The University of Melbourne
School of Computing and Information Systems
SWEN90007 Software Design and Architecture
Semester 2, 2022

Project Overview

In this subject, we will discuss a variety of design principles and architectural patterns for enterprise applications. In this project, students will have the opportunity to gain practical experience in the implementation of these patterns by developing a large-scale enterprise application.

The project aims to give students the experience to:

- Produce an architectural design for an enterprise system.
- Choose suitable design patterns to be applied for the different architectural layers.
- Implement an application for your design.
- Develop employability skills: teamwork, good communication, collaboration, problem-solving, critical-thinking, adaptability, initiative and leadership.

Application Domain

This year, you and your team will choose one of the two available projects to implement (descriptions available on Canvas and on GitHub). Make sure you read them all carefully before you make your decision.

- Project Idea #1: Hotel Booking System [https://canvas.lms.unimelb.edu.au/courses/130935/modules/items/3967460]
- Project Idea #2: Marketplace System [https://canvas.lms.unimelb.edu.au/courses/130935/modules/items/3967461]

Programming Language and Architecture

The system that you design and develop must be a web-based system.

The back-end (i.e., server-side components) must be implemented in Java (no exceptions). The use of frameworks to facilitate any (back-end) infrastructure is not allowed. The idea is for you to implement the patterns from scratch, rather than using a framework with the patterns built in. This will give you hands on experience, strengthen your ability to design and architect applications, and deepen your understanding of the patterns.

For the front-end (i.e., UI/client-side components) of your application, you are free to use any technology/framework/library of your choice. Keep in mind that all workshop and lecture materials will focus on the use of JEE technology, namely JSPs + Servlets. We will provide resources and support on how to set up and use this technology during the lectures and the workshops. We also ask that you consider the fact that integrating JavaScript web UI frameworks with Java-based back-ends may be a non-trivial task. It is your responsibility to design and implement the integration between the front and back ends of your application (e.g., by exposing your application services via a REST API and accessing them using AJAX). We will not cover, nor provide support, on how to do this.

All design diagrams (domain model, class diagram, use case diagrams, sequence diagrams) must be in UML.

Plagiarism

All code and documentation submitted as part of this project must be your own team's work, it must be based on your own knowledge and skills. Collaboration between teams is not allowed, if we find that a submission has been copied from the work of another team, all students submitting the copied work will receive zero marks for the assignment. Similarities will be determined by Turnitin (reports) and Moss (for Measure of Software Similarity – source code).

Teams

This assignment will be carried out in teams of 4, no exceptions. Team members must be enrolled in (and attend) the same workshop. This is because the workshops have been structured to support your teamwork, e.g., by allowing time for the team to work in the project and consult queries with the tutor.

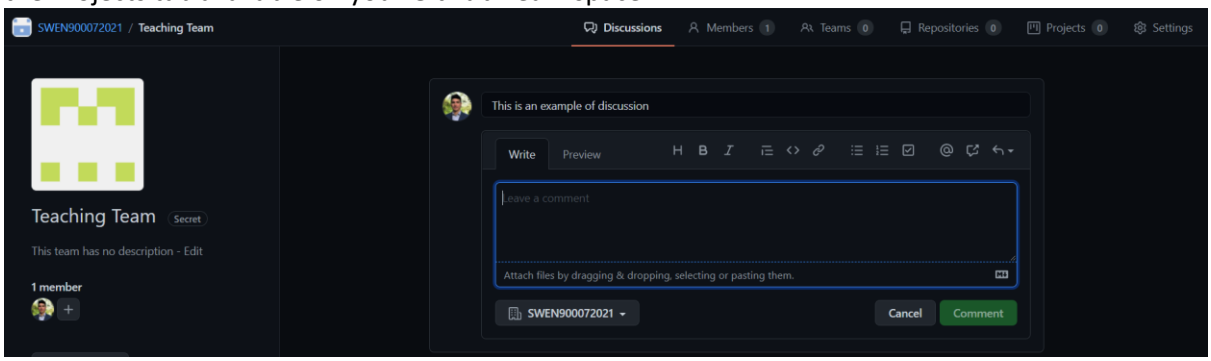
This is a semester long project that will require constant interaction and contribution from all team members, we strongly encourage you to create your own teams and make a conscious choice of whom you will be working with. You are free to team up with friends or peers you are already acquainted with. Students without a team will be randomly allocated into teams by Eduardo and Maria.

