (Note these also do not have to be the same people for each cycle – roles can be rotated)
* 2.5 The EasyPHP Platform and CodeIgniter framework  
  The EasyPHP platform may be downloaded from the following location: <https://drive.google.com/file/d/0B0Ou1lzzphIJaG9xajlZSkJzbXM/view?pref=2&pli=1>.

The **CodeIgniter** framework is already included within the EasyPHP platform above. Further information and documentation can be obtained from the following location: <https://www.codeigniter.com/>.

CodeIgniter supports the addition of template libraries (like add-ons). Some of these are produced by CodeIgniter itself and can be found at the link above. We have also included a template library called **groceryCRUD**, which is already included within the sample application. Further information and documentation can be obtained from the following location: <http://www.grocerycrud.com/>   
  
See also the supporting teaching materials posted on Studynet as lectures, tutorials, or lab exercises.

A MySQL database is provided within the EasyPHP platform. You can view the database via the included application phpMyAdmin. Once EasyPHP is running, use the following URL to call the application:   
<localhost:8080/phpmyadmin>

The “Active Record” database pattern means that you should also adhere to the following constraints:

* Each table and corresponding columns from the MySQL database must have the **same** case-sensitive name used by Active Record, i.e. tables declared in the code (using CodeIgniter and groceryCRUD syntax) must competitor what has been set up in phpMyAdmin for the Object-Relational Mapping (ORM) to occur.
* Tables and columns should not be named using reserved keywords. Some of these may be obvious, but some are less so e.g. “union” which is not allowed since it is a relational operator. For a complete list of reserved (disallowed) words, see here: <https://dev.mysql.com/doc/refman/5.6/en/keywords.html>
* 2.6 Getting started

We assume you have attended lectures/practicals and can use a test-driven approach, understand architecture, constraints and use of EasyPHP platform /CodeIgniter framework and template libraries such as groceryCRUD.

* read the **User Stories** and **Client Priorities** provided and identify the stories/priorities for the current sprint cycle – identify the priority tasks for the current sprint cycle
* At each sprint cycle you should add functionality to your system
* However, you should have an overall idea of how you are going to develop the system and, in particular, use wireframes to provide an overall design for your user interface. During your development you should:
* use a test driven approach to develop your system
* translate functional requirements and those in the UAT into tests
* save tests in a manageable file and use it to track how tests evolve, which tables they affect, what actions are required in order to achieve successful outcomes. Add tests as you progress.
* Produce a draft of the data model - *best done by discussion in the team*
* read Section 2.2 to identify and name entities required
* check data in **Client's Sample data** to identify some required fields. **NOTE:** this does NOT meet all functional requirements, nor have all of the tables/fields required by the framework
* check functional requirements in 2.2 and 2.4.1 to identify any further tables
* check 2.3 for the constraints & requirements of a database in the framework
* produce a data dictionary showing all fields in all tables (on a spreadsheet)
* enter the sample data from client, create and add any other data which is required
* Create a MySQL database. Use your data model to create the DB for WECAN and load in the sample data
* Develop the application and user interface: CodeIgniter and the template libraries included such as groceryCRUD provides a specimen application and a basic GUI, both of which you need to modify, improve and customise to implement WECAN.
* You should concentrate on ensuring that tasks for single competitors can be done through your user interface, before considering automation (batch processing) for teams. But there are marks for automation (see the UAT), if you can get it to work properly.
* Ensure that you allocate plenty of time to the development of the customised GUI. Continuous adjustments to CSS code, JavaScript and HTML editing can be very time-consuming
* Use the UAT to anticipate and design solutions to issues which may arise during the demo
* Consider “advanced” features , if you have time.
* To prepare for the presentation, record evidence and maintain a record during the development of how your team has organised itself and managed each stage.

NOTE: Section 5 of the UAT lists some advanced features (which includes login). You should NOT attempt these until you have a basic working system. You do not get a lot of marks for these advanced features, but they are there for ambitious students who want to extend their knowledge. You should certainly not be spending time on these, if you are having problems with your basic system (just look at the allocation of marks !!!)

Part 3: Deliverables

* 3.1 Deliverables

**3.1.1 For your coursework submission** - **All documentation to be submitted in a plastic wallet**

* your data model documentation (printed copy)
* your software (soft copy on USB– identified with your team code)
* your 3 signed Sprint cycle plans
* your test plans
* a completed “GP3 Roles and Contributions Form”, (individual forms if you disagree as a team; see later);

3.1.2 For your demo (at a later date)

You should bring to the demo (the process and schedule *to be announced later*)

* your presentation – a soft copy to use at the demo, and a print out to be handed in at the demo

At the demo, your software must be identical to that submitted on Studynet.

* 3.2 Data Model Documentation

The documentation for your database design should consist of:

* a full E-R diagram
* the data dictionary

The diagram should be drawn using suitable specialist software or a standard drawing tool. Hand-drawn diagrams will be accepted providing they are legible.

