NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

SC2002 Object Oriented Design and Programming

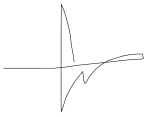
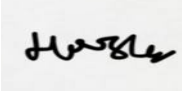
AY 24/25 Semester 2 | Group Project Report | Lab Group: FDDA | Group: 1

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	Matriculation No.	Signature	Date
Juirnarongrit Nathan	U2321878E		23 April 2025
Kongkijpipat Peeranat	U2323360A		
Nur Farhana Binte Mohammad Salleh	U2230582E		
Sai Harsha Venugopal	U2321873F		
Sathwiik Sai Krishna Kumar	U2420780E		
Sean Kong Kai Jie	U2130632E		

1. Requirement Analysis & Feature Selection

1.1 Understanding the Problem and Requirements

The application is a BTO flat booking system. A login page authenticates users and redirects them with features like password encryption and password changes. Applicants can view, filter, and apply for projects, track their application status, request withdrawals, and make enquiries. Officers can register for a separate project (excluding their applied ones), access project details, respond to enquiries, book flats for applicants, and generate receipts. Managers can manage one project per application period, perform CRUD operations on projects, approve applications and officer registrations, respond to enquiries, and generate reports.

1.2 Deciding on Features and Scope

Features that are explicit requirements for the BTO application system and to maintain the core capability of the application process are deemed to be core features. Optional features include enhancements that could improve the user experience but don't impact the operation of the booking system, including user's expectations. These may be skipped if time is limited or implementation is too complex. Lastly, features are also excluded if they are peripheral to the main goals of the system due to similar reasons.

Table 1. Comprehensive list of all features with their categorisation

Page	Categorisation	Feature
Login	Core	User login and authentication
		Redirecting to role-specific pages
		Password encryption
	Optional	Change password
		Retry if wrong credentials
	Excluded	"Remember Me" functionality
Applicant	Core	View applicable projects
		Apply for and withdraw from project + view application status
		Book flat
		Make, edit and delete queries
	Optional	View profile
	Excluded	Application status notification
Officer	Core	Register for project + view registration status and project detail

		Respond to applicant queries
		Schedule flat booking
		Generate booking receipts by applicant
	Optional	Generate booking receipts by project
	Excluded	Project analytics dashboard
Manager	Core	Manage projects
		Approve or reject officer registration and applicants applications
		Generate reports
		View all enquiries and respond enquiries for projects they are working on
	Optional	Change applicant application status, not only approving or rejecting
	Excluded	Audit trail of actions

2. System Architecture & Structural Planning

2.1 Planning the System Structure

We broke the system down into the 3 main class stereotypes, entity, control and boundary. We first described each user's journey, allowing us to identify all the nouns needed to translate into the entities for our design. We then grouped the entities together to identify base classes to utilise polymorphism, followed by identifying all actions to translate to methods in our design. We grouped similar methods together to create a single control class to execute these methods. Lastly, the boundary classes serve as user interfaces for the user and control classes to interact. A short excerpt of a user journey description demonstrating the process is as follows where the bolded words translate to an entity where the underlined words translate to a method.

"An **applicant** views all applicable **projects** and can apply for one. Afterwards, they can check the status of their application, make **queries** and book a flat with an **officer**."

