SOFTENG 306 PROJECT 2

Version 1.1

Dr. Seyed Reza Shahamiri
Department of Electrical, Computer, and Software Engineering
Faculty of Engineering
The University of Auckland

This document provides information about SoftEng 306 Software Engineering Design 2, Part II Project (aka Project 2). For this project you are provided with a case study and starter code in which you need to 1) reverse engineer the case study to its blueprints, and 2) refactor the case study design as requested in this document. Please read this document carefully.

One of the team members must be appointed as the *Team Coordinator*. This student is responsible for keeping the team together, organize and host meetings, keep track of progress, communications, submit the project deliverables on behalf of the team, etc. The team coordinator can be responsible for appointing the tasks to team members or the team can decide to be self-managed, either way **task allocation, completion, and deadlines have to be documented properly and supplied to the teaching team**. The team coordinator will also be the point of communication with the teaching team. To compensate for the additional tasks performed in the project, the team coordinator workload should be 20% less than the rest of the team.

1. ASSESSMENT CONDITIONS

- To be done in teams of **7** students (with no more than **three** members from Project 1 team), with both team and individual assessment components.
- Open book
- No time limits (please refer to the due dates for each deliverable)

2. TEAM FORMATION AND AGREEMENT

You are to immediately form your team and allocate your team coordinator. Furthermore, to make team working as hassle free as possible, you are provided with this team agreement template. Teams are required to complete and submit the agreement on Canvas by the end of the first week of Part II, which is the same deadline to confirm your team. Please note that the team agreement is mandatory, but it is not marked. Failure to supply the agreement may result in 20% penalty in your overall mark.

3. PROJECT MANAGEMENT TOOL

It is recommended that you use electronic project management tools, such as **Trello,** for task allocation, tracking, etc**.** Please ensure that you provide access to the teaching team.

4. GITHUB CLASSROOM AND STARTER CODE

All students need to use GitHub Classroom to work on the project collaboratively. Team coordinators, please follow this link to setup your teams and get access to the starter code.

5. THE CASE STUDY[1]

The case study is a Student Management System written in Java. Once you clone the repo from GitHub Classroom link provided above[2], you should:

[1] The case study was not developed in the UoA. We have received authorization from the original developers to modify and use the source code for the intention of this project. Students are not to commercialize this project.

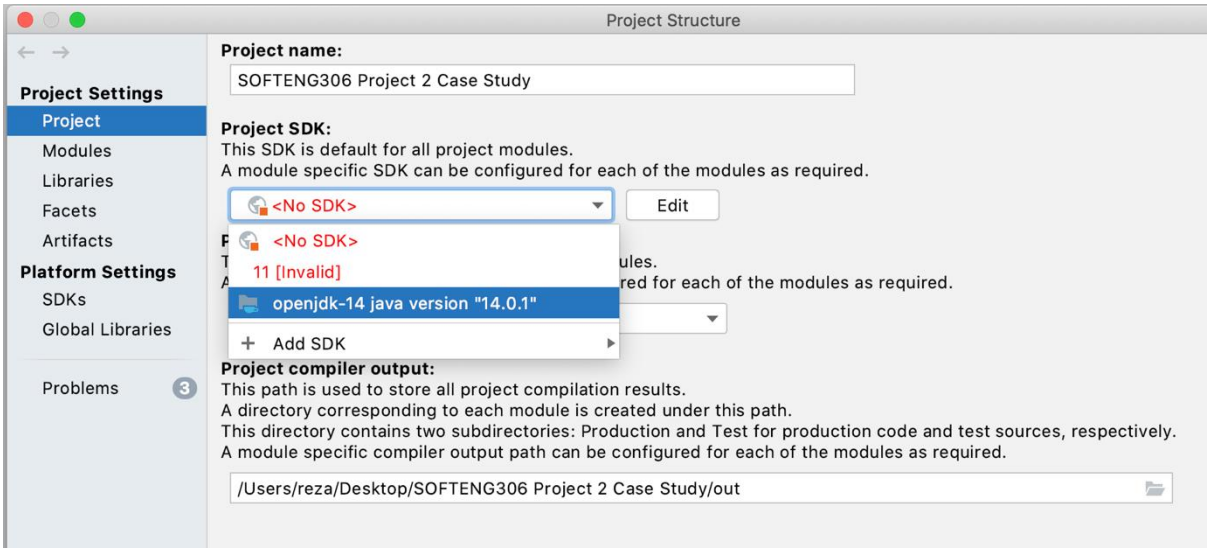
[2] Alternative link to download the started code here. However, all students must use GitHub Classroom.