* 3.3 The Software

Your WECAN should:

* implement the functional and non-functional requirements given in *[Section 2.4];*
* consist of a set of fully working .php files plus any .css and .js files
* connect to a database (in our EasyPHP platform: this has to be an instance of MySQL) that can be accessed independently by a database management system (DBMS) (in the EasyPHP platform: phpMyAdmin). This will permit the examiners at the demo to verify that operations performed using your WECAN have actually made appropriate changes to the database;
* store all the sample data provided by the client.

While you are testing your system for validation/verification purposes prior to the demo, you may add additional data. However, the version of your application submitted for the demo MUST contain all of the sample data provided. **This test data set will be used by the examiners to assess your software against the “User Acceptance Test” checklist shown in Part 4 of this document to establish specific input/amendment/retrieval facilities.**

* 3.4 Sprint Cycle Plans

See section 2.5 and the Sprint Cycle Plan documentation provided

* 3.5 Test plans

The testers in your team should devise some test plans which test the expected aims of the system.  
Test plans should include: aim of the test, test data, expected outcome, actual outcome.

* 3.6 The Demonstration

The assessment of your WECAN will take place at your team’s demo; marks will be awarded according to how well your system satisfies the functional and usability (HCI)-related requirements given above.

It is your team’s responsibility to ensure that your WECAN (including the application and the database) (i.e. phpMyAdmin, which is provided within the EasyPHP platform) will be ready for use and working during the demo.

***Your WECAN will need to run on a PC in a CS lab, so you should test your software on this platform in advance. You should use Firefox as your web browser as it is standards-compliant.***

Personal laptops can **NOT** be used for the demonstration, **unless explicitly authorised** by the module leader, **in advance**. You are reminded to check the module pages on StudyNet regularly for further details about the demo timetable and other arrangements.

* 3.7 The Presentation

As part of the demo, you should deliver a short (about 5 minute) presentation with 6 slides maximum to explain the processes you used to ensure the quality of your software and the management of your project. You should cover:

1. your team working – roles & responsibilities
2. an evaluation of your system: what it does well, what still has bugs, what was not implemented
3. overall analysis and reflection: lessons learnt; what we did right and what wrong

Produce the slides in advance (in a format such as Powerpoint) and submit a printout (6 slides to a page)

Note that the presentation should relate to your actual experience; this is **not** a role play.

Part 4: Submission Requirements

1. Submit to **CS Reception** by Mon 20/03/17 by 11.00 , in a **clear plastic wallet**:

* **on a USB** (renamed with your team code) all the software and other files for your system, (including your instance of EasyPHP containing your WECAN application) – so the demo can run just from your USB
* **on A4 paper**:
  + Assignment Front sheet **signed by all contributing team members** and including **your team id**
  + your database design (E-R diagram)
  + your 3 Sprint Cycle Plans
  + your test plans
  + completed GP3 Roles & Contibutions Form

1. Submit to **Studynet**, by Mon 20/03/17 by 11.00

* the above documentation and files – for a backup only. No need to include the EasyPHP application, but include a .sql file (export in PHPMyAdmin) and the whole application folder for WECAN

Only one copy of each is required per team. Diagrams/Sprint Plans should be scanned.

1. Your team must attend their **demonstration** as shown by a timetable to be announced later

* The USB submitted above should be used at your team’s demonstration.
* Your USB should contain all your software, both source and executable, with any other files needed for the software to run, including the database pre-loaded with test data. Include also, a copy of the complete EasyPHP platform.
* Submit a printed copy of your presentation slides

***You should test that your software works on a PC in one of the CS labs with Windows 7 and the Firefox web browser (not just on your own PC) before submitting your USB.***

Part 5: Assessment

* 5.1. Marking Scheme – Assignment Mark

This assessment is worth 20% of the mark for the module, with each team member awarded the same basic score, moderated by the member's contribution. During the demonstration, your assignment will be initially assessed against the UAT and given an assignment mark (out of 150). This will be converted to a percentage and scaled to 20% of the module marks.

5.1.1 Marking Scheme – Assignment

**“GP3 User Acceptance Test (UAT)”** has been provided by the client *[examiners from the module team]* and gives a detailed breakdown of how your WECAN demonstration will be assessed *[marked]*.

* The UAT shows the **type** of tasks that you will be asked to perform during the demonstration of your WECAN system.
* At the demonstration, the client will tell you what actual data to use.
* It is essential that your database contains **ALL** of the data which the client has provided in “**Clients Sample Data & User Stories”.**
* **NOTE**: this is a **first version** of the UAT. The client may find it necessary to make amendments to the content/order of the tests, in which case you will be alerted (by Studynet) and a UAT with a new version number will be posted to Studynet. Always consult the latest version.
* The **UAT** provides you with an assignment feedback form.

5.1.2 Questions at the Demo

The examiners will ask each individual team member one or more questions during or after the demo – and may ask specific individuals to demonstrate part of the software. Failure to respond suitably may affect your individual score since it indicates that you did not fully engage with the assignment.