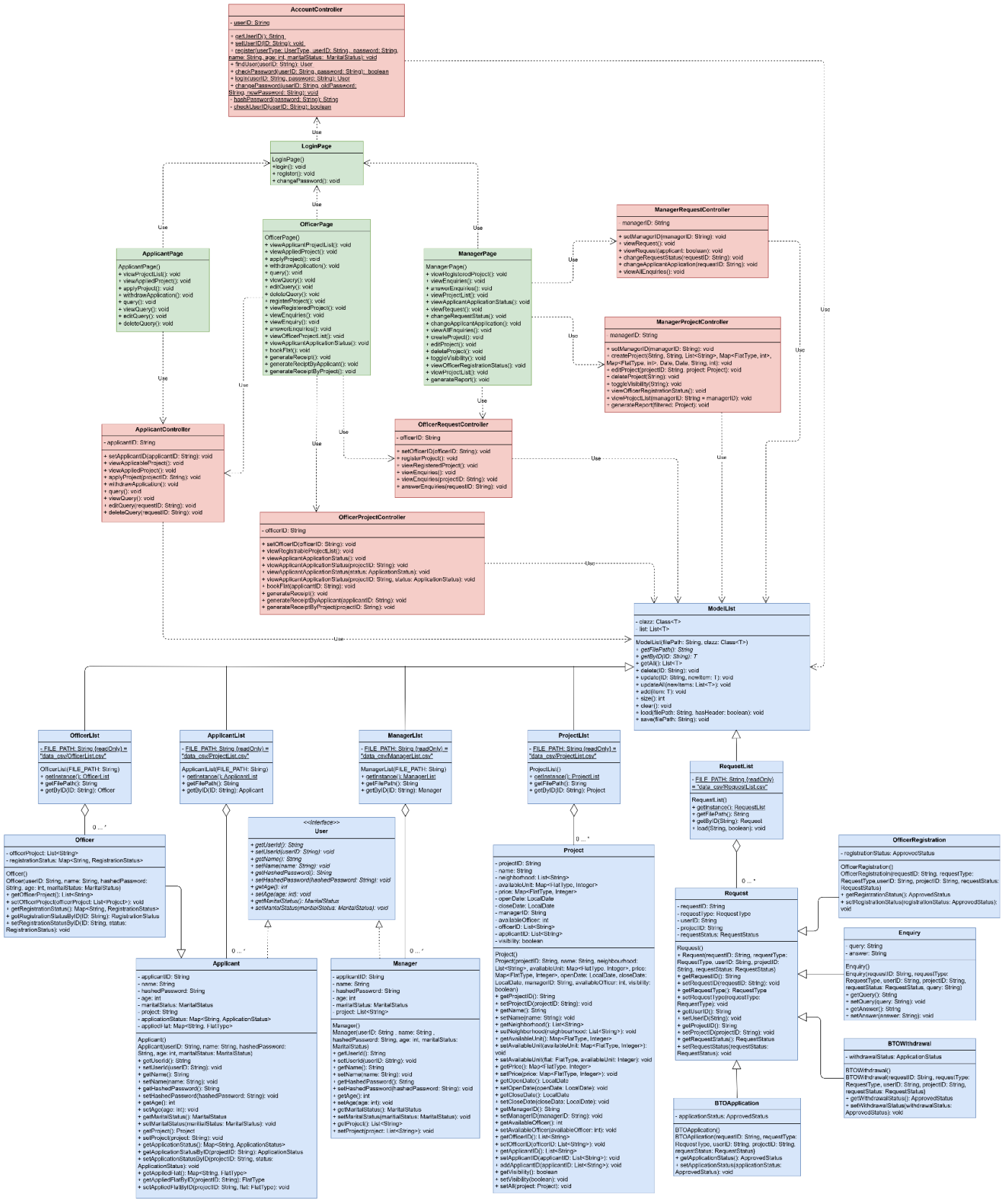
2.2 Reflection on Design Trade-offs

We considered creating a separate class for each application to allow easy data storage and future extensibility. However, this could lead to performance issues due to a possibility of a large database of application objects which could cause latency issues. As each application only holds minimal data, we chose a simpler design by storing application details as attributes of the applicant.

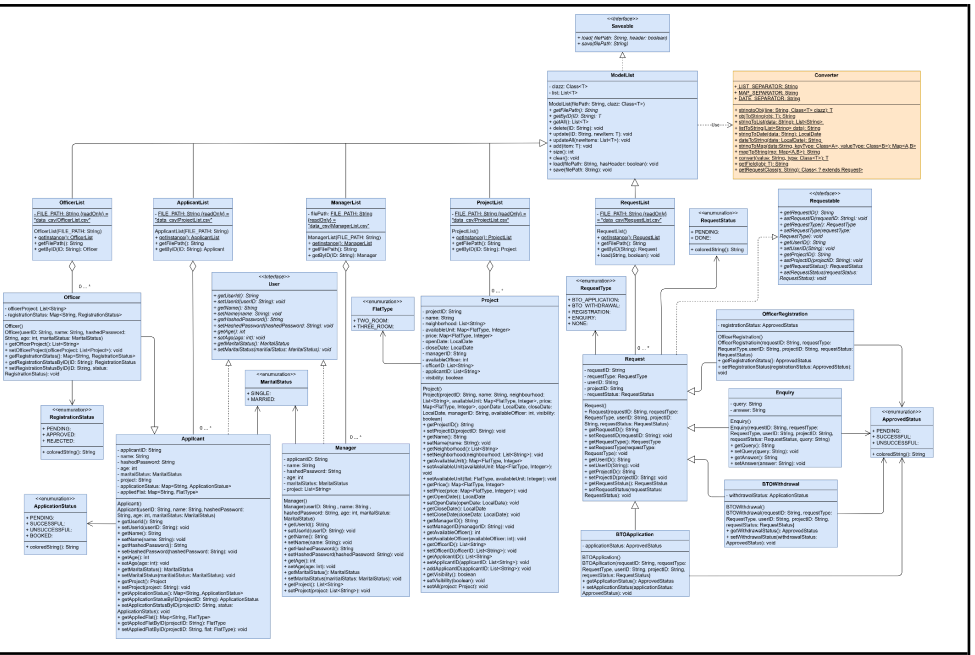
Another point of consideration is the use of a generic abstract ModelList<T> class. We used ModelList<T> for extensibility and to avoid code duplication. Using reflection, we allow the system to dynamically handle different object types. However, we believe this could result in a more complex code structure, potentially making it harder for new developers to understand. We concluded that having reusability and extensibility is more essential and with good documentation, the task of understanding the system can be made easier.

