**5COM1053\_61 Computer Science/Information Technology Development Exercise**

**2019 Mega Independent Cinema Experience - MICE Film Festival**

**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 “MICE Booking System” (MICE).

Note: Part 2 of this document provides an update of some changes in the specification of the MICE 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.

**Contents of this document:**

[1: Introduction 2](#_Toc906514)

[1.1 Aims 2](#_Toc906515)

[1.2 Background Briefing 2](#_Toc906516)

[1.3 Tasks Overview 2](#_Toc906517)

[Part 2: System Design 3](#_Toc906518)

[2.1 System Outline 3](#_Toc906519)

[2.2 Data Model 3](#_Toc906520)

[2.3 System Requirements 5](#_Toc906522)

[2.4 Teamwork & Agile system development 6](#_Toc906523)

[2.5 Using EasyPHP Platform and CodeIgniter framework 6](#_Toc906524)

[2.6 Getting started 7](#_Toc906525)

[Part 3: Deliverables 8](#_Toc906526)

[3.1 Deliverables Overview 8](#_Toc906527)

[3.2 Data Model Documentation 9](#_Toc906528)

[3.3 The Software 9](#_Toc906529)

[3.4 Sprint Cycle Plans 9](#_Toc906530)

[3.5 Test plans 9](#_Toc906531)

[3.6 The Demonstration 9](#_Toc906532)

[3.7 The Presentation 9](#_Toc906533)

[Part 4: Submission Requirements 11](#_Toc906534)

[Part 5: Assessment 11](#_Toc906535)

[5.1 Marking Scheme 11](#_Toc906536)

[Part 6: Assignment Checklist 12](#_Toc906537)

1: Introduction

1.1 Aims

This assignment is designed to give you practice in software engineering implementation 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 Background Briefing

MICE want you showcase the capability of the EasyPHP development platform and to use it to implement a prototype booking system for their "Old 1940s Film Festival" (MICE-OFF).

The advantage of this is that you will not have to implement a complete booking system and will have an opportunity to identify and fix a smaller range of problems. Using a platforms means that you will only need to edit existing code, not to create programs from scratch, and so will speed up development. The platform's underlying technologies are all industry standards: HTML, CSS, PHP, JavaScript and SQL, and so are scalable

The MICE-OFF booking system requires only some of the specification [GP1] and design [GP2] elements already produced for CS/ITDE and you can take advantage of work done for the DB module. Much of this earlier work will be useful, but you should only use what is relevant to the GP3 specification. In addition, your data model must take into account the requirements and constraints imposed on a data model by the CodeIgniter framework.

MICE has provided documentation:

* Client’s Sample Data & User Stories,
* Client Priorities
* User Acceptance Test (UAT)

to be used in Agile planning and which MICE will expect your system to handle.

At the demonstration, we will use the **UAT** to test your MICE-OFF system, but may add some unscripted tests. Before the demonstration, you should test your software thoroughly against the UAT.

1.3 Tasks Overview

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

* identify an overall design (including data model)
* implement the required MICE system

For each sprint cycle:

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

By the end of the project :

* demonstrate the software to the client *[module staff]*
* evaluate your project and your project management.

Part 2: System Design

2.1 System Outline

MICE intend to run "Old 1940s Film Festival" for their members, between 20th - 26th June 2019

* bookings will only be available to Members (with their guests) and can only be made online
* your MICE-OFF prototype will not include payment facilities
* the ONLY **users** of this GP3 MICE system will be MICE members, and the Festival Director, Selly Lloyd

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

* maintain festival information
* manage bookings for festival performances
* respond to and log results of attempts by people to enter cinema screens for festival performances.

At the demo, your MICE must show that it can handle information about the following:

* members
* cinemas and screens
* festival films
* festival performances
* bookings for festival performances
* responses to requests by members to enter cinema screens for festival performances
* a log of actual entries to screens by a member & their guests [optional feature]

The Festival Director, Selly Lloyd has provided information about cinemas, screens and films to be used in the festival. During the festival, she may ask for performances to be cancelled, changed or new ones added.