5.1.3. Marking Scheme – Team Contribution

As a team you should consider and submit the “**GP3 Roles and Contributions Form”** posted with the assignment. This asks you to describe the role of each individual and provide a percentage “contribution” that each has made to GP3 (including preparation for the demo). For example:

*Example (a):* if there are five team members and you consider that everybody has contributed equally, then everyone should be allocated 20%. The total must of course be 100%.

*Example (b):* if there are four members and you all agree that one person has contributed twice as much as everybody else, then he/she gets 40% and the other three get 20% each.

If you all agree then you should complete a single team form and bring it to the demo. If you cannot agree, then each team member who disagrees should complete the form individually instead. The module assessors [your lecturers] will resolve conflicting assessments by asking questions during the demo, consulting Sprint Cycle documentation and interviewing team members.

5.1.4. Marking – Individual

Individual student's GP3 mark will be calculated on the assignment mark. No individual team member can score higher than the mark awarded for the assignment. Where a team member's contribution is significantly lower than that of other team members, they will be awarded only a proportion of the assignment mark (as for GP1 & GP2)

|  |  |
| --- | --- |
| **Contribution difference from the norm** | **% of assignment mark** |
| < 5% difference | 80% |
| >= 5% AND <10% difference | 60% |
| >=10% AND <15% difference | 40% |
| >=15% difference | 5% - 0% |

**Grading Criteria**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0–19 | 20–29 | 30–39 | 40–49 | 50–59 | 60–69 | 70–79 | 80–100 |
| no... merit | clear fail | marginal fail | satisfactory | good | very good | excellent | outstanding |

Part 6: Assignment Checklist

|  |  |  |
| --- | --- | --- |
| **Task** | **Item** | **Done** |
| **1** | Read all the documentation, especially **User Stories** & **Client Priorities** |  |
| **2** | Identify an overall **project plan**, and produce a **Data Model** (E-R diagram) |  |
| **3** | Decide on priorities and produce Sprint Cycle Plan 1  Get it signed by the Sprint Master  **Carry out the planned the tasks**, including:   * develop your software * produce test plan/s * test your software against the test plan using the sample data and your own test data based on the client sample data * review, make any corrections, re-test, and repeat.   At the end of Sprint Cycle 1 evaluate what has been achieved and identify what still needs to be done. Use this to decide on aims for Sprint Cycle 2: |  |
| **4** | Decide on priorities and produce Sprint Cycle Plan 2  Get it signed by the Sprint Master  **Carry out the planned the tasks**, including:   * develop your software * produce a test plan * test your software against the test plan using the sample data and your own test data based on the UAT * review, make any corrections, re-test, and repeat.   At the end of Sprint Cycle 2 evaluate what has been achieved and identify what still needs to be done. Use this to decide on aims for Sprint Cycle 3: |  |
| **5** | Decide on priorities and produce Sprint Cycle Plan 3  Get it signed by the Sprint Master  **Carry out the planned the tasks**, including:   * develop your software * produce a test plan * test your software against the test plan using the sample data and your own test data based on the UAT * review, make any corrections, re-test, and repeat.   At the end of Sprint Cycle 3 evaluate what has been achieved and identify what still needs to be done. By this stage you should have completed D4 |  |
| **6** | Produce **the deliverables for the coursework hand-in:**   * your database design documentation (printed copy); * your software, including the specimen test data (on a USB); * your 3 sprint cycles plans, signed & evaluated * your presentation (printed copy at the demo) |  |
| **7** | **Sign** the Assignment Front Sheet. All individual team members who contributed must do so; no signature means no mark. |  |
| **8** | Complete the GP3 Roles and Contributions Form(s). |  |
| **9** | Package your solution with the completed Assignment Front Sheet and Roles & Contributions Form in a transparent plastic folder with your team id **clearly visible** and the USB securely stored inside. |  |
| **10** | **Submit to CS Reception**: Your coursework by 20/0417 11.00 |  |
| **11** | **Submit to Studynet:** ONE zipped filecontaining **ALL** of your key files for the demo, including your software, database, and demo slides (but not the EasyPHP platform). This should be by the same deadline as the hard copy/USB hand-in – only **ONE** submission per team is needed   * it will only be used for **backup or quality assurance** purposes; * responsibility for this submission is with the team as a whole |  |
| **12** | **Prepare for the demo:**   * establish the time and the place for your demo and ensure all team members attend; non-attendance means no marks for the individual; * test that your software works on the lab PCs with Windows 7 and Firefox; * allocate roles and practise both the software demo and the presentation. |  |
| **13** | Produce **the deliverables for the demo:**   * your software, ready to run (e.g. on USB flash memory); * your presentation slides, ready to display; |  |
| **14** | **Attend the demo**. Then it’s all over! Wait for the written feedback, due after all the demos have finished. Use the experience you have gained to help pass the exam, do well in your final year project, and get a job in software development! |  |