3. Object-Oriented Design

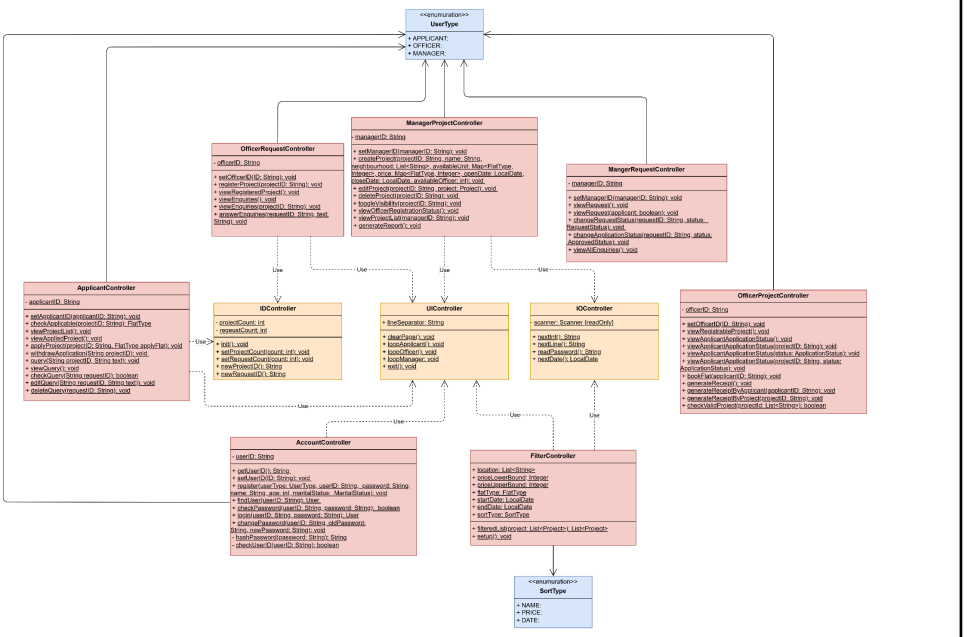
3.1 Class diagram



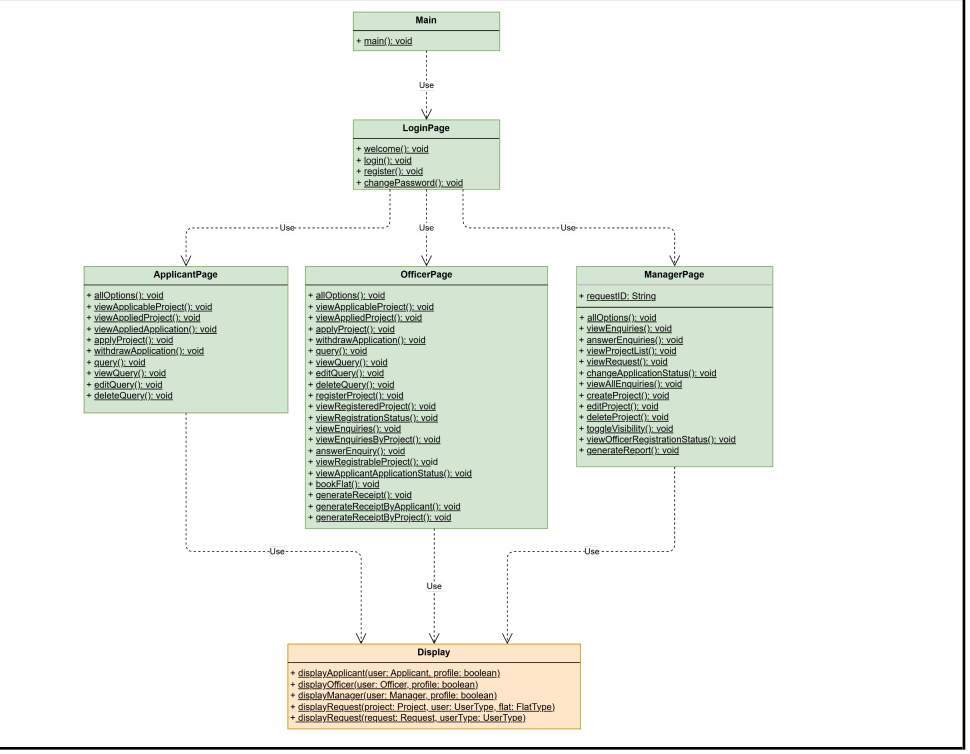
Entity Class Diagram



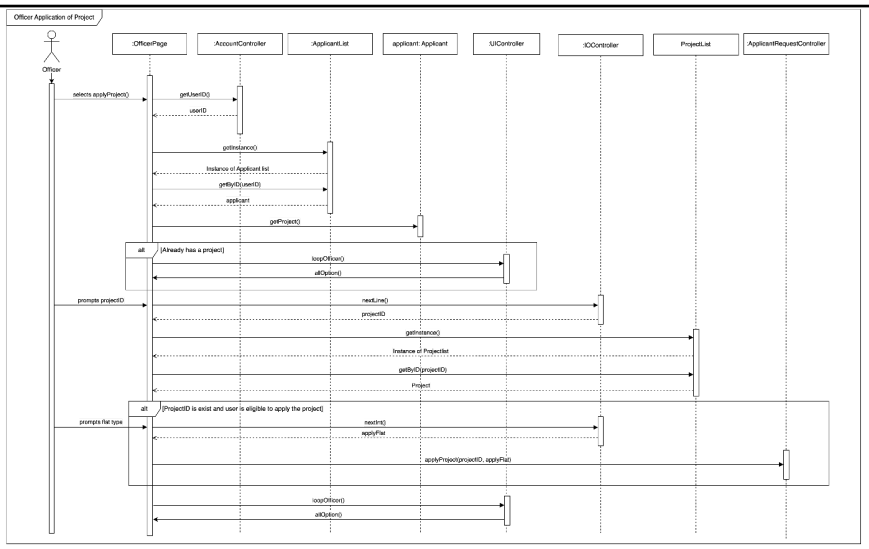
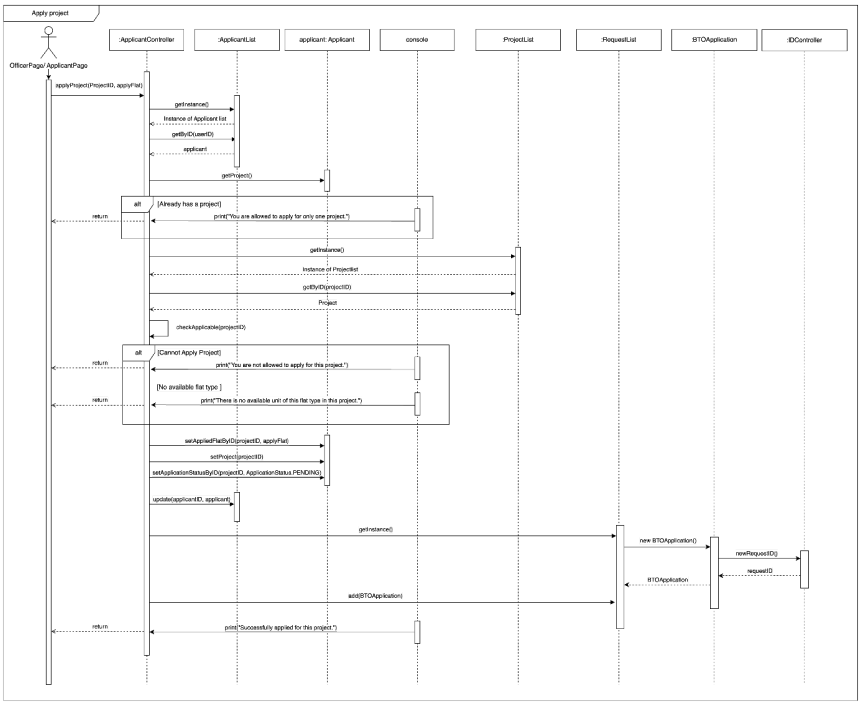
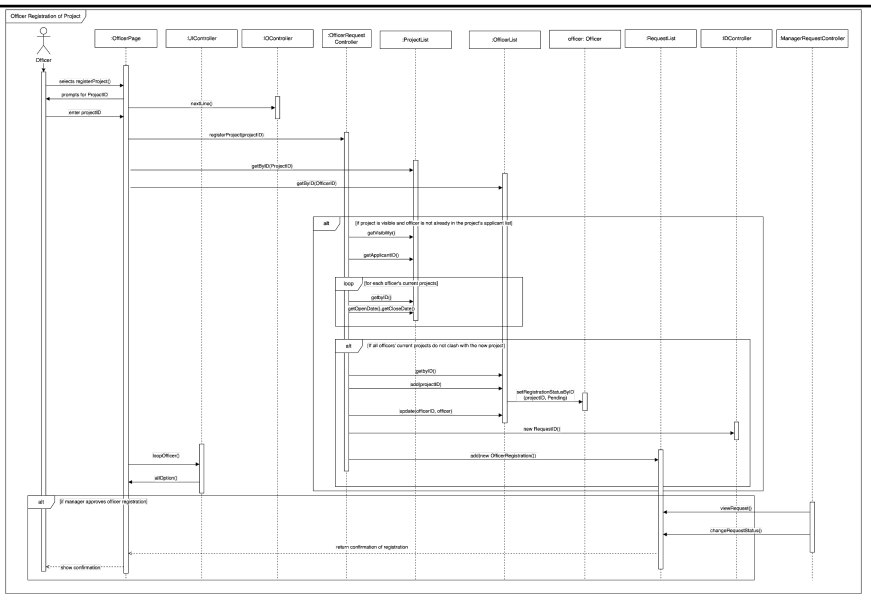
Controller Class Diagram