a. Set up the development environment

Download and install IntelliJ Idea. You can either use the free Community version or use your UoA student account to get the Ultimate version from this link for free.

b. Open the case study in the IDE and set SDK

The first time you open the source code you may need to set the JDK version for the project. You can go to File -> Project Structure and then set Project SDK as shown below. If no SDK is available, you can Add SDK.



c. Set up GitHub

You are required to properly set GitHub Classroom and commit changes as you go through the project. Every document and code created and used for the project, including task allocation and tracking documents (if e-tools like Trello not used), meeting minutes/notes, etc. must be in this GitHub repo as we refer to each student's commit and code participation history to assess his/her participation, meeting deadlines, team commitment, etc. **Do not commit all your changes at the end of the project or you may receive a zero mark if you cannot provide us evidence of your ongoing commitment to the project and team.** Lack of continuous commit history from the beginning of the project results in, at least, 30% penalty in your mark.

Please note the team repo is not an indication of submission and all deliverables must be submitted via Canvas as well. Your repo must be private and only be shared with the teaching team. **You cannot share anything about this project with other students if they are not your Project 2 team member.**

6. PROCESS

This project is conducted in two phases, a reverse engineering phase and a refactoring phase, as explained below.

a. Phase One: Reverse Engineering

The objective of this phase is for you to understand the *as-is* system better. As you may have already noticed, the source code does not include any documentation. In this phase, you are required to study the code and the behavior of the software and generate its class diagram and use case diagram. For the use case diagram, you only need to supply the high-level diagram and

briefly explain the use cases. You are also required to measure $WMC_{Fan\ In}$, DIT, and CBO for each class supplied in your class diagram. Details of these metrics are provided in the lectures.

You can use tools and libraries to generate the requested diagrams and metrics automatically. However, be mindful that tools' generated diagrams may not be completely accurate and sometimes miss some information. For example, they may not include static associations between classes, consider class dependencies and associations differently, overuse aggregation/composition, etc. As such, you must review the deliverables are both complete and accurate, especially when they are generated automictically. As an illustration, a class must be associated with at least one other class with no dead (i.e., unreachable) class, or two classes should not have multiple associations with each other.

Please note that you do not make any change to the as-is system in this phase.

b. Phase Two: Refactoring

After understating the as-is system, your task is to analyze the system to identify design smell(s) and propose a revised class diagram design that removes those smells via applying SOLID principles, and then re-measure the metrics mentioned above and critically compare them via the as-is metrics you measured in phase one. You need to explain the smells you identified and how your proposed design is better (or worse) than the as-is design by objectively analyze different software quality attributes.

You are also required to modify the given starter code and implement the refactored design without modifying the user interface. You must design and implement at the same time to ensure your design is feasible and viable.

7. DELIVERABLES

a. Design Doc (10 marks)

This is the first deliverable of this project to be submitted at the end of Phase one. The Design Doc should include all information requested in the Reverse Engineering phase to explain the as-is system properly. Particularly you need to provide the following information:

1. The team and member names
2. A table to link your GitHub user name to your real name
3. Introducing the software. Design a high level "use case diagram" to explain the primary functionalities of the software and provide Use Case Descriptions to explain use cases identified. For a template for the use case descriptions and more information on the diagrams, please refer to the resources section of this document. Ensure that all relationships are identified correctly, and the diagram sounds both semantically and grammatically.
4. The structural design of the system. Extract the class diagram of the as-is system. Again, ensure all classes are identified, associations and their types and cardinality (one-to-one, one-to-many, etc.) are correct, and classes include all their fields/attributes, properties, and methods with their access type. See class dependencies (for example, when one class object uses a field or calls a method of another class object) as a simple association between two classes.
5. Measure $WMC_{Fan\ In}$, DIT, and CBO for each class supplied in your class diagram.
6. Beefily explain the roles and responsibilities of each member in this phase. Not submitting this information results in 20% penalty of the overall mark for this deliverable.

