The University of Melbourne
School of Computing and Information Systems
SWEN90007 Software Design and Architecture

# Project Specification, Semester 2, 2019

## Project Description

In the project, you will design, build and evaluate some features of a prototype Enterprise Software System for a domain of your choosing.

The project aims to give students experience in small teams to:

- identify the functional and non-functional requirements for an enterprise application;
- produce an architectural design for an enterprise system;
- choose suitable design patterns to be applied for the different architectural layers;
- implement an executable architecture for your design; and
- experience the complete software development life cycle, while meeting specified milestones.

This will be a team project carried out in teams of **two students**. The teams will be self selected **(students must register and attend the same workshop)**, and you are expected to plan, coordinate and manage activities within the team.

### Application Domain

You are free to choose the application domain and the list of use cases you want to design and develop, as long as the chosen application domain has the characteristics of a typical enterprise application, such as *persistent data*, *big data*, *concurrent access*, *many user interface screens*, *users with different permissions* and *complex business logic* as per the subject notes. It is understood that you will be developing an executable architecture rather than a full scale enterprise application, however, it is expected that you would approach the problem as if it were a large-scale enterprise system.

### Architecture

The system that you design and develop must be a web-based system. The system will be developed using J2EE technology. The system will adopt the patterns taught during the lectures. The implementation will be done from scratch, no frameworks will be used to facilitate any infrastructure. Before beginning your work please read through all of the requirements and attempt to design an infrastructure that will be flexible and easy to build upon. Make sure to keep early versions of your work in your repository so you may reflect on the changes made as part of the last deliverable.

## Project Milestones and Deliverables

Following are the project deliverables (submissions) throughout the semester, and the marks associated with the different submissions (adding up to a total of 40 marks).

1. **Submission 1 [5 marks]:** Features A & B and use cases.

   **Due date:** Week 3 — 11:59pm, Sunday, 18 August 2019.

2. **Submission 2 [15 marks]:** Software Architecture Document (SAD) and executable architecture implementation for Feature A.

   **Due date:** Week 7 — 11:59pm, Sunday, 15 September 2019.

3. **Submission 3 [15 marks]:** Software Architecture Document (SAD) and executable architecture implementation for Feature B.

   **Due date**: Week 10 — 11:59pm, Sunday, 13 October 2019.

4. **Submission 4 [5 marks]:** Reflection, performance and maintainability report.

   **Due date**: Week 12 — 11:59pm, Sunday, 27 October 2019.

Feedback will be provided for each stage within two weeks of submission. It is the student team's responsibility to seek advice from their tutor to ensure that the proposed system is sufficiently complex. If after submissions it is assessed that the system lacks the required complexity, extra requirements will be added to your system when you receive feedback for deliverables.

## Project requirements

Below are the details for all parts of the project.

1. **Part 1 [5 marks]:** This submission must include the following:

   (a) Your team members and their University of Melbourne usernames (2 members per team)

   (b) An overview of the enterprise application chosen

   (c) Detailed use cases (including scenarios) for both features of your system (*Features A & B*), which must involve a user creating, reading, updating and deleting a data source (e.g. a database table).

   (d) A (private) Git repository created on `http://bitbucket.org` or `http://github.com`, shared with the subject staff: `eduardo.oliveira@unimelb.edu.au`, `maria.rodriguez@unimelb.edu.au` and `your tutor` . Please assign the staff both read and write access to the repository. This repository should be used for all source code and related configuration items throughout your project. Documentation should also be stored here.

**NOTE :** There should be at least two roles over the two use cases, with different requirements and permissions for each role. This can be achieved by Feature A being used by one role and Feature B by another, or by roles having different attributes for each feature. The features should permit concurrent *write* access to *at least* one shared data source; that is, the data source should be available in *both* features. Each use case should cover all four of the CRUD (create, read, update, delete) operations for a single feature.

2. **Part 2 [15 marks]:** As part of the implementation of Feature A, the following patterns will be implemented:

   - Domain model and service layer
   - Data mapper
   - Unit of work, identity map, lazy load
   - Identity field, foreign key mapping, association table mapping, embedded value, and one of the inheritance patterns

   1. Document: if any change, change it
   Class diagram should be readable, break into different layer: data source layer, domain model layer, has a diagram show how they connected
   2. Code: excutable, sourcecode
   3. Test

   The *Software Architecture Design* (SAD) must include the following:

   (a) A domain class diagram for your feature, specified in UML

   (b) The high level architecture for your feature – static view only – that presents all relevant architectural design patterns used

   (c) The complete architecture for your feature: class diagram, component diagram and interaction diagram