Boundary Class Diagram



3.2 Sequence diagram

Officer Application of Project	
Apply Project	
Officer Registration of Project	

3.3 Application of OOD Principles (SOLID)

We applied the **Single Responsibility Principle** in our control classes. This can be seen especially in our OfficerProjectController and OfficerRequestController classes. We split the actions that officers can take into project-related or request-related actions. The OfficerProjectController is responsible for handling project-related actions while the OfficerRequestController is responsible for handling request-related actions. This makes the system easier to test and maintain as users can easily make changes to one aspect of officer activity while ensuring that the other aspect does not get affected.

We applied the **Open/Closed Principle** in our user classes.We created a base User class which encapsulates the attributes and methods associated with it. We then designed the applicant, officer and manager classes as subclasses of the user class, extending on the capabilities of the user but also allowing them to add role-specific actions. This allows us to extend the basic functionality of a user without changing the implementation of the user classes. This also allows for easy extensibility if any new types of users are needed to be added to the system, further supporting the concept of polymorphism.

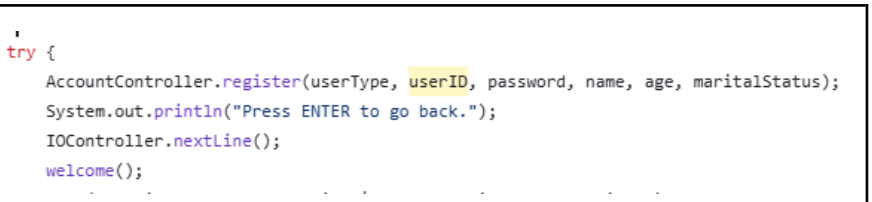
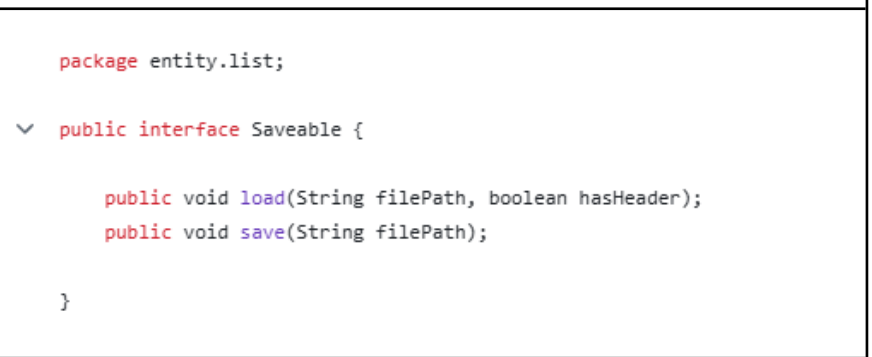
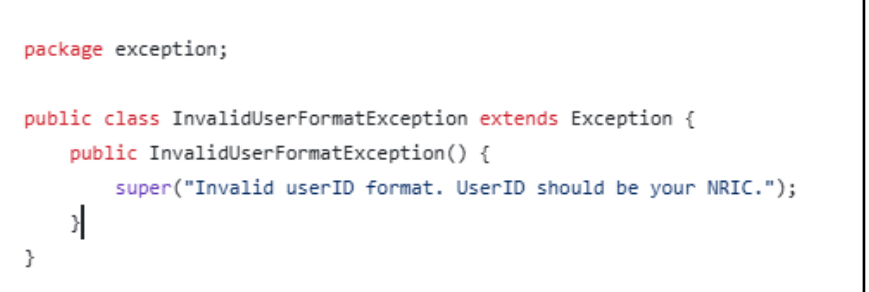
We applied the **Liskov Substitution Principle** in our Applicant and Officer classes. We designed the Officer class to be a subclass of the Applicant class, allowing us to pass an officer object into the system wherever an applicant object is expected. This allows us to make the code reusable and prevent duplicate work of creating another control class for the same implementation. We ensured that the Officer class follows design by contract ensuring that officers do not expect more from inputs than applicants and also ensuring that the outputs are no less than that of applicants.