2.2 Data Model

The data model for your MICE-OFF must store 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): supplier, vehicle, driver/id card, venue. Your MICE should allow CRUD operations on these tables
* at least one “join table” to show and support the relationships between vehicle, driver/id cards and venues needed to register an “authorisation” to make a delivery to a venue
* Other tables (expected by the CodeIgniter framework) holding supporting information. For example, a “state” table, i.e. containing membership states of a card, or people titles, such as “Mr”, “Ms”, “Dr”, etc. These tables should appear in your data model, but MICE-OFF need NOT have CRUD operations on them.
* any additional tables required for normalising the database
* Note: the entry log may be implemented in a number of different ways, not necessarily as a table

You will need to design and implement field names, structures, and relationships. You are advised to look at Client Sample Data, but should remember that this data has NOT been normalised (as required by Relational DB design) and may not be in a format required by Easyphp. Data model for this application may differ slightly from the data model for your DB assignment (differences shown in red)

* A **member** has:
* a unique membership number
* a name, title date joined and membership status
* membership status will be one of the following
  + active - member has paid their membership fees
  + lapsed - member has not yet renewed their membership
  + cancelled - member has cancelled their membership

ONLY active members can make bookings for festival performances.

* A **cinema** has :
* a unique cinema number
* a name, location, address and manager (only name required).
* may have several screens
* A **film** has:
* a unique film number
* title, director & release year
* A **screen**
* is located in only one cinema and has a screen number
* has a specified number of seats.
* has one price for all seats. There are NO seat numbers and customers can sit anywhere.
* has only ONE performance per day
* A **performance** has:
* a unique performance id
* ONE Film, showing on ONE screen on ONE date at ONE start time
* only ONE performance on a screen on one date
* number of seats remaining (available for booking)
* A **booking** has
* a unique booking number
* must include a membership number, performance details and number of seats required.
* once booked, the system should display booking details including total cost

The system should maintain information about all bookings.

* An optional advanced feature of your system should check that there are seats available before a booking is accepted. If the booking is acceptable, it should update the number of seats available, otherwise a suitable message should be displayed.
* An Entry - this is not necessarily a record in a table (some of these are optional advanced features)
* The screen entry system must determine whether a member (with guests) is allowed to enter the screen on that date. When implemented it will have facilities to enter required data. However, for the demonstration, your system should have keyboard input for the booking number, screen & date. It should respond with a message saying whether entry is allowed, or not. If booking number is not known then membership number can be used
* Attempts to enter a screen must be logged. A log entry may just be a line of text in a text file recording the member, screen and date, and whether entry was allowed or not, or may be stored as a DB record with fields to allow selected access to data. Entries in the log are created after a booking/member attempts to enter a venue (and NOT pre-loaded into the DB)

Notes:

* different screens have differing numbers of seats,
* different screens at one cinema may have different prices for their seats.
* start times of performances on different screens may differ
* start times on one screen may be different on different dates
* a film may be shown:
* on different screens on the different dates
* on different screens on the same date
* on the same screen on different dates
* but NEVER on the same screen on the same date

2.3 System Requirements

2.3.1 Functional Requirements

The system should

* support basic CRUD operations:
* create/add cinemas, screens, members, films, performances, bookings
* retrieval of information about cinemas, screens, members, films, performances, bookings
* update for cinema(change manager name), member(change membership status), screen (change seat price)
* delete booking.
* amending a booking should be done by cancelling the old booking and creating a new one
* retrieve access the log showing entry attempts to screens
* provide facilities to:
* simulate an attempt to enter a screen for a performance : allow input of booking details and respond with a message saying whether the member(and guests) are allowed to enter.
* display and log the result of an entry attempt

Your system may have additional optional functionality (see Part 4 of UAT):

* automation of some common tasks, such as: display of selected information using selection criteria
* check that there are enough seats available for a booking
* update the number of seats available after making a booking, or after a booking cancellation
* log entry attempts and allow flexible access to data
* handle the deletion of a performance
* any additional features you consider desirable, but not explicitly stated in this specification

2.3.2 Non-Functional (Usability) Requirements

* The user interface should satisfy established HCI usability good practice.
* The system design should consider sensible "workflow" with an efficient sequence of actions, for example:

a form to enter a member details should confirm the data has been stored and issue a membership numbers

* System functionality should be as flexible as possible. Searching, selecting and amending individual records should be possible in different ways.
* 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, or by looking for arbitrary keys
* *utilise CodeIgniter and GroceryCRUD functions for ensuring data is presented in a user-friendly way*
* the user should be supported by error prevention/handling and meaningful/helpful comments. Error prevention could include asking users to select from menus, rather than entering data by typing into text fields.
* Optional Advanced Requirement (see Part 4 of UAT)

Security may be done by providing a login procedure. Login may differentiate between different types of users. Only the festival director should be able to maintain festival information such as cinemas, screens, members, films, performances, In addition, she should be able to do all member operations. Members should only be able to maintain their own bookings and attempt entry.

