

University of Wollongong
School of Computing and Information Technology
CSIT314 Software Development Methodologies
SIM S4 2020

Group Project (40 marks)

TASKS

Your tasks are to:

1. Carefully read this document;
2. Form and structure your group, allocating roles and responsibilities to your members. **Make sure you communicate frequently and contribute significantly to your group work. If you have little contribution to your group project, your individual mark will be significantly lower than your group's mark (or even 0 mark if you have almost no contribution).**
3. Complete the development of a given software system (enclosed in this document). This should cover **all software development activities** from requirements elicitation and specification, design to implementation and testing. **Your design must be based on the b-c-e framework discussed in the subject.**
4. Choose an agile method (Scrum is highly commended) for your group to follow. Choose a tool which supports that methodology for your group to use. **Taiga <https://taiga.io/> for agile method is strongly recommended** (choose the public option which allows multiple team members).
5. Demonstrate the practice of test-driven development and Continuous Integration/Continuous Delivery (CI/CD) in your project.
6. Produce a report detailing the group's work.

SUBMISSION

1. Final deliverable (**35 marks**): **20th November 2020**
 - Final report
 - A 15 minute recorded video of a live demo of your product
 - Source code
2. Final presentation (**5 marks**): **The last lab session**

All submission must be made to Moodle by one member of your group by the deadline.

GUIDELINES

1. The **final report** should cover at least the following:
 - A complete list of user stories.
 - For each user story, a list of complete tasks.
 - A complete and detailed design including **UML use case diagrams, use case descriptions, sequence diagrams, class diagrams, data persistence and user-interface aspects.**
 - **Sufficient evidence (with screenshots and detailed text description/explanation) to demonstrate that your group has followed an agile methodology and has used a tool which supports the methodology**

from the beginning to the end of the project. Note: **make sure that you record these evidence every week (e.g. screenshots of the product backlog, sprints, the work-in-progress software system, etc.)**

- Test plans, test cases (include **unit test cases**), and test data that is sufficiently large enough to simulate the scale of the developed system. Details of unit testing procedures that have been conducted to clearly demonstrate (with sufficient evidence) that your group has followed test-driven development
- Sufficient evidence (with screenshots and detailed text description/explanation) to demonstrate the use of CI/CD in your project (i.e. the development and deployment of **at least** one feature/functionality).
- Identify and discuss ethical considerations in developing the software system in this project and how your team have addressed them.
- Identify a feature of your software application that can be developed using data-driven software development. Present a detailed plan of how this feature would be developed and integrated into your software application using the data-driven approach discussed in the subject.
- True group meeting records: agendas and meeting minutes which includes at least the following: meeting date, attendance, progress reports, review and tracking (e.g. snapshots of **Gantt chart** tracking or backlogs, etc.), discussion summaries, and action plans/items.
- **Member contribution for the whole project (with each member's signature):**
 - On the cover page of your progress/mid-project/final report, you need to provide **rating** for the contribution of each team member and a **detailed explanation of what the team member did for the project** to justify the rating (e.g. the roles that they filled, the tasks they completed, and the artefacts that they successfully delivered).
 - Everyone in the team should sign the cover page. The individual contribution of each team member is assessed by all the other members.
 - The rating scale can be a percentage number (e.g. 60%). Alternatively, the scale be in the form of “contributed”, “very little”, and “almost no contribution”. For a team member who has “contributed”, he/she will receive 100% of the group mark; for a team member who contributed “very little”, he/she will receive 50% of the team mark; for students who made “almost no contribution”, he/she will receive 0 marks for the entire group project. Your tutor/lecturer may make adjustment to this marking criterion based on practical situations.

A suggested plan:

- The first week after the lectures: finalize group and start working on the project. Produce the first complete version of the requirements.
- One **iteration/sprint per week** until the end of project. In each iteration:
 - Design, implement, and test a number of functionalities;
 - **Demonstrate the working system to the clients (i.e. tutors) to receive feedback during the second half of weekly labs.** *Weekly progress will be observed and noted by the tutors and will contribute to the project management component of your project marks. Check the marking scheme in the last page for details.*
 - Continue eliciting and clarifying further requirements.

The project description is in the next page.

Project Description

Important Notes:

- *This Project Description provides only the **high-level goals** of this project. The development team **MUST elicit more detailed and specific requirements AND get feedback from “the client”** (the tutor).*
- *The tutors will note your group’s progress and interaction with the “client” since they are one of the factors contributing to the final marks.*
- *The requirements may change during the course of the project (this is to simulate real-life projects).*
- *At least the backend/middleware of your software product (i.e. the main code that controls/runs all application logic and hold data in memory) needs to be object oriented.*
- *You need to form a group of 6-7 people in your same lab ASAP and register your group with your tutor (see below):*
 - *Full time cohort: Terence Chew (tchew@uow.edu.au)*
 - *Part time cohort: Kheng Teck Tan (ktan@uow.edu.au)*

You are asked to develop a bug-tracking system using **object-oriented design and implementation**. Using this system, users can report bugs, triage bugs, and manage a bug’s lifecycle (see https://bugzilla.mozilla.org/show_bug.cgi?id=1429672 for an example).

This system has different user types: **bug reporters, triagers, developers, and reviewers**. Reporters here can be end-users, testers, and any other users. Developers are those assigned to fix a bug, and hence their expertise and experience (e.g. the bugs that have been fixed by them) are important. Reviewers check if a bug has been fixed with the patches provided by the developer and close it accordingly. **Triager** is responsible for managing all quality aspects of a bug, such as assigning it to a suitable developer, checking if it is a duplicate or invalid bug. All of these must be kept and managed by the system.

Throughout the lifecycle of a bug, users can participate in discussing how to fix it (by providing comments). User can also search for bugs through keywords, titles, assignee, etc. The system can also generate various reports such as the number of bugs reported in a month, the number of bugs resolved in a week and the best performed reporters or developers.

You must create test data that is sufficiently large enough to simulate the system (e.g. at minimum 50 users and 500 bug reports). You could write a script to randomly generate these data.

The marking scheme is in the next page for your reference.

Marking scheme

<i>Component</i>	<i>Out of</i>	<i>Marks</i>	<i>Comments</i>
Final Project Presentation (Recoded demo video + Q&A session)	5		
Final Deliverables			
Overall quality of the Final Deliverables	2		
User stories and tasks	8		
Analysis and Design (use cases, detailed design models, consistency between design models, consistency between design with code, etc.)	9		
Implementation (quality of code, functionalities implemented, sophistication of the solutions, consistency with design, etc.)	6		
Test-driven development	2		
CI/CD	2		
Ethical consideration and discussions	2		
Data-driven development	2		
Effective use of methodologies (e.g. evidence of the use of a methodology and tool support, true meeting records, weekly progress as observed and noted in the labs , etc.)	2		
Total	40		