We applied the **Interface Segregation Principle** in our boundary classes. In the system, there are three types of users. Hence, we separated the 3 user interfaces into 3 distinct boundary classes instead of using one UI page for all users. This kept the boundary class code cleaner and tailored to its corresponding user type. This also helps make the code easier to maintain as the pages can be changed in isolation without risking affecting the rest.

We applied the **Dependency Inversion Principle (DIP)** as seen in the AccountController. While we did not explicitly use interfaces, in our AccountController, we worked with the User class which is a base class of Applicants, Officers and Managers. Hence, for example, when a manager logs in, it works on the User class instead of specifically the Manager class which supports DIP as it works with a higher-level abstraction.

4. Implementation

Sample Code Snippets: The use of OOP concepts are implemented throughout the code.

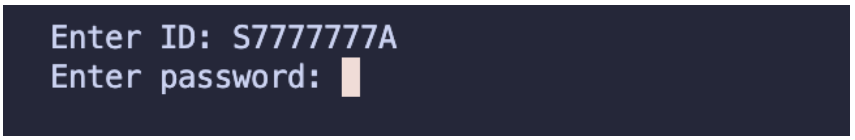
Use of encapsulation	
Use of inheritance	
Use of Polymorphism	
Interface Use	
Exception Handling	

5. Testing

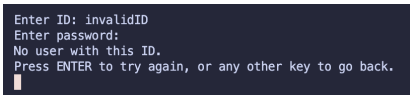
We adopted a **manual testing approach** to validate the system based on its functionalities and use cases.

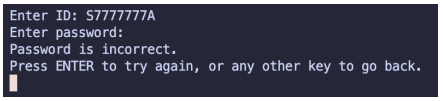
5.1 Registration System

Welcome Page: Upon entering the system, the user can choose 4 different choices.



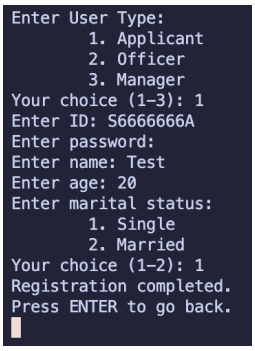
Login: Users are able to access their dashboard based on their role after successfully login.

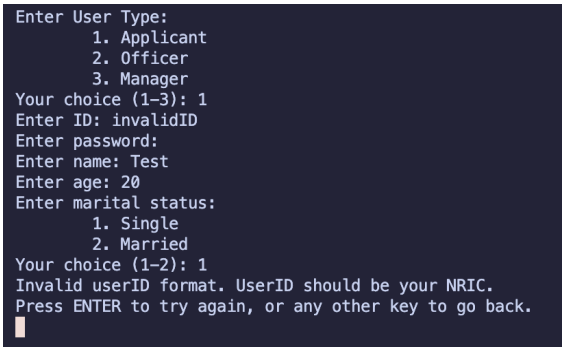


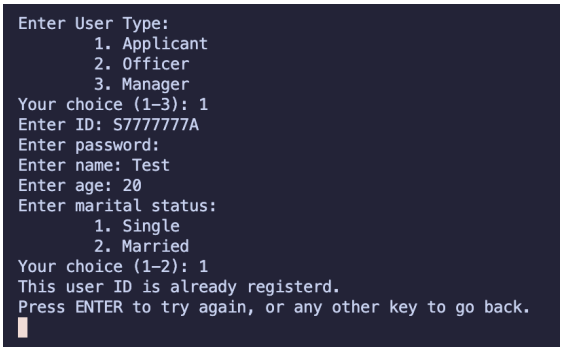


Invalid Login: Users are prompted to try again in the following scenarios:

1. User logins with invalid ID
2. User logins with incorrect password

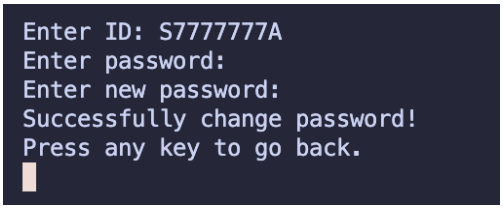


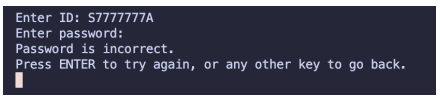


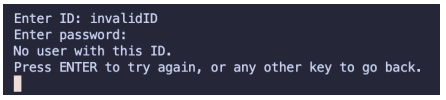


Register: Users are able to register for an account with their user type.
Invalid ID: Users are prompted to try again in the following scenarios:

1. User inputs invalid ID format (ID must be NRIC)
2. User inputs ID with existing account



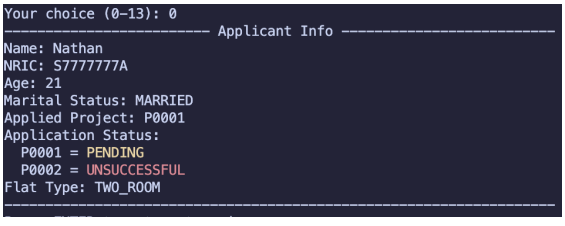




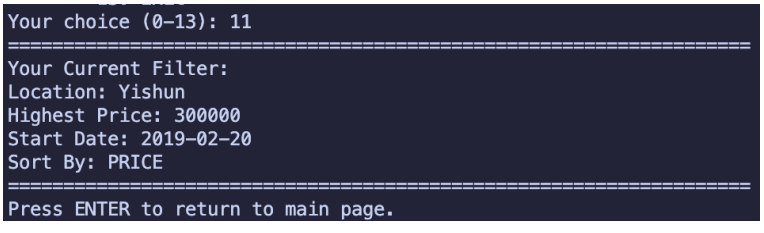
Change Password:
Users are able to change their password by entering their ID and password.
Invalid Information:
Users are prompted to try again in the following scenarios:

1. User inputs invalid ID.
2. User inputs incorrect password.

5.2 All Users



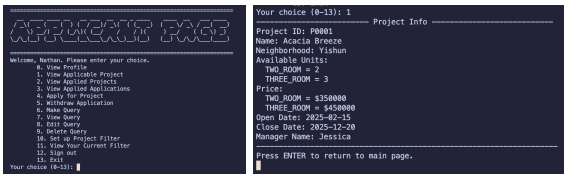
View Profile:
All users are able to view their profile.