2.4 Teamwork & Agile system development

**2.4.1 Teamwork -** This is a **team-work** assignment   
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.

As part of GP2, you were required to use Azure devOps for project planning, communication and version control. There are no marks for doing this in GP3, BUT you should see this as a sensible way to manage a project and maintain communications with your team members, especially if you cannot meet in person. There is no need to use Azure devOps - you can use any project management application (a range of these, with advantages/disadvantages was provided in L30). Most useful are those which provide repositories which allow members to "commit" work to be viewed by others.

An inability to meet in person will not be accepted as a reason for not completing coursework. Iif you use a project management application, it will be easy to provide evidence when a team member is not making an equal contribution, or is not responding.

**2.4.1 Agile system development using Sprint Cycles**

* Based on **User Stories &** **Client Priorities** and analysis of how to develop the system, at the start of each Sprint Cycle, your team 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)
* You may append screen shots from project management software to Sprint plan documentation , in order to provide evidence of project management
* Sprint Cycle Documentation must be signed by a sprint master (staff member) by a deadline
* Sprint Plan 1 **by Wed 20th Feb**
* Evaluation of Sprint Plan 1 & Sprint Plan 2 by **Wed 6th March**

Evaluation of Sprint Plan 2 & Sprint Plan 3 by **Wed 20th March**

2.5 Using EasyPHP Platform and CodeIgniter framework

**2.5.1 EasyPHP & CodeIgniter**

The EasyPHP platform may be downloaded from the following location: [**https://drive.google.com/file/d/0B0Ou1lzzphIJbE12UncwQlQyb2M/view?usp=sharing**](https://drive.google.com/file/d/0B0Ou1lzzphIJbE12UncwQlQyb2M/view?usp=sharing)**.**

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/>   
  
A MySQL database is provided within the EasyPHP platform. You can view the database via the included application phpMyAdmin. Once EasyPHP is running, use this 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 match 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>

See also the supporting teaching materials posted on StudyNet under lectures, tutorials, or lab exercises. You should also consult the **Development Platform FAQs** under StudyNet practicals folder for queries/problems.

**2.5.2 Support for the Software Implementation**

Support is available to clarify both the **MICE system requirements** and for **technical information**, but we would like to direct questions to the appropriate staff member.

* for system requirements, **post Studynet thread prefixed “GP3: requirements - ...”** for example “GP3: requirements - is there a target screen size ?”.
* for technical support, **post Studynet thread prefixed “GP3: technical - ...”**

You can start a new thread if you like but be sure to check all existing threads and documentation for answers to your question, we will not answer questions we have already answered.

You can use the practical lab clinics to get support for coding, but we will not accept requests by email for new information nor clarification - since, to be fair to everybody, we wish to make such information available to all. We may also post updated information if we become aware of general issues (so please check )

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, you may wish to use your wireframes from GP2 to support the overall design for your user interface. ***However, producing/submitting new wireframes is NOT part of this GP3 submission.***
* 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.
* Note: a test format was provided in L31 & L32 and you would be wise to use those early tests
* Produce a draft of the data model - *best done by discussion in the team*
* read Section 2.2 to identify and name entities attributes 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 2.3.2 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 MICE 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 MICE.
* The most efficient way to develop the system is NOT to implement the whole DB and then try to code the functionality BUT to start by implementing no- or low-dependency tables together with their interfaces to access their data. small sub-teams can develop these in parallel and test each others work, so all team members get a chance to design, code and test
* Do not leave all of your testing to the end. Design tests BEFORE you start coding. Then test code as it becomes available.
* 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
* 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 4 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 Overview

**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 (printed copy)
* your test plans (printed copy)
* a completed “GP3 Roles and Contributions Form”, (individual forms if you disagree as a team; see below);

3.1.2 For your demo

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 MICE-OFF should:

* implement the functional and non-functional requirements given in *[Section 2.3];*
* 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 MICE have actually made appropriate changes to the database;
* store all the sample data provided by the client.

