



**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

SC/CE/CZ2002: Object-Oriented Design & Programming

ASSIGNMENT

Build-To-Order (BTO) Management System

Lab/Tutorial Group: FCS2 (Group 03)

Group Members:

Name	Student ID
Lee Zhe Chian	U2421027L
Khurtsbilguun Onolt	U2420410F
Kunal Harsh	U2423502G
Lee Loong Kiat (Li Longjie)	U2420557J
Leck Kye-Cin	U2423244A

Table of Contents

Table of Contents.....	1
Chapter 1: Requirement Analysis & Feature Selection.....	2
1.1 Understanding the Problem and Requirements.....	2
1.2 Deciding on Features and Scope.....	2
Chapter 2: System Architecture & Structural Planning.....	3
2.1 Planning the System Structure.....	3
2.2 Reflection on Design Trade-offs.....	4
Chapter 3: Object-Oriented Design.....	4
3.1 Class Diagram.....	4
3.2 Sequence Diagrams.....	5
3.3 Application of OOD Principles (SOLID).....	6
3.3.1 Single Responsibility Principle (SRP).....	6
3.3.2 Open/Closed Principle (OCP).....	6
3.3.3 Liskov Substitution Principle (LSP).....	6
3.3.4 Interface Segregation Principle (ISP).....	7
3.3.5 Dependency Injection Principle (DIP).....	7
Chapter 4: Implementation.....	7
4.1 Tools Used.....	7
4.2 Sample Code Snippets.....	8
Chapter 5: Testing.....	9
5.1 Test Strategy.....	9
5.2 Test Case Table.....	9
Chapter 6: Documentation.....	13
6.1 Developer Guide:.....	13
Chapter 7: Reflection & Challenges.....	13
Appendix A.....	15
Declaration of Original Work for SC2002 Assignment.....	15

Chapter 1: Requirement Analysis & Feature Selection

1.1 Understanding the Problem and Requirements

The program is designed to streamline the BTO application process. From the start, we faced significant challenges due to the layered restrictions and overlapping responsibilities of each user role, which added complexity to both the design and implementation. Additionally, the project constraints limited us to using only pure Java and CSV files, further complicating development.

1.2 Deciding on Features and Scope

Based on the project guidelines, we outlined the following as essential features of the program:

- (i) Authentication functionalities:** Different users would be able to login using their NRIC and password.
- (ii) Applicant functionalities:** Able to view the list of projects open to their user group if visibility is “on”, Able to apply for a project and view application status, Able to book a unit if the application is successful through an Officer, Able to submit, edit or delete enquiries, Able to withdraw application
- (iii) HDB Officer functionalities:** Able to register to join a project team (based on restrictions), Able to view team application, Able to reply enquiries related to assigned project, Able to make a booking for applicants and generate booking receipt.
- (iv) HDB Manager functionalities:** Able to create, edit and delete BTO Projects, Able to approve/reject HDB Officer’s team application, Able to view all enquiries and reply to them across each project, Able to view all projects.

Understanding these essential features enabled us to start planning for our project. We focused on the foundational features like Authentication first to get a feel of designing and implementing OODP systems. From there, we slowly added on the other features.

Chapter 2: System Architecture & Structural Planning

2.1 Planning the System Structure

We began by analyzing the project requirements and identifying key user roles: Applicant, HDB Officer, and HDB Manager. From there, using SRP we separated them into the following logical components:

- **UI Layer:** Handles user interaction for each role.
- **Controller Layer:** Acts as a controller between UI and the system's logic.
- **Repository Layer:** Handles data storage, retrieval, and management.
- **Entity/Base Classes:** Represent domain objects such as User, Flat, Application etc.

Modeling User Flows and Use Case Mapping

Before finalizing the class diagram, we created rough use case diagrams to understand how each user would interact with the system.