Filter Project:
All users are able to apply filters when searching for projects or viewing their projects. They can view their current filters on the main page. The filter options are saved during their session (once they are logged out, it will change back to default options).

5.3 Applicants

Applicant Page: Upon entering the system, the applicant can choose 14 different choices.
View Applicable Project: Applicants are able to view their applicable projects based on attributes and visibility.



```
Your choice (0-13): 4
Enter the project ID to apply: P0001
Enter flat type:
        1. Two Room
Your choice: 1
Successfully applied for this project.
Press ENTER to return to main page.
```

Apply Project:
Applicants are able to apply for their applicable projects.

```
Your choice (0-13): 4
Enter the project ID to apply: P0002
Enter flat type:
        1. Two Room
Your choice: 1
You are not allowed to apply for this project.
Press ENTER to return to main page.
```

```
Your choice (0-13): 4
You are allowed to apply for only one project.
Press ENTER to return to main page.
```

Unsuccessful Application:
If the applicants try to apply for a project in the following cases, the system will notify the applicant.

1. Inapplicable project
2. Applicant already applied for a project

```
Your choice (0-13): 5
Enter the project ID to apply: P0001
Successfully request for withdrawal.
Press ENTER to return to main page.
```

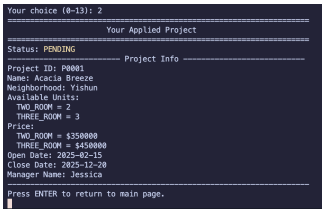
Withdraw Project:
Applicants are able to withdraw from their project. Once applied for the withdrawal, applicants can see its status in applied applications. Once withdrawal is approved, the project will be removed from that applicant profile.

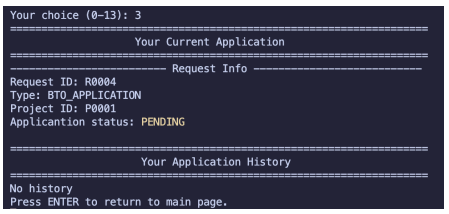
```
Your choice (0-13): 5
Enter the project ID to apply: P0001
You already applied withdrawal application for this project.
Press ENTER to return to main page.
```

```
Your choice (0-13): 5
Enter the project ID to apply: P0002
You haven't applied to this project.
Press ENTER to return to main page.
```

Unsuccessful Withdrawal Application:
If the applicants try to withdraw a project in the following cases, the system will notify the applicant.

1. Project that applicant didn't apply
2. Project that already applied for a withdrawal





```
Your choice (0-13): 0
---------- Applicant Info ----------
Name: John
NRIC: S1234567A
Age: 35
Marital Status: SINGLE
Applied Project: P0001
Application Status:
  P0001 = PENDING
Flat Type: TWO_ROOM
------------------------------------------------------------
Press ENTER to return to main page.
```

```
Your choice (0-13): 3
============================================================
                Your Current Application
============================================================
You haven't applied to any project.
============================================================
                Your Application History
============================================================
---------- Request Info ----------
Request ID: R0001
Type: BTO_APPLICATION
Project ID: P0001
Applicantion status: UNSUCCESSFUL

---------- Request Info ----------
Request ID: R0003
Type: BTO_WITHDRAWAL
Project ID: P0001
Withdrawal status: SUCCESSFUL

Press ENTER to return to main page.
```

View Applied Project:
Once applied, applicants can see their applied project details, and it will be displayed in applied applications and their profile.

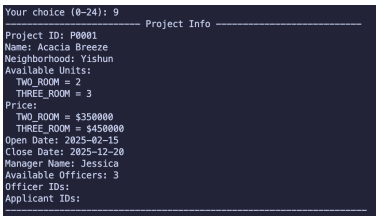
```
Your choice (0-13): 6
Enter the project ID to enquiry: P0001
Enter your query: Hi, how long is it to process?
Press ENTER to return to main page.
```

```
Your choice (0-13): 7
---------- Request Info ----------
Request ID: R0007
Type: ENQUIRY
Project ID: P0001
Status: PENDING
Query: Hi how long is it to process?
```

Enquiry:
Applicants are able to enquire about the applicable project they are interested in, and they are able to **view, edit, and delete** their enquiries. They are not allowed to inquire about the irrelevant project and view, edit, or delete other applicant's enquiries.

5.4 Officers

Officer Page:
Upon entering the system, the officer can choose 25 different choices, where choices 1-8 are the applicant's capabilities.