A survey to submit your team was created in Qualtrics https://melbourneuni.au1.qualtrics.com/jfe/form/SV_0MzJAczpZTNdtJQ (password: SWEN90007_2022).

Use of Code Repositories and Collaborative Tools

For this project, we will adopt GitHub Classroom. Your team will be created by the teaching team and will be available here: https://github.com/SWEN900072022 (you'll be able to access this space once you receive our invitation in Week 2). After that, you'll be able to create your private repositories. **Your repository must be kept as private**.

Once your team and repository are created, we strongly encourage you to explore the Discussions and the Projects tab available on your GitHub Team space:



This repository must be used for the duration of your project to manage source code and related configuration items. You should also use it to store a copy of your design documentation (e.g., class diagrams) and the reports submitted for each stage. Make sure you structure your repository properly.

Example of repository structure:

```
├── docs/       # Docs (you can create subfolders to organize your project reports and minutes)
├── │── part1/       # Part1 report
├── │── part2/       # Part2 report
├── │── part3/       # Part3 report
├── │── part4/       # Part4
├── │──data-samples/     # Data (inputs) necessary to simulate/demonstrate your project
├── src/         # Source code
└── README.md
```

README file must explain your GitHub repository: what's included, link to your deployed App, changelog, details about team members and so on.

At the end of each deliverable, generate a RELEASE TAG from your repository (master branch).
Format: SWEN90007_2022_Part<number>_<team name>. Make sure you add your release tag to the checklist and README file.

Keep in mind that your Git repository will also be used to assess individual contributions. This means that it is very important that you commit and push your work frequently. It is also good practice to do this as it protects your work from accidental crashes or deletions!

For your submission reports, you are free to choose a collaboration mode that works best for your team. We suggest using GitHub Teams, it facilitates teamwork, stores copies of your work, and keeps a log of individual contributions to the document. This may come in handy during the peer review process if disputes about contributions arise in the team.

Peer Review

During this journey, students will be asked to participate in two peer assessment survey (Week 6 and Week 11).

This is an individual mandatory task. Everyone needs to participate, assess and provide feedback to peers. Regardless of having a great or bad team experience, students will be encouraged to provide feedback. The survey shouldn't take students longer than 10 minutes to be completed. Students' guidelines for peer review will be provided in the LMS.

The project deliverables (Part 1, Part 2, Part 3 and Part 4) will be weighted by students' peer review scores. These are continuous assessments focused on communication, teamwork, engagement, commitment and so on. It is essential that everyone can contribute equally to group work. The peer assessment form will give us (teaching team) an indication when this is not the case.

Teamwork during Workshops

Workshops will be mostly dedicated for project teamwork. You are encouraged to think of this time as your team's weekly meeting. You can use the time to plan activities for the week ahead, work collaboratively on a task, discuss any issues that need to be resolved, etc. Most importantly, your team should use this time to seek clarification and guidance from the tutor.

You are required to take minutes during these workshop team meetings. The minutes should contain:

- Workshop date/week.
- Team members that attended.

- A few bullet points summarizing the team's activities during the workshop meeting.

These minutes must be stored in your Git repository (docs/meeting-minutes/) and will be used as part of the peer review process.

Detailed Project Description

Part 1 [15 marks]

Task:

Once you decide on your project description, identify the use cases necessary to meet the requirement specification for the application users.