All the above information should be supplied in one document and to be submitted on Canvas by the team coordinator (this is the same for the other deliverables too). In case the class diagram is

too big to fit in one page, present it as a separate JPG file and supply its link to your GitHub repo in your report. Please note that handwritten diagrams and/or photos of handwritten drawings are not acceptable; **it is your responsibility to ensure every figure and information supplied is easily readable.**

You will be assessed based on the following criteria:

Criteria	Mark
The introduction of the system. For example: • The system is properly introduced • The use case diagram is accurate, reflects the primary functionalities of the system, semantically and grammatically correct, and relationships are correct and identified accurately • Use case descriptions are supplied and follow the template • Etc.	3
The design of the as-is system. For example: • All classes are identified in the class diagram • Class associations, their types and cardinality are provided and accurate • Class details are captured correctly • No dead class • No multi-associations between any two classes • Etc.	3
$WMC_{Fan\ In}$ for all classes provided and accurate	2
DIT for all classes in an inheritance hierarchy provided and accurate	1
CBO for all classes provided and accurate	1

b. System Refactoring Report (20 marks)

This report reflects on the second phase of the project and explains your refactored system based on the as-is system. You are required to provide the following information:

1. Identify any design smell(s) and explain them (what smell(s), where, why), and whether the metrics you measured in phase one are any indication of those smells (3 marks).
2. Propose a new class diagram for the system that removes the smell(s) you identified. Supply it as a separate high resolution JPG file too (5 marks).
3. Explain how you applied SOLID principles, and which principles you considered, to refactor the class diagram, and how the smell(s) are removed or improved, or even got worse (5 marks).
4. Re-measure the three metrics used in the previous phase for all classes and interfaces in your proposed design. Compare and analyze them with the as-is metrics and explain whether there have been any improvements. It is possible that your proposed design improves some metrics but worsen others. If so, explain why you think this happened (4 marks).
5. Analyze, compare, and explain whether the new design has seen improvements in respect to maintainability, reusability, coupling, and coherency compared to the as-is system. Your

judgment should be objective and based on the metrics you measured. If needed, measure other metrics to justify your arguments (3 marks).

6. Briefly explain the roles and responsibilities of each team member in this phase. Not submitting this information results in 20% penalty of the overall mark for this deliverable.

Please limit this report to maximum 15 pages, Calibri font size 11, double spacing. The team coordinator to submit this report (and the class diagram JPG) on behalf of the team.

c. Refactored Code (20 Individual Mark)

Here you are required to submit the implementation of the refactored system based on the designs you proposed via the System Refactoring Report. It is highly recommended that you implement the refactored system while working on its design instead of waiting to submit the previous deliverable and then starting to code. Remember that software development is an iterative process in which your code and design highly affect each other, and SDLC phases are interleaved. Hence, start refactoring the code while exploring your design options, and only use the extra time between submitting your System Refactoring Report and Refactored Code to finalize and polish the code rather than developing it from scratch.

The calculation of your mark for this deliverable is based on:

1) Your mark for System Refactoring Report - This is the *maximum* mark you may receive for this deliverable, which means losing marks on your Refactoring Report will be reflected in your code.

2) Whether the implementation and proposed refactored design are consistent.

3) The refactored system does not crash.

4) The system still delivers the original functionalities. **Your refactoring should not change the functionalities of the system or add new functionalities as long as end users are concerned** hence ensure the user interface is not changed during the refactoring process**.**

Based on the criteria mentioned above, every project will receive a mark. Then, individual marks will be calculated and allocated based on your Participation Report, explained below.

The team coordinator is to compress the code folder and submit it via Canvas.

d. Participation Report (unmarked, mandatory)