View Registrable Project:
Officers are able to view projects that they can register for, including additional information of that project.

```
Your choice (0–24): 12
Enter the project ID to register: P0001
Successfully applied registeration.
Press ENTER to return to main page.
```

Project Registration: Officers are able to register for a project

```
Your choice (0–24): 11
================================================================
                    Your Registration History
================================================================
------------------------- Request Info -------------------------
Request ID: R0008
Type: REGISTRATION
Project ID: P0001
User ID: S8888888A
Registration status: PENDING
```

View Registration: Once registered, officers can see their registered project details, and it will be displayed in registration applications and their profile as well.

```
Your choice (0–24): 16
------------------------- Project Info -------------------------
Project ID: P0001
Name: Acacia Breeze
Neighborhood: Yishun
Available Units:
  TWO_ROOM = 2
  THREE_ROOM = 3
Price:
  TWO_ROOM = $350000
  THREE_ROOM = $450000
Open Date: 2025-02-15
Close Date: 2025-12-20
Manager Name: Jessica
Available Officers: 2
Officer IDs: S8888888A
Applicant IDs:
------------------------------------------------------------------------
Status: PENDING
------------------------ Applicant Info ------------------------
Name: Nathan
NRIC: S7777777A
Age: 21
Marital Status: MARRIED
Flat Type: TWO_ROOM
------------------------------------------------------------------------
Status: SUCCESSFUL
------------------------ Applicant Info ------------------------
Name: John
NRIC: S1234567A
Age: 35
Marital Status: SINGLE
Flat Type: TWO_ROOM
------------------------------------------------------------------------
Press ENTER to return to main page.
```

View Handling Projects: Officers can view their projects (successfully registered)

```
Your choice (0–24): 18
------------------------ Applicant Info ------------------------
Name: John
NRIC: S1234567A
Age: 35
Marital Status: SINGLE
Flat Type: TWO_ROOM
------------------------------------------------------------------------
------------------------- Project Info -------------------------
Project ID: P0001
Name: Acacia Breeze
Neighborhood: Yishun
Available Units:
  THREE_ROOM = 3
  TWO_ROOM = 1
Price:
  THREE_ROOM = $450000
  TWO_ROOM = $350000
Open Date: 2025-02-15
Close Date: 2025-12-20
Manager Name: Jessica
Available Officers: 2
Officer IDs: S8888888A
Applicant IDs: S1234567A
------------------------------------------------------------------------
```

Generate Receipt:
Officers are able to generate receipts for each project and applicant with BOOKED status.

```
Your choice (0–24): 13
------------------------- Request Info -------------------------
Request ID: R0007
Type: ENQUIRY
Project ID: P0001
User ID: S7777777A
Status: PENDING
Query: Hi how long is it to process?
```

```
Your choice (0–24): 14
Enter the project ID to view: P0002
You are not allowed to view enquiries of other's project.
```

```
Your choice (0–24): 15
Enter the request ID to answer: R0007
Enter your answer: Normally 2 months
Press ENTER to return to main page.
```

```
------------------------- Request Info -------------------------
Request ID: R0007
Type: ENQUIRY
Project ID: P0001
User ID: S7777777A
Status: DONE
Query: Hi how long is it to process?
Answer: Normally 2 months
```

Manage Enquiries:
Officers are able to **view and answer** enquiries of their projects.

```
Your choice (0–24): 17
Enter the applicant ID to book a flat: S1234567A
Successfully booked a flat for this applicant.
Press ENTER to return to main page.
```

```
Your choice (0–13): 0
------------------------ Applicant Info ------------------------
Name: John
NRIC: S1234567A
Age: 35
Marital Status: SINGLE
Applied Project: P0001
Application Status:
  P0001 = BOOKED
Flat Type: TWO_ROOM
------------------------------------------------------------------------
```

Book Flat: Officers are able to book a flat for applicants with successful BTO application status. Then, the status of that applicant will change to BOOKED.

```
Your choice (0–24): 17
Enter the applicant ID to book a flat: S7777777A
This applicant hasn't been approved yet.
Press ENTER to return to main page.
```

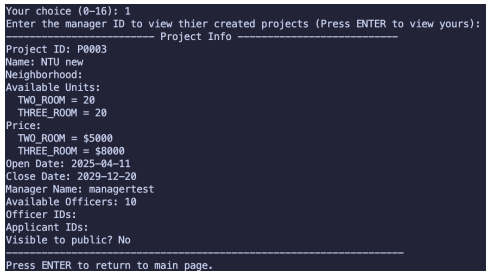
```
Your choice (0–24): 17
Enter the applicant ID to book a flat: S1234567A
Flat has been booked for this applicant already.
Press ENTER to return to main page.
```

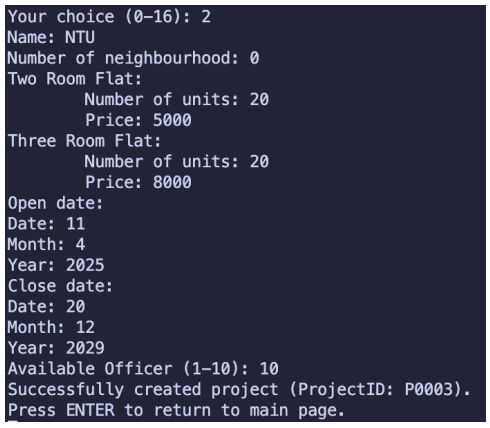
Invalid Book: Officers cannot book a flat in other officer's projects or more than one flat for applicants.

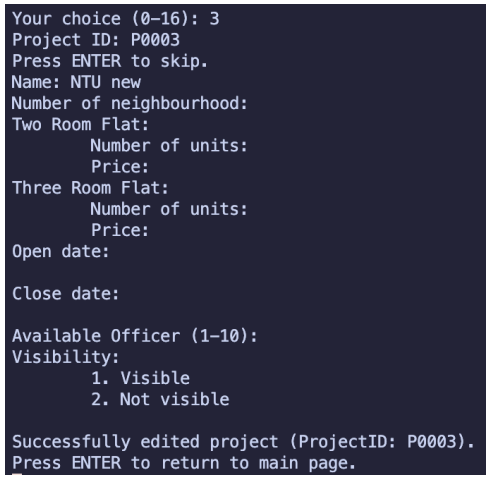
5.5 Managers

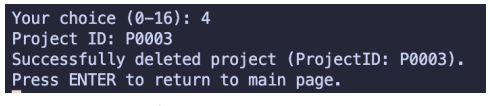
Manager Page:
Upon entering the system, the officer can choose 17 different choices.