For example: Applicants can register, log in, and apply for a BTO flat, HDB Officer can review and process applications and HDB Manager can manage officers and flats. Each use case was mapped to potential class responsibilities. For instance, the "Apply for Flat" use case led to the creation of an ApplicationController, ApplicationRepository, and Application entity class. We made sure each use case had a clear flow from UI to controller to repository/entity and back.

Creating Early Visual Models

To ensure clarity in logic, we created sequence diagrams for key processes. One example is the “Flat Application” process:

1. Applicant logs in.
2. System retrieves available flats.
3. Applicant selects a flat and submits application.
4. System validates application.
5. Application is stored and marked as “Pending”.

The sequence diagram helped us visualize the flow of interactions and ensure the correct order of method calls, also allowing us to consider potential edge cases (e.g., flat availability).

2.2 Reflection on Design Trade-offs

(i) Role-Based Class Structure vs Shared Logic:

- Option 1: Single User class with role flags (simpler but messy conditions).
- Option 2: Dedicated classes for each role.
- Chosen: Dedicated classes for each role with shared User abstract class.
- Trade-off: More files and some duplicate methods. However, cleaner and more organized code, and safe maintenance (e.g. changing HdbManager won't affect logic for Applicant). Also, better testing and team Scalability.

(ii) Layer Duplication (Controller/UI/Repo for each class):

- Option 1: Unified class.
- Option 2: Separate classes for each layer (Controller/UI/Repo).
- Chosen: Separate classes for each layer.
- Trade-off: Some duplications, but organized, isolated changes and clearer ownership.

Chapter 3: Object-Oriented Design

3.1 Class Diagram

Based on the project description, we identified the main classes by modeling them after real-world entities. For example, people—applicant, officer, manager and objects—BTO project, Applications, reports. By segmenting these core components, we were able to clearly visualize and design the class diagram. The following describes each main component of the class diagram:

(i) User component: Interface to model applicants, officers and managers components after. Defines common behaviour and properties like `LoadFile()` and `Login()`.

(ii) Applicant Component: Represents users who apply for BTO flats. Main responsibilities include submitting new applications, viewing available BTO projects, monitoring the status of their applications, and handling enquiries related to their applications.

(iii) Officer Component: Represents HDB officers responsible for administrative oversight. Responsibilities include reviewing and approving applications, addressing applicant enquiries and flat-booking.

(iv) Manager Component: Represents HDB managers who oversee the entire BTO project lifecycle. Responsibilities include creating and managing BTO project listings, assigning officers to projects and generating comprehensive reports.

(v) Applications Component: Manages and processes both team and residential applications. This component includes functionality for application submission, status tracking, validation checks, and retrieval.

- (vi) Project Component:** Manages details specific to individual BTO projects, including project information such as available flat types, unit counts, application deadlines, visibility status, and associated administrative assignments.
- (vii) Enquiry Component:** Facilitates the submission, tracking, and resolution of enquiries raised by applicants regarding their applications or general BTO project information.
- (viii) Report Component:** Similar to enquiry, facilitates report creation, tracking etc.

The different components work together via their controllers, transferring logic flow from one to one another depending on the user's input. Within each component, the controller also serves to dictate the logic flow to the respective UI/Repo/Entity class. Hence, the associative relationship.

Hdb Officer extends (inheritance relationship) from Applicant as it allows us to utilise polymorphism, when implementing the residential application feature for officers, as they behave as an Applicant under those circumstances.

3.2 Sequence Diagrams

We carefully selected use cases that represent the **core and more complex functionalities** of our BTO Management System. These scenarios involve multiple components—such as the Boundary (UI), Controller, and Entity (Base and Repository) — and demonstrate how these classes interact to fulfill essential features like applying for a BTO and subsequently booking a flat.

(i) Hdb Officer helping to book flat for successful application.