Deliverables:

1. A report, in pdf format, that includes the following:
   - Your team's name, team members' names, student ids, unimelb usernames, github usernames and emails.
   - A use case diagram that illustrates the interaction between the actors and the use cases, as well as any relevant relationship between use cases.
   - A simple description of each use case. Each use case must contain the list of actors associated with the use case, a high-level description of the use case (Appendix E – Simplified use cases).
   - A domain model with an explanation of it (description for domain model).
   - A Git release tag: you must create a release tag in your Git repository for this deliverable in the following format: SWEN90007_2022_Part1_<team name>[1]. The tag must be created before the submission deadline as this will be used to assess your deliverable (no exceptions). The created release tag must be added to your report.
   - This report must be available in your GitHub (docs/part1).
   - Part1 checklist (Appendix A – Checklist for Part1 Deliverable) must be submitted to Canvas assignments.

Part 2 [40 marks]

Task:

In this part you will design and implement your app as described in Application Domain and as defined by the use cases identified in Part 1 (the use cases are specific to each team and project and must be approved by the teaching team). At this stage, you will assume that your application is only accessed by one user at a time (i.e., there is no need to handle concurrency yet).

You must include the following patterns in your design and implementation (see Appendix F – Expanded use cases and Appendix G – Architecture Document for more details):

- Domain model
- Data mapper
- Unit of work
- Lazy load
- Identity field

[1]Git Basics – Tagging: https://git-scm.com/book/en/v2/Git-Basics-Tagging

- Foreign key mapping
- Association table mapping
- Embedded value
- One of the inheritance patterns
- Authentication and Authorization (you are free to use any library/framework to implement these patterns)

Deliverables:

1. The application coded and deployed. The source code of your application and updated documents must be committed to your Git repository. The application must be deployed in Heroku.
2. You must create a tag in your Git repository for this deliverable in the following format: SWEN90007_2022_Part2_<team name>. The tag must be created before the submission deadline as this will be used to assess your deliverable (no exceptions). After creating the release tag, you deploy this tag (not your main repository branch).
3. A Software Architecture Design (SAD) report with the following items:
    - The class diagram of your application.
    - A description of all patterns used. The description must be contextualized, this means it must include details of how the pattern was implemented as part of your application design. The description must include relevant diagrams (e.g., sequence and entity relationship diagrams) that illustrate the use and the implementation details of the patterns in your application.
    - Design rationale for unit of work and lazy load: an explanation on where the patterns were used and why.
    - This report must be available in your GitHub (docs/part2).
    - Part2 checklist (

- Appendix B – Checklist for Part2 Deliverable) must be submitted to Canvas assignments.

4. README.md updated to include the link to your Heroku deployed app
    a. Your deployed App must include a populated database with a range of realistic data samples/information that are necessary for the teaching team to test your application. You can create a document explaining your database, add it to docs/data-sample and add the information to your README.
5. README.md must also provide additional instructions on how to use the existing data that you created in your system, for example: the administrator username and password. We do not expect to create any test data from scratch. Again, this is your project. Make sure you test your deliverable before your submission and that you have real/appropriate data in the deployed system. Meaningless data such as abc, 123, blah blah, subject ASDFG and so on will compromise the assessment of your project (and your final marks). You can create a document with data samples, add it to docs/data-sample and add the information to your README. See Appendix H – Test Cases and Data Entry Document for more details.

Note that for the UML diagrams, it is your responsibility to decide the level of detail required to appropriately convey the design and implementation of your application. Use your judgement and knowledge in software engineering to make this decision; another software engineer, new to your team, should be able to implement your design based on the information contained in the SAD report.

Part 3 [35 marks]

Task:

In this part, the ability to have multiple users in the application will be enabled. This gives place to concurrency issues that you must address.

You will need to address different concurrency issues and scenarios depending on the project topic you chose and your design and implementation of the application. Below are some examples of concurrency issues that may arise across the two given application domains:

- Multiple customers can try to book the same hotel room simultaneously.
- Multiple customers can try to purchase the same set of goods simultaneously.