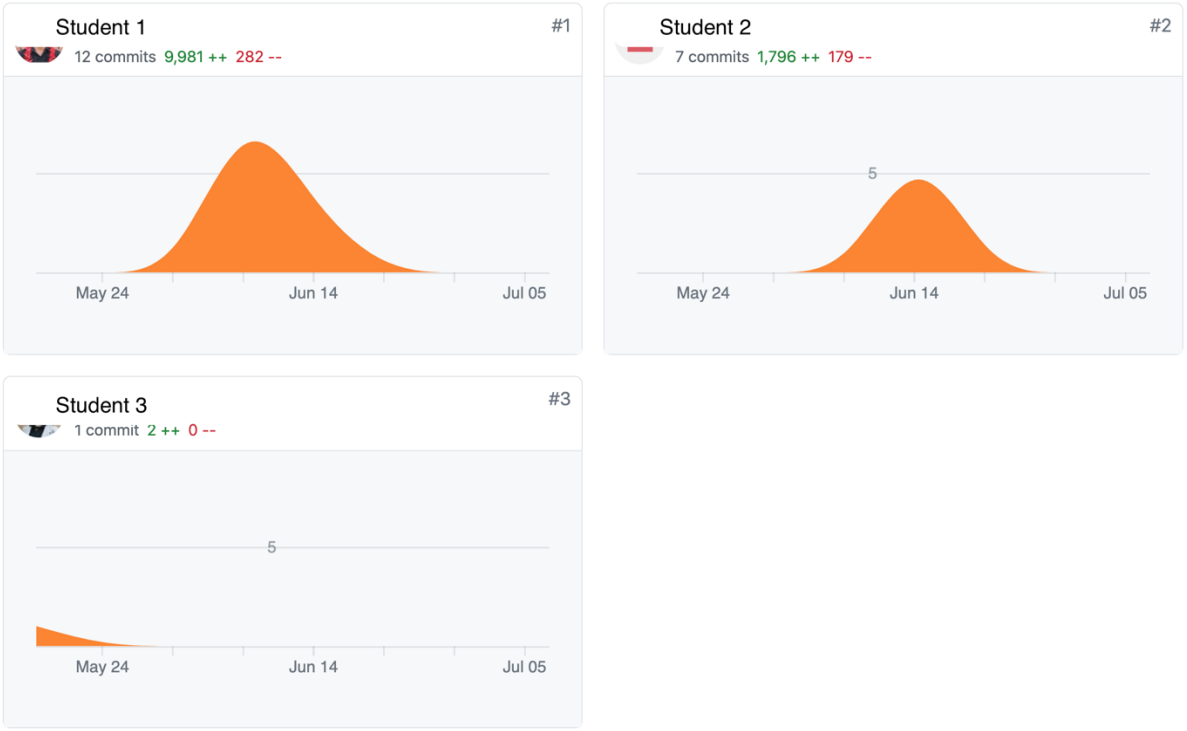
Individual participations will be identified based on two factors:

1. *Your GitHub Performance,* which will be identified based on your ongoing commitment to the project based on your commit history and code participation.

2. *Peer Review:* The project team is required to complete and supply the following **participation table** in which **participation factors** (from 0 to 100%) are given to each team member. Reiterating that the participation factors must be based on the evidence collected during the project, schedules, attendance, git commits and code participation, etc. In case of any disagreement between the team members, the team coordinator conducts voting and the participation factors with the majority vote to be assigned for each member. You should also include information about any disagreement raised.

Student Name	Criteria	Participation Factor (0-100%)
	Attendance and Engagement	
	Task Completion	
	Meeting Deadlines	
	Professional Attitude and Behavior	
	Knowledgeful and Currency	
	Code and GitHub Commit Contributions	
Average Participation Factor (%)		

If the participation of a team member is identified to be considerably lower than others, the individual team member may lose on not only the individual mark but also any team mark for that individual will be affected. Hence, **marks given to any deliverable may change at any point**. For example, suppose a team's GitHub contribution is as shown below. In this case, Student 2 may only receive 16% of the mark given to the project while Student 3's mark may be zero:



It is the students' responsibility to inform the teaching team of any issues raised, including lack of member(s) participation, well before the due dates so that we have enough time to intervene and assist when needed. Otherwise, we rely on the information available to decide on individual participations.

You need to supply all the above information in one document per project submitted on Canvas by the team coordinator. Kindly be advised that this deliverable is mandatory, and failure to provide it will result in a zero mark for this project for all students.

8. INTERVIEW

For some teams, we may need you to attend an interview with us. We may have questions regarding your submissions, member participation, etc. If this is the case, attending the interview is mandatory, and you may receive a zero mark for this project if you do not attend without any

reasonable excuse send to us in writing before the interview date. Please get in touch with us immediately if you cannot attend the interview to avoid a zero mark.

9. PROJECT 2 SCHEDULE

The project schedule is provided below. The due dates are supplied on Canvas.

	To Do	Deliverables
Week 7	• Formation and confirmation of teams • Appointment of the Team Coordinator • Lecture	Team Agreement
Week 8	Lecture	
Week 9	Lecture	Design Doc - Friday
Week 10		
Week 11		System Refactoring Report - Sunday
Week 12		Refactored Code - Friday Participation Report - Sunday

S

10. RESOURCES

In case you need to refresh your memory with the requested UML diagrams, you can refer to your previous courses or online materials, or the following lessons: Use Case Diagram, Class Diagram.