This use case highlights the responsibility of a Hdb Officer, which is to help successful applicants with the booking of flat. It involves **user authentication**, **data retrieval** of Applicant's flat application and **data update** (e.g. changing Applicant's application status to **BOOKED**). The flow includes **conditional logic** - can only help to book flat if the HdbOfficer is assigned to the project and can only apply flat for applicants who have their application approved by the HdbManager in-charge of the project (application status is **SUCCESSFUL**).

(ii) HDB Officer applies for a BTO and register to handle a project.

This use case highlights both the basic requirements of the project, as well as represents a core function of the system where the Hdb Officer applies for a BTO and registers to handle a different project. This use case also involves **user authentication**, **data retrieval** of projects (and its details) and showcases the **interaction** between the involved classes. There is conditional logic in the sequence diagram, like checking for Hdb officer's eligibility in applying for the project (whether he is already assigned to handle the same project, as well as age and marital status checks) and a reference fragment that reuses the sequence diagram for Hdb Officer helping to book flat for successful application. In the same sequence diagram, after the officer's application for a BTO, it showcases the flow of the officer registering to handle a project.

3.3 Application of OOD Principles (SOLID)

3.3.1 Single Responsibility Principle (SRP)

Single Responsibility Principle (SRP) states that each class should only be made for one specific function. In our design, we adhered to SRP by implementing Entity, Control, Boundary (ECB) design pattern. This allows us to segment code into different classes, each with their own respective functions.

- Entity classes - base classes (e.g., Applicant, HdbOfficer, HdbManager, etc) are responsible for storing essential attributes of their respective entities (e.g., name, applicantID, password, etc) as well as repository classes (e.g, ApplicantRepo, HdbOfficerRepo, ProjectRepo, etc) are responsible for loading and saving the program's data into csv files. It also serves as a database during run-time to access objects.
- Boundary classes (e.g, ApplicantUI, ProjectUI, TeamAppicationUI, etc) are responsible for displaying information to the user and taking in their input. It interacts directly with its controller based on the user's input.
- Controller classes (e.g, ApplicantController, ProjectController, EnquiryController, etc) are responsible for the interactions between entities. It dictates the logic flow of the program by interacting with other controllers.

3.3.2 Open/Closed Principle (OCP)

Open/Closed Principle (OCP) states that software entities should be open for extension, but closed for modification. We adhered to this principle in our program by relying on a Flat Interface to contain information of a project's flat types. TwoRoom and ThreeRoom classes implement Flat Interface and hold information of their respective project's details like number of units and selling price. This implementation allows future developers to extend the program by adding more flat types (e.g, 4-Room, 5-Room, etc) easily. Only their new concrete class that implements the Flat interface has to be created, and only a minimal amount of code needs to be modified in the program.

3.3.3 Liskov Substitution Principle (LSP)

Liskov Substitution Principle (LSP) states that objects of a superclass shall be replaceable with objects of its subclasses without breaking the application.

In our program, we utilized polymorphism in the ResidentialApplication component of the program to allow HdbOfficers (which extends Applicant) to utilize methods meant for Applicants. This design aligns well with LSP.

3.3.4 Interface Segregation Principle (ISP)

Interface Segregation Principle (ISP) states that many specific interfaces are better than one general purpose interface. This principle allows clients to only rely on interfaces that are relevant to their functions, hence not forcing them to implement irrelevant methods.

In our program, we adhered to this principle by segregating our ApplicationController interface into two subinterfaces—ResidentialApplicationController and TeamApplicationController. This prevents TeamApplicationController from implementing methods like `displayApplicationMenu(Applicant applicant)`

which it does not use because applicants do not need access to a team application menu.

3.3.5 Dependency Injection Principle (DIP)

Dependency Injection Principle (DIP) suggests that high-level modules should not depend on low-level modules, but both should depend on abstractions. To adhere with DIP in our program, we inject controller dependencies via the constructors rather than instantiate them directly. This can be seen in our MainMenu class, where we inject ProjectControllerInterface into the controllers like ApplicantController which depends on ProjectController methods. This serves to decouple logic, and enable easy modification as we can easily inject a new ProjectController implementation from MainMenu.