It is your responsibility to identify all cases in which concurrency might cause issues in your application and address them by making use of the concurrency patterns covered in the subject. For each case, you can choose to use a pessimistic or an optimistic online lock (or both). In your report, the following must be included for each concurrency issue you identify:

- A clear description of the issue.
- The choice of pattern(s).
- Implementation details of the pattern, including sequence diagrams.
- A clear and convincing argument for your choice of pattern(s).
- Testing strategy and outcome.

Deliverables:

1. The application coded and deployed. The source code of your application must be committed to your Git repository. The application must be deployed in Heroku.
2. You must create a tag in your Git repository for this deliverable in the following format: SWEN90007_2022_Part3_<team name>. The tag must be created before the submission deadline as this will be used to assess your deliverable (no exceptions). After creating the release tag, you deploy this tag (not your main repository branch).
3. A Software Architecture Design (SAD) report with the following items:
    - The updated class diagram of your application, with any updates made to it highlighted.
    - A discussion on concurrency that includes for each concurrency issue identified: a description of the issue, the choice of pattern(s) and its implementation details including sequence diagram(s), the rationale for your choice of pattern(s), the testing strategy used to validate your concurrency approach, and a summary of the outcomes of the concurrency tests carried out.
    - A link to your Heroku deployed App including data/information that we need in order to be able to test your application.
    - This report must be available in your GitHub (docs/part3).
    - Part3 checklist (Appendix C – Checklist for Part3 Deliverable) must be submitted to Canvas assignments.
4. README.md updated to include the link to your Heroku deployed app
    a. Your deployed App must include a populated database with a range of realistic data samples/information that are necessary for the teaching team to test your application. You can create a document explaining your database, add it to docs/data-sample and add the information to your README.
5. README.md must also provide additional instructions on how to use the existing data that you created in your system, for example: the administrator username and password. We do

not expect to create any test data from scratch. Again, this is your project. Make sure you test your deliverable before your submission and that you have real/appropriate data in the deployed system. Meaningless data such as abc, 123, blah blah, subject ASDFG and so on will compromise the assessment of your project (and your final marks). You can create a document with data samples, add it to docs/data-sample and add the information to your README. See Appendix H – Test Cases and Data Entry Document for more details.

NOTE: There is not a single correct answer in terms of pattern choice. Patterns should be chosen based on the specific needs of the enterprise system (context) and justified accordingly. Discuss differences, pros and cons in regard to your specific project.

Part 4 [10 marks]

In this part, you will produce a report reflecting on the performance of your system.

In your report, you are required to provide a meaningful discussion of system performance including:

- How could your system have improved by using different design patterns? Give concrete examples and justifications
- A description of what design principles you used that affect performance in your application. For those that you haven't used, explain why and how they would affect the performance of your system.

Deliverables:

1. You must create a tag in your Git repository for this deliverable in the following format: SWEN90007_2022_Part4_<team name>. The tag must be created before the submission deadline as this will be used to assess your deliverable (no exceptions). After creating the release tag, you deploy this tag (not your main repository branch).
2. A report with the following items:
    - A discussion on performance.
    - This report must be available in your GitHub (docs/part4).
    - Part4 checklist (Appendix D – Checklist for Part4 Deliverable) must be submitted to Canvas assignments.

Submissions

Submission of checklists will be via the LMS, under the Assignments menu. Only one member of the team should submit checklist via the LMS.

For the source code and reports, we will assess your (GitHub) RELEASE TAGS for marking purposes. Make sure you generate the tags, deploy them and add them to your checklist before the submission deadline.

Following are the project deliverables (submissions) throughout the semester, and the marks associated with the different submissions (adding up to a total of 100 marks).