While 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 staff to assess your software against the “User Acceptance Test” checklist (see additional document) to establish specific input/amendment/retrieval facilities.**

3.4 Sprint Cycle Plans

See section 2.4 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. Examples which you can use as a starting point, were provided in L31 & L32. Try to maintain this during development instead of doing it at the end

3.6 The Demonstration

The assessment of your MICE-OFF 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 MICE-OFF (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 MICE 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 (no more than 5 minutes) presentation with maximum 6 slides 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 01/04/19 by 13.00 , 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 MICE 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 and data dictionary)
  + your 3 Sprint Cycle Plans
  + your test plans
  + completed GP3 Roles & Contibutions Form
  + evaluation of user interface (if seletected in UAT part 5)

1. Submit to **Studynet**, by Mon 01/04/19 by 15.00

* 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 MICE (i.e. root folder for MICE as stored under EasyPHP/data/localweb)

Only **ONE COPY of each is required per team**. Diagrams/Sprint Plans should be scanned for electronic copy.

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, i.e. complete EasyPHP stack.
* 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 10 and the Firefox web browser (not just on your own PC) before submitting your USB.***

Part 5: Assessment

5.1 Marking Scheme

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 and the assessment of the clients [module staff]. 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 MICE demonstration will be assessed *[marked]*.

* The UAT shows the **type** of tasks that you will be asked to perform during a demo of your MICE system.
* At the demonstration, the client will tell you what actual data to use.
* It is essential that your DB stores **ALL** data provided in “**Clients Sample Data & User Stories”.**
* **NOTE**: The client may find it necessary to make amendments to the content/order of the tests in the UATIf this happens, you will be alerted (by StudyNet) and a UAT with a new version number will be posted

Always consult the latest version.

* The **UAT** provides you with an assignment feedback form.

5.1.2 Questions at the Demo

During or after the demo, the client [module staff], may ask each individual team member one or more questions about your system, your software, or your contribution to the work of the team. While it is not essential for all team members to be involved with all of the tasks, it is expected that each member makes a significant contribution:

Based on a member's demo performance, answers to questions and roles listed on the GP3 Roles and Contributions Form and sprint cycle forms, staff reserve the right to change the % contributions attributed to team members

5.1.3. Marking Scheme – Team Contribution

Your team should consider and submit the “**GP3 Roles and Contributions Form”** posted with the assignment. This asks you to describe the role of each individual team member and provide a total 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 listed on the “**GP3 Roles and Contributions Form”** (or adjusted by staff as listed above) is significantly **lower than the norm** (20% for a team of 5, 25% for a team of 4, etc.), they will be awarded only a proportion of the assignment mark (as for GP1 & GP2) shown

|  |  |
| --- | --- |
| **Contribution difference from the norm** | **% of assignment mark** |
| < 5% difference | 80% |
| >= 5% AND <10% difference | 60% |
| >=10% AND <=15% difference | 50% |
| >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 an initial **Data Model** (E-R diagram) |  |
| **3** | Decide on priorities. Produce Sprint Cycle1 Plan. Get it signed by the Sprint Master  **Carry out the planned the tasks**, including:   * produce test plan/s * develop your software * test your software against the test plan using 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. Produce Sprint Cycle2 Plan. Get it signed by the Sprint Master  **Carry out the planned the tasks**, including:   * produce test plan/s * develop your software * test your software against the test plan using client sample data * 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. Produce Sprint Cycle3 Plan. Get it signed by the Sprint Master  **Carry out the planned the tasks**, including:   * produce test plan/s * develop your software * test your software against the test plan using client sample data * 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 GP3 |  |
| **6** | Produce **the deliverables for the coursework hand-in:**   * your database design documentation (ERD and data dictionary) (printed copy); * your software, including the specimen test data (on a USB);   + - test that your software works on the lab PCs (Windows 7 and Firefox) * your 3 sprint cycles plans, signed & evaluated * test plans & sprint cycles |  |
| **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) for submission. |  |
| **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 01/04/18 11.00am |  |
| **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). ONLY **ONE** submission per team is required   * 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; * allocate roles (who will click through the system for the software demo, and who is speaking during the presentation). |  |
| **13** | Produce **the deliverables for the demo:**   * bring your presentation slides, and be ready to display/present; * be expected to load your software (submitted USB) on a lab machine during the demo as instructed by staff |  |
| **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! |  |