Chapter 4: Implementation

4.1 Tools Used

We used the following tools for our project:

(i) Java 24:

Java24 provided an object-oriented programming language for us to showcase OODP. Java24 also allowed us to utilise modern java enhancements like generics and streams in our code.

(ii) draw.io:

draw.io provided us a platform to collaborate and craft our UML class and sequential diagrams. This enabled us to plan efficiently at the start of the project.

(iii) Github:

Github provided a platform for us to collaborate and merge our code. Its version-saving features also allowed us to debug and make changes easily.

4.2 Sample Code Snippets

The snippets from our code are as following displaying the following features:

(i) Encapsulation

Example below shows Encapsulation, where all attributes of User are private and can only be accessed through setters and getters.

```
public class User {  
    private final String name;  
    private final String nric;  
    private int age;  
    private MaritalStatus maritalStatus;  
    private String password;  
    ...  
}
```

(ii) Inheritance

Example below shows Inheritance, where HdbOfficer class inherits Applicant class.

```
public class HdbOfficer extends Applicant{...}
```

(iii) Polymorphism

Example below shows Polymorphism, where a TwoRoom or ThreeRoom object can be passed through displayFlatDetails method as they are both instances of Flat.

```
public void displayFlatDetails(Flat flat){  
    int totalUnits = flat.getTotalUnits();  
    int availableUnits = flat.getAvailableUnits();  
    int price = flat.getSellingPrice();  
    InputUtils.printSmallDivider();  
    System.out.println(flat.typeToString() + "-Room units left: " + availableUnits + "/" + totalUnits);  
    System.out.println(flat.typeToString() + "-Room selling price: $" + price);  
}
```

(iv) Interface use

Example below shows DIP, where Applicant Controller depends on the interface of Project Controller.

```
public class ProjectController implements ProjectControllerInterface {...}  
  
public class ApplicantController implements UserController{  
    //attributes  
    private final ProjectControllerInterface projectController;...  
    //constructor  
    public ApplicantController(...,ProjectControllerInterface projectController,...) {  
        this.projectController = projectController;...  
    }  
}
```

(v) Error handling

Example below shows how we handle `NumberFormatException` when reading in a user input for choice.

```
public static int readInt() {  
    while (true) {  
        try {  
            return Integer.parseInt(SCANNER.nextLine().trim());  
        } catch (NumberFormatException e) {  
            System.out.print("Invalid input! Please enter a number: ");  
        }  
    }  
}
```

Chapter 5: Testing

5.1 Test Strategy

The testing process was divided into two main sections. The first focused on authentication, where standard inputs were tested to verify that valid credentials correctly grant users access to the application. Exceptional inputs, which we defined as invalid user inputs were also tested to ensure appropriate error handling. The second part of testing centered around the user experience. This involved decomposing the program into its three main user roles (Applicant, HDB Officer, and HDB Manager) and testing each role individually. The goal was to ensure that each user type could navigate and use the system smoothly without encountering functional issues.

To facilitate testing, we reset all passwords to “password” and used standardized NRIC formats (S1234XXXXF) for our dummy users. The first digit following 1234 indicated the user type (1xxF for Applicants, 2xxF for Officers, and 3xxF for Managers). This allowed us to test the program without having to memorise or refer to our csv files for the details of each dummy user.

We also manually edited each csv file, creating enough dummy users with relevant details to provide an environment suitable for testing each feature of the application.

5.2 Test Case Table

5.2.1 Login System and Password Management

Test Case	Category	Input Data	Description of Expected Results	Actual Output
NRIC	Normal	S1234104F	User should be prompted for a password if the NRIC exists in the database to continue using the app	Enter password:
	Normal	s1234104f	Case-insensitive on both ends allows user to continue using the application without needing to capitalise their NRIC	Enter password:
	Exceptional	S123456A	Not 7 digits. An error should be printed.	Invalid NRIC format!
	Exceptional	S1234567	Missing last letter. An error should be printed.	Invalid NRIC format!