Deliverable	Marks	Due Week	Due Date
Part 1	15	3	Monday 15 Aug, 5pm

Part 2	40	9	Friday 23 Sep, 5pm
Demonstration Part 2		During Week 9	
Part 3	35	12	Friday 21 Oct, 5pm
Demonstration Part 3		During Week 12	
Part 4	10	First week of exams	Friday 4 Nov, 5pm

Feedback will be provided for each stage within two weeks of submission.

Extensions

Extensions will only be granted if students are able to provide appropriate supporting documentation. If an extension is needed you must contact the lecturer as soon as possible, before the submission date. It will not be possible to apply for an extension after the submission date.

For further details please see:
https://ask.unimelb.edu.au/app/answers/detail/a_id/5667/~/applying-for-an-extension

Late Submissions

Unless an extension is in place, a 10% penalty will be applied to every day you delay your submission (Checklist, RELEASE TAG and/or Deployment).

Example: Part 2 - Deadline September 23, 5PM

Scenario 1: For Part 2 deliverable, you can score 40 maximum. You and your team submitted the PDF report on Canvas before the deadline (the day before, for example). However, two days later you were having a coffee together and realised that you did not create a release tag to your source code. You finish your coffee quickly, go to GitHub and generate a release tag to your source code. Unfortunately, as that happened two days after the deadline, this means we will need to work with the calculator: 2 days x 10% = 20%. Your maximum possible mark for this deliverable is now 32.

Scenario 2: You and your team submitted the PDF report on Canvas before the deadline (the day before, for example). On Sunday afternoon (the BIG deadline day), you are watching the footy on TV and realise that you did not create a release tag to your source code. As it is still lunch time, you finish watching the game, celebrate your team's victory, go to GitHub and generate a release tag to your source code. Luckily, as much as the resources were submitted and generated in different days, everything happened before Part 2 deadline. No calculator needed for you and your team. Your maximum possible marks remain unchanged.

Assessment Criteria

The goal of this project is for you to apply and demonstrate a clear understanding of the design principles and architectural patterns covered in the subject. Your project will be assessed based on the following high-level criteria:

- Complete implementation of the functionality specified in the application domain.
- Clear, complete, and correct presentation of the application's design and architecture using UML.
- Correct implementation of the patterns.

- Design quality and rationale. This includes appropriate selection of patterns (if applicable), sound reasoning behind their selection, and accurate explanations of the design principles and patterns used.

Note that proper use of UML notation is expected. Inaccuracies in your UML diagrams will affect your mark. In particular, you are expected to produce class diagrams, use case diagrams, and sequence diagrams. Also, your implementation must match the design presented in the documents. Any discrepancies between code and documentation will impact your mark.

Please be aware that simply producing an application that provides the required functionality is not enough to pass the assessment.

Assessment breakdown

Part 1

Criterion	***Description***	***Marks***
Use case diagram	Correctly identified use cases and their relationships. Correct use of UML notation.	4
List of use cases	A simple description of each use case. Each use case must contain the list of actors associated with the use case, a high-level description of the use case.	4
Domain model diagram	Correct use of UML notation, appropriate level of detail and abstraction.	5
Domain model explanation	Description for the domain model	2
	Total	***15***

Part 2

Criterion	***Description***	***Marks***
Class diagram	Correct use of UML notation, appropriate level of detail and abstraction.	5
Pattern description	Correct, complete, and **contextualised** implementation details for each pattern, correct sequence diagrams with an appropriate level of detail.	15
Design rationale (UoW and Lazy Load)	A clear and convincing rationale for the use of unit of work and lazy load patterns in the design of the application.	5
Functionality and pattern implementation	A correct and complete implementation of the architectural design, including a correct implementation of the patterns and application functionality.	15
	Total	***40***

Part 3

Criterion	***Description***	***Marks***
Class diagram	Correct use of UML notation, appropriate level of detail and abstraction.	2
Concurrency issues and pattern description	Correct identification and description of the concurrency issues that may arise in the application.	10

