SC2002 Object-Oriented Design and Programming

Build-to-Order Management System Final Report

Link to Github Repository: https://github.com/tanck04/btoms

Team Members:

AGARWAL DHRUVIKAA, U2423574H

HE HAOYU, U2420885A

KHOR HAOJUN, U2420982C

LEE BECKHAM, U2422260H

TAN CHUEN KEAT, U2420792G

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 the 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
AGARWAL DHRUVIKAA	SC2002	FDAF	D.Agarwal 22/4/2025
KHOR HAOJUN	SC2002	FDAF	22/4/2025
HE HAOYU	SC2002	FDAF	22/4/2025
LEE BECKHAM	SC2002	FDAF	22/4/2025
TAN CHUEN KEAT	SC2002	FDAF	22/4/2025

1. Introduction

Our group developed a Build-To-Order Management System (BTOMS) using object-oriented programming concepts and design principles in this project. This report outlines the key design decisions made throughout development, presents a UML class diagram that illustrates the structural relationships within the system, details various test cases used to validate the core functionalities of BTOMS, and shares our group's reflections on the development journey. It also highlights the challenges we encountered, how we overcame them, and the insights we gained while designing and implementing this real-world application.

1.1 Understanding the Problem and Requirements

We began by carefully analyzing the BTO document line by line, highlighting all system requirements and use cases. From this, we identified the main problem domain as a centralized system for managing BTO applications involving different user roles: Applicants, HDB Officers, and HDB Managers. Explicit requirements such as user login, project visibility, application submission, and role-based actions were clearly noted. We inferred implicit expectations like status tracking, input validation, and data consistency. Any ambiguities—such as age limits or overlapping roles—were resolved through group discussion and assumptions aligned with the project's scope.

1.2 Deciding on Features and Scope

We categorized all identified features into three groups: core, optional, and excluded. Core features include user authentication, project creation and visibility control, BTO application and booking, officer registration, and application processing—essential for meeting the requirements. Optional features, such as enquiry editing and advanced filtering in reports, were considered only if time allowed. Features like integration with external systems (e.g., Singpass verification) were excluded due to time constraints and complexity. This prioritization helped ensure a functional and manageable system demonstrating strong object-oriented principles.

2. Design Consideration

2.1 Model-View-Controller (MVC) design architecture/framework

We designed our Build-To-Order Management System (BTOMS) following the Model-View-Controller (MVC) architectural pattern. This design allows for a scalable, modular, and maintainable structure by separating the system into three main components: *Model, View, and Controller*. With this separation of concerns, business logic is kept distinct from user interface logic, making it easier to update or extend the application in the future. For example, changes made to data handling in the Model do not require changes to the View or Controller, enabling more efficient development and testing.

2.1.1 Model

The model package contains classes such as *User, Applicant, Officer, Manager, Project,* and *Application*. These classes represent the core data entities in our system. They define the structure and behavior of data used throughout BTOMS, including the various roles and their attributes, project details, and application records

2.1.2 View

User interaction is handled within the view package, which includes classes like *OfficerView, ApplicantView*, and *ManagerView*. These classes present different menu interfaces to users based on their roles, prompt them for input, and relay that input to the corresponding controller for processing. This separation ensures the user interface can evolve independently from the application logic.

2.1.3 Controller

The controller package includes classes such as *LoginController, ApplicationController, HDBOfficerController*, and *EnquiryController.* Controllers serve as the bridge between the *View* and *Model*. They receive user input from the *View*, process that input (including validation), and call upon *Model* classes to perform the necessary operations, such as retrieving data or updating records.

2.2 Fundamental principles of Object-Oriented Programming (OOP):

2.2.1 Abstraction

Abstraction refers to hiding complex implementation details and exposing only the relevant information to users. In BTOMS, the *User* class serves as an abstract superclass that defines only the essential attributes shared across different user roles, such as *NRIC*, *name*, *age*, and *maritalStatus*. Unnecessary details that are irrelevant to the system's functionality are excluded. The MVC architecture also enhances abstraction by organizing the system into components that only expose the necessary interface for interaction.

2.2.2 Encapsulation

Encapsulation protects an object's internal state by restricting direct access to its fields and instead exposing them through public methods. For instance, all attributes within the *User, Project*, and *Application* classes are declared as *private* and are accessed or modified only through their respective getter and setter methods. This ensures that data is securely managed and accessed in a controlled manner. Additionally, the MVC structure enforces encapsulation by separating the responsibilities of the Model, View, and Controller layers — each interacts through well-defined interfaces without accessing each other's internal logic directly.

2.2.3 Inheritance

Inheritance allows child classes to reuse attributes and methods from a