	Exceptional	""	Empty string. An error should be printed.	Invalid NRIC format!
	Exceptional	S 1234567 A	Spaces included. An error should be printed.	Invalid NRIC format!
Password	Normal	password	Correct password (default) should lead to dashboard showing successful login message	Login successful: Welcome, Applicant James!
	Exceptional	""	Empty string	Incorrect password
	Exceptional	PassWORD	Mixed case	Incorrect password
Change Password	Normal	passNew_123	The new password should allow the user to access the dashboard.	Enter NRIC: S1234105F Enter Password: passNew_123 Login successful: Welcome, Applicant Rachel!
	Exceptional	password	Old password should no longer work as only there should only be a single password that leads to successful authentication	Enter NRIC: S1234105F Enter Password: password Incorrect password

5.2.2 Applicant Actions

Test Case	Input Conditions	Test Objective	Actual Output
Apply for project	User selects option 1 to apply for a project, successfully applies, then selects option 1 again to apply for a different project	Applicants are unable to apply for more than one project. Applying for additional projects will result in an error message	You have an active application. You cannot apply to more than 1 project.
View Application Status	Upon entering the dashboard, user selects option 1 to view the application	User should see the application status	Applied to: West BrickVille Flat type: TWOROOM Application Status: PENDING
Make Booking	User has a PENDING application and selects option 3 to make a booking	Before the application has been approved, attempts to make a booking should raise an error message.	You cannot make a booking yet. Please wait until your application is successful.
Submit Withdrawal Request	From the application menu, user selects option 2 to withdraw their application, confirms the withdrawal by selecting option 1 (YES), and submits the request successfully	Application status should be updated	Applied to: West BrickVille Flat type: TWOROOM Application Status: WITHDRAWING
Perform CRUD operations on Enquiries	User selects option 3 to access the Enquiry menu. Within this menu, options 1–4 are used to perform Create, Read, Update, and Delete operations. The user performs CRUD operations on multiple enquiries during the session	Verify that the system correctly allows the user to perform all CRUD operations on multiple enquiries within a single session, and that changes are accurately reflected in the enquiry list	----- Your Enquiries ----- [PENDING] Booking date for West BrickVille When does booking open for West BrickVille? EN000005 ----- [PENDING] availability of 4-room flats hi, are 4 room flats available in West BrickVille? thanks! EN000004 -----
Project Visibility Based on User Group	A 35 year old single user selects option 2 to view 3-room projects	Project that is not open to user group should not be visible, providing an error message instead	You are not eligible to apply for 3-Room flats.

5.2.3 Officer Actions

View and respond to	From the assigned project menu, officer inputs 2 to “View enquiries to this project”.	This should lead to the enquiry menu for the officer	===== Enquiry Menu ===== 1. Respond to Enquiry
---------------------	---	--	---