Generate Report:
Managers are able to generate the report with filters for each project.

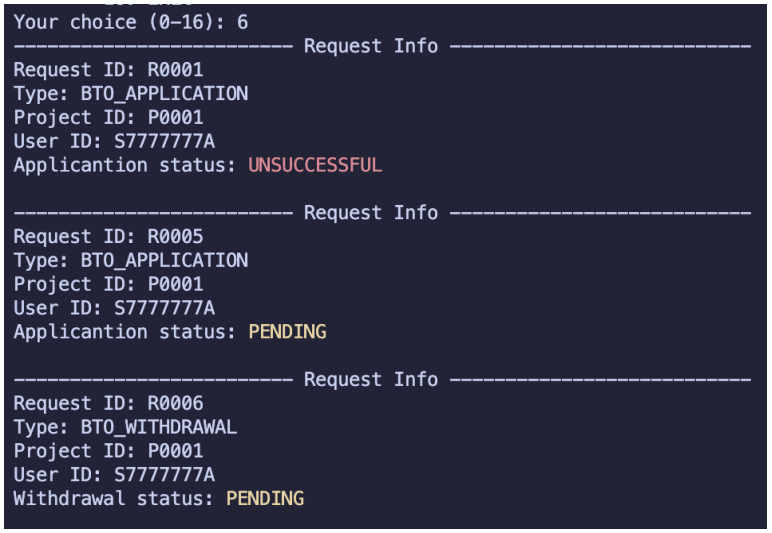


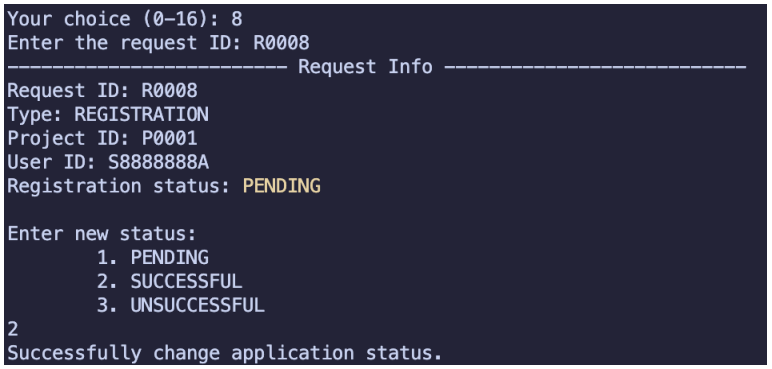


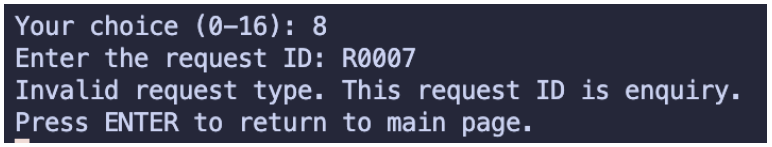




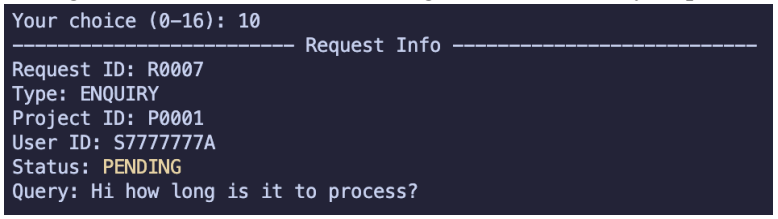
Manage Project:
Managers are able to **view, create, edit, and delete** projects.







Manage Request:
Managers are able to view and change the status of any request.



Manage Enquiries:
Managers are able to tackle any enquiries.

6. Documentation

Link to Javadoc: https://autoastt.github.io/BTO-OOP/javadoc

7. Reflection & Challenges

7.1 What Went Well

Our team successfully built a BTO application system that mirrors the real-world housing process by leveraging an inheritance hierarchy (User → Applicant → Officer) to model user roles and enable polymorphism. By applying SOLID principles, particularly the Single Responsibility Principle, we maintained a clean, modular codebase—evident in our separation of controllers like OfficerProjectController and OfficerRequestController. The use of boundary, control, and entity classes improved organization and maintainability. Our structured requirement analysis and feature prioritization framework ensured we focused on core functionalities within project constraints, making development efficient and goal-oriented.

7.2 What Could Be Improved

While the file-based persistence system suffices for the current project scope, it poses limitations in scalability, data integrity, and query efficiency—challenges that a proper database could address more effectively. The command-line interface, though functional, could be improved with a graphical interface to enhance usability and accessibility for non-technical users. Additionally, the system's error handling could be more robust, with clearer messages and better recovery mechanisms, especially for complex validations and potential file operation failures.

7.3 Lessons Learned About OODP

This project emphasized the value of thorough planning, object-oriented design, and clean architecture. By analyzing requirements early, we avoided major redesigns and applied inheritance and polymorphism to model user roles effectively—allowing the Officer class to extend Applicant, and Applicant to extend User, enhancing reusability and maintainability. Encapsulation helped manage complexity, particularly in refining the application status logic. We also learned to navigate design trade-offs, such as simplifying the model by embedding status in the Applicant class. Implementing the Model-View-Controller pattern further improved code organization and maintainability. Overall, the project deepened our understanding of software design and development.

8. Appendix

8.1 GitHub Repository

The complete source code project is available on GitHub at: https://github.com/autoastt/BTO-OOP

Link to documentation: https://autoastt.github.io/BTO-OOP

8.2 Tools and References

draw.io, Java 17, Eclipse/IntelliJ IDEA/VSCode, GitHub, GitHub Desktop, Javadoc, Google Docs

All file persistence operations were implemented using Java's standard I/O libraries to store and retrieve data in CSV format. This approach was chosen to keep the implementation straightforward while providing the necessary data persistence capabilities for the BTO application system.