	Correct, complete, and contextualised implementation details of the concurrency patterns for each concurrency issue. Correct sequence diagrams with an appropriate level of detail.	
Design rationale (concurrency pattern(s))	A clear and convincing rationale for the use of concurrency patterns for each concurrency issue.	8
Testing strategy and outcomes	A sound and robust concurrency testing strategy and a clear summary of the test outcomes.	5
Functionality and pattern implementation	A correct and complete implementation of the architectural design, including a correct implementation of the patterns and application functionality.	10
	Total	***35***

Part 4

Criterion	***Description***	***Marks***
Performance discussion	A meaningful discussion of system performance, including an explanation on patterns and principles you have used that affect performance, and a discussion on those that you haven't used and how they would affect the performance of your system.	10
	Total	***10***

FAQ

Available here:
https://www.notion.so/505fe268ba3241689001390087842 3d6?v=fd8a0ce7daab40d1b2478ec75887fa6d

Appendix A – Checklist for Part1 Deliverable

Checklist must be submitted to Canvas by only one team member

Team name:
Repository name:
Release tag:
URL to access your application:
Delivered:

	Use case diagram
	List of use cases
	Domain model diagram
	Domain model explanation/rationale

Additional comments (optional):

Appendix B – Checklist for Part2 Deliverable

Checklist must be submitted to Canvas by only one team member

Team name:
Repository name:
Release tag:
URL to access your application:
Delivered:

	List of use cases agreed with teaching team (Part 1 feedback)
	Class diagram
	Pattern descriptions
	Design rationale (UoW and Lazy Load)
	Functionality and pattern implementation

List of use cases implemented (as agreed with teaching team – Part 1 feedback). List them all – just use cases IDs.

Fully implemented and deployed patterns (list them all – just pattern names)

Additional comments (optional):

Appendix C – Checklist for Part3 Deliverable

Checklist must be submitted to Canvas by only one team member

Team name:
Repository name:
Release tag:
URL to access your application:
Delivered:

	Class diagram
	Concurrency issues and pattern description
	Design rationale (concurrency pattern(s))
	Testing strategy and outcomes
	Functionality and pattern implementation

Fully implemented and deployed patterns (list them all – just pattern names)

Additional comments (optional):

Appendix D – Checklist for Part4 Deliverable

Checklist must be submitted to Canvas by only one team member

Team name:
Repository name:
Release tag:
URL to access your application:
Delivered:

	Performance discussion

Additional comments (optional):

Appendix E – Simplified use cases

You can create new documents and/or templates to document your project. For your convenience, we provide a sample template on Canvas. You can modify and use that document as much as needed. Again, the use of sample templates we share on Canvas is not mandatory and you are welcomed to come up with your own ideas/documents/templates.

Template Available on Canvas:
https://canvas.lms.unimelb.edu.au/courses/130935/modules/items/3967463

Appendix F – Expanded use cases

You can create new documents and/or templates to document your project. For your convenience, we provide a sample template on Canvas. You can modify and use that document as much as needed. Again, the use of sample templates we share on Canvas is not mandatory and you are welcomed to come up with your own ideas/documents/templates.

Available on Canvas:
https://canvas.lms.unimelb.edu.au/courses/130935/modules/items/3967464

Appendix G – Architecture Document

You can create new documents and/or templates to document your project. For your convenience, we provide a sample template on Canvas. You can modify and use that document as much as needed. Again, the use of sample templates we share on Canvas is not mandatory and you are welcomed to come up with your own ideas/documents/templates.

Available on Canvas: https://canvas.lms.unimelb.edu.au/courses/130935/modules/items/3967465

Appendix H – Test Cases and Data Entry Document

You can create new documents and/or templates to document your project. For your convenience, we provide a sample template on Canvas. You can modify and use that document as much as needed. Again, the use of sample templates we share on Canvas is not mandatory and you are welcomed to come up with your own ideas/documents/templates.

Available on Canvas:
https://canvas.lms.unimelb.edu.au/courses/130935/modules/items/3967466