enquiry		to respond to enquiries	2. Back Enter choice:
Receipt Generation for Flat Booking	Officer inputs 2 to generate a receipt of the applicants with their respective flat booking details, which are printed to the console	Accurate and complete receipts are generated and should include the following information: Applicant's Name, NRIC, age, marital status, flat type booked, project details.	Applicant's name: James Applicant's NRIC: S1234104F Applicant's age: 30 Applicant's marital status: MARRIED Flat type booked: TWOROOM ----- Name: Acacia Breeze Neighbourhood: Yishun Applications open from: 10/02/2025 to 20/07/2025 (project details truncated for brevity)
Project Detail Access for HDB Officer	1. Residential Application Menu 2. Team Application Menu 3. Assigned Project Menu 4. Change password 5. Exit Enter your choice (1-5): 3	Test whether the officer can access full project details, when the visibility is turned off.	Name: Acacia Breeze Neighbourhood: Yishun Applications open from: 10/02/2025 to 20/07/2025 ----- 2-Room units left: 2/2 2-Room selling price: \$350000 ----- 3-Room units left: 3/3 3-Room selling price: \$450000 ----- Manager-in-Charge: Michael (MA-301F) ----- Assigned Officers: - Daniel (OF-201F) - Emily (OF-203F) ----- Pending Officers: None
HDB Officer Registration Eligibility	An ineligible officer selects 2 to make a registration attempt	Test whether the system disallows it, status should be set to pending instead of approved	OFFICER DASHBOARD ----- Name: Preston Marital status: SINGLE Age: 35 ----- ----- 1. Residential Application Menu 2. Team Application Menu 3. Assigned Project Menu 4. Change password 5. Exit Enter your choice (1-5): 2 Applied to : West BrickVille Team Status : PENDING

5.2.4 Manager Actions

Able to access created projects	Manager inputs 1 to create a project and selects 4 to view the project	Test whether the created project is saved in the system	Name: Example project Neighbourhood: Nanyang Technological University Applications open from: 25/04/2025 to 26/04/2025
Single Project Management per Application Period	In trying to manage another project, manager inputs dates whereby manager is already handling a project	Test if system prevents assignment of more than one project to a manager within the same application dates	The manager can not manage the project at the time!
Respond to enquiries	Manager inputs the enquiry ID to select an enquiry to respond to, which is stored as a string	Responses on manager portal should update applicant portal as well	(Manager portal) Enter enquiry ID to respond: EN000004 Enter your response: Answering the test enquiry Response submitted successfully! (Applicant portal) ----- Your Enquiries ----- [ANSWERED] Test Enquiry Testing Enquiry Response: Answering the test enquiry EN000004

View All and Filtered Project Listings	Please choose an option:----- 1. View 2-Room projects2. View 3-Room projects 3. Back Enter your choice (1-3): 1	Test whether the relevant project type selected by the manager can be viewed (e.g. 2-room in this example)	TWOROOM Project (1/5):----- ----- Name: West BrickVille Neighbourhood: Chua Chu KangApplications open from: 10/01/2025 to 20/06/2025 ----- ----- 2-Room units left: 3/32-Room selling price: \$340000 ----- Please choose an option:----- 1. Next project2. Previous project 3. Exit to menuEnter your choice (1-3): 1 ----- ----- TWOROOM Project (2/5):----- ----- (output is truncated for brevity)
Approve Applicant's BTO Application	===== Residential Application Menu ===== Please choose an option: 1. View applications 2. Approve application 3. Reject application 4. Approve withdrawal 5. Exit Enter your choice (1-5): 2	There should be different outputs for the manager portal and applicant portal.	(Manager Portal) Enter the applicant ID: AP-104F Successfully approved! (Applicant Portal) Applied to: West BrickVille Flat type: TWOROOM Application Status: SUCCESSFUL
Reject Applicant's BTO Application	From the residential application menu,manager inputs 3 and receives a confirmation message upon successfully rejecting an application.	Test whether rejection by HDBManager correctly updates applicant portal.	(Manager Portal) Enter the applicant ID: AP-102F Successfully rejected! (Applicant Portal) Applied to: Acacia Breeze Flat type: TWOROOM Application Status: UNSUCCESSFUL
Approve Withdrawal	From the residential application menu,manager inputs 4 and receives a confirmation message upon successfully approving a withdrawal.	Test whether withdrawn status is updated in applicant portal as well	(Manager portal) Enter the applicant ID: AP-102F Successfully approved! (Applicant portal) Applied to: Acacia Breeze Flat type: TWOROOM Application Status: WITHDRAWN
Generate and Filter Reports	From the report menu, manager is able to select 2 to generate a report. Generated reports can be viewed by selecting 1 and a filter can be applied to selectively view relevant reports. For example, input 3 filters by marital status.	Test whether generated reports have the necessary information and are displayed selectively, depending on filter	----- Reports with Marital Status: SINGLE ----- Report ID: RE000001 Project Name: Acacia Breeze Applicant ID: AP-101F Flat Type: TWOROOM Applicant Age: 35 Marital Status: SINGLE

Chapter 6: Documentation

6.1 Developer Guide:

Prerequisites

1. Java (Version 17 or higher)
2. IDE (Eclipse, IntelliJ, etc.)

Steps to run the program:

1. Clone the repository: <https://github.com/onoltde/SC2002-Group-3/tree/main>
2. Open the project and run in your IDE or compile and run using command line

Chapter 7: Reflection & Challenges

What Went Well:

Throughout this project, our team encountered numerous challenges but showed remarkable adaptability and rapid learning despite being new to working in a team to develop an application. We conducted weekly recurring meetings to coordinate development efforts, align on responsibilities, and make joint decisions. These meetings helped us resolve ambiguities and ensured smooth collaboration, keeping everyone on track. Considering the project's complexity and tight timeframe, we were still able to deliver a functioning application. This stands as a testament to our collective effort, determination, and capacity to quickly assimilate new knowledge. Collaboratively, we gained deeper insights into applying design principles and learned practical skills like team coordination and technical skills like using git.

What Could Be Improved:

One significant area for improvement was our initial team communication and planning. Early in the project, inadequate communication delayed task allocation and workload distribution, ultimately impacting our efficiency. In future projects, we would place greater emphasis on comprehensive initial planning, clearly defining roles and responsibilities at the start of the project.

Furthermore, our application of design principles could have been more consistent and thoroughly integrated throughout our development process. While we made efforts to implement these principles, our limited experience occasionally led to fragmented application and caused us to revise certain aspects of the design. Future iterations would benefit from more structured and deliberate integration of design patterns and principles, which would enhance maintainability and extensibility.

Finally, the user-friendliness of our application did not reach our desired standards, largely due to the constraints of limited experience. Given more time and a clearer roadmap from the beginning, we would

prioritize user interface improvements, even within the limitations of a command-line interface, to significantly enhance the user experience.

Overall, this project gave us a valuable opportunity to apply OODP concepts in a real-world context. It challenged us to make thoughtful design decisions and to structure our code in a way that emphasized separation and reusability. We also came to appreciate UML diagrams and planning as it proved a major help in guiding development and collaboration.

Individual Contributions:

Name	Contributions
Lee Zhe Chian	Lead Developer, Class Diagram Designer
Khurtsbilguun Onolt	Feature Developer
Kunal Harsh	Documentation Lead, Feature Developer
Lee Loong Kiat (Li Longjie)	Feature Developer, Sequence Diagram Designer
Leck Kye-Cin	Lead Tester, Feature Developer


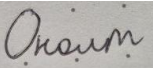



Appendix A

Declaration of Original Work for SC2002 Assignment

We hereby declare that the attached group assignment has been researched, undertaken, completed, and submitted as a collective effort by the group members listed below.

We have honored the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work.

We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work. In addition, disciplinary actions may be taken.

Name	Course	Lab Group	Signature/Date
Lee Zhe Chian	CSC	FCS2	 24/4/2025
Khurtsbilguun Onolt	CSC	FCS2	 24/04/2025
Kunal Harsh	CSC	FCS2	 Date: 24/04/2025
Lee Loong Kiat (Li Longjie)	CSC	FCS2	 24/4/2025
Leck Kye-Cin	CE	FCS2	 24/04/2025

Important notes:

1. Name must **EXACTLY MATCH** the one printed on your Matriculation Card.
2. Student Code of Academic Conduct includes the latest guidelines on usage of Generative AI and any other guidelines as released by NTU.