   (d) A description of all architectural design patterns used for Feature A, contextualized and contrasted with alternative patterns (if applicable)

   (e) An executable architecture (coded and deployed) for Feature A

3. **Part 3 [15 marks]:** As part of the implementation of Feature B, the following additional patterns will be implemented:

   - Choose one of the most appropriate session patterns
   - Choose one of the most appropriate patterns for input controller, and for view
   - Choose either a pessimistic off-line or optimistic on-line lock, *and* must implement an implicit lock
   - Implement both Data Transfer Object and Remote Façade
   - Implement all relevant security patterns

   This submission must include the following:

   (a) A domain class diagram for Features A & B, specified in UML

(b) The high level architecture for Features A & B – static view only – that presents all relevant architectural design patterns used

(c) The *new* and *updated* parts of the architecture for Feature B: class diagram, component diagram and interaction diagram

(d) A description of all architectural design patterns used for Feature B, contextualized and contrasted with alternative patterns (if applicable). Please justify your chosen design patterns with regards to your specific implementation

(e) An executable architecture (coded and deployed) for Features A & B

**NOTE:** There is not only one 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 cons with regards to your specific project. For item 3c, only include the *new and updated* parts of the architecture for Feature B. ALSO, if you need to implement a pattern as part of Feature A or Feature B deliverable that is not really necessary in the scope of your system (for learning purposes), please document that and communicate with your tutor.

4. **Part 4 [5 marks]:**

   For the last deliverable please provide the following:

   - Meaningful discussion of system performance. How could your system have improved by using different design patterns? Give concrete examples and justifications
   - A description of what principles you used to affect performance in your prototype *and* justification for each of these. For those that you haven't used, explain why and how they would affect the performance of your system
   - Reflect on how the implementation of feature B impacted your architecture design and how could that be avoided in future (including explicit examples of the impact of future feature development). Reflect on your future professional practices

## Criteria

Submissions will be marked on the quality of the system and related artifacts, with a focus on the architectural design. Part of the marks will go towards the detailed design, the code, etc.

**Part 1**

| **Criterion** | **Description** | **Marks** |
|---|---|---|
| Overview | A clear overview of a system that has properties consistent with the definition of an enterprise system | 1 marks |
| Features | A clear, correct, complete, and consistent description of two features (A & B) exhibiting the required properties | 1 marks |
| Use cases | Clear, correct, complete, and consistent use cases describing Features A and B | 3 marks |
| Total | | 5 marks |

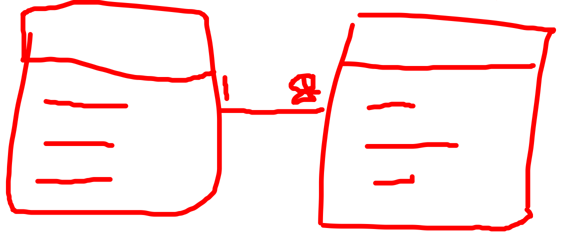
**Part 2**

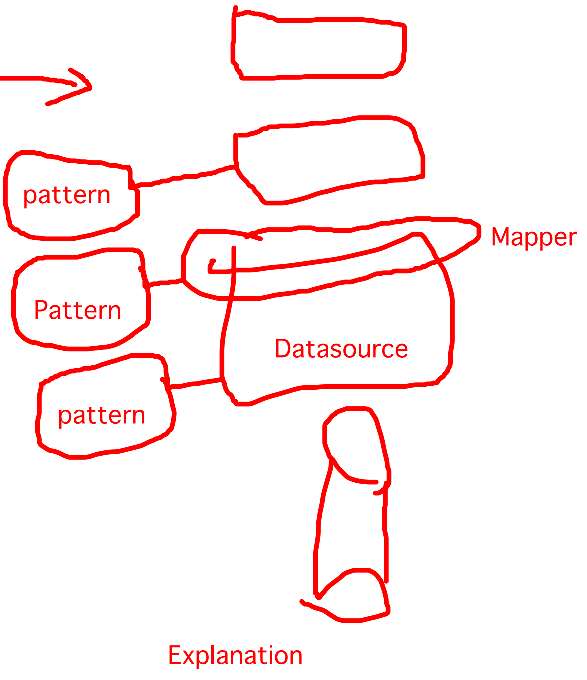
| **Criterion** | **Description** | **Marks** |
|---|---|---|
| Domain diagram | A clear, correct, complete, and consistent domain diagram for Feature A | 1 marks |
| High-level model | A clear diagram outlining the architectural patterns and components for Feature A | 1 marks |
| Detailed architecture | Correct, complete, and consistent structural and interaction views of the architecture for Feature A | 3 marks |
| Pattern discussion | A clear and convincing discussion for all implemented architectural design patterns contextualised and contrasted with alternative patterns | 2 marks |
| Executable architecture | An executable architecture (coded and deployed) for the architectural design for Feature A | 8 marks |
| Total | | 15 marks |

1. Overview
2. High-level architecture
3. Feature description details for feature A, how to implement, anyone can implement when seeing this.

*Database diagram

Description: for each pattern, primary key, foreign key...





4. Component diagram, have packages, interface between packages, description, half to 4 pages
5. Class diagram, for whole system, put every single thing for feature A, must be readable.
if big, cut it. Eg, domain, source code, explain the link between them.
Explanation and description.
6. Activity diagram, or sequence diagram.

**Part 3**

| **Criterion** | **Description** | **Marks** |
|---|---|---|
| Domain diagram | A clear, correct, complete, and consistent domain diagram for Features A & B | 1 mark |
| High-level model | A clear diagram outlining the architectural patterns and components for Features A & B | 1 mark |
| Detailed architecture | Correct, complete, and consistent structural and interaction views of the architecture for Feature B | 3 marks |
| Rationale | A clear and convincing rationale for the choice of the architectural design patterns used for Feature B | 3 marks |
| Executable architecture | An executable architecture (coded and deployed) for the architectural design for Features A & B | 7 marks |
| Total | | 15 marks |

**Part 4**

| **Criterion** | **Description** | **Marks** |
|---|---|---|
| Report | A meaningful discussion of system performance, considering alternative patterns for the complete architecture including justification of chosen and discarded patterns | 2 mark |
| Reflection | Reflective report on evolution of your design architecture throughout the project. Consider the maintainability of your system (including explicit examples of the impact of future development) and future professional practices | 3 marks |
| Total | | 5 marks |

## Submission

Submission of reports (feature definition, use cases, SADs, and reflective report) will be via the LMS, under the *Assessment* menu. All reports must be submitted as a PDF file. Only one member of your pair should submit reports via the LMS.

For the source code, repositories on BitBucket or GitHub must be shared with the subject staff, as outlined in the Project Submissions section. Further details on the deployment details of the executable architecture (for parts 2 and 3) will be provided at a later stage.

## Executable Architecture

You produced (or you will produce) an Executable Architecture (EA) as part of Sprint 1 in SWEN90014: Masters Software Engineering Project. To remind you of what an EA is, we will consider the example of an enterprise system which allows users to shop online for books and other products, similar in style to Amazon. To simplify the discussion, we will only consider that the system offers the features below. **Please, note that the example below is not exhaustive, there will be other patterns that you will need to cover, such as messaging, concurrency, distribution, etc.**

An (incomplete) example of a system:

- Any visitor to the website can browse products by category or search specific items using keywords
- Logged in customers can add products to their shopping cart
- Logged in customers can check out: indicate shipping address, make a payment, and get the products shipped to them
- Logged in administrators can perform CRUD operations on stocked items

An EA for this system will demonstrate that all architectural requirements are supported by your implementation without implementing the complete features. For instance, if we consider that important architectural requirements of the online shop are:

- The EA will support browsing data (the UI will show *only one* category). By clicking on that category, the EA will retrieve *dummy* data classified under that category. Dummy data refers to fake, non realistic data that can act as a place holder
  - For instance the GUI for a non-logged in user shows a category called ”Books”. By clicking on it, the GUI will show the items ”Book1”, ”Book2” and ”Book3”
- The EA will support finding items using a search engine
  - For instance, the GUI for a non-logged in user shows a text box and a ”Search” button. By typing ”2” and clicking on the ”Search” button, the GUI will show the items ”Book2”
- The EA will support logged in admins to perform CRUD operations on dummy data
  - For instance when a user with admin privileges logs in to the system, the GUI shows a listing of all products in the system (”Book1”, ”Book2” and ”Book3”). There are 3 buttons, ”Add”, ”Delete” and ”Edit”. The buttons do the expected actions on the selected item. Changes will be preserved in the DB
- If a logged in customer, logs out and logs in again, the items that were added to the shopping cart will still be there
- There will be a interface to send ”dummy” data to an external service (e.g. the shipping company, bank). This will be used to illustrate the transferring of data between your program

and an external service so a simple transaction will suffice. The external service can be a stub

  - For instance, from the shopping cart page, there will be a button that sends the value typed in a text box (e.g. ”I want this [Book1]”) to the shipping company. The service receives the value and sends it back, slightly altered (”There goes the [Book1]”), to be printed back by the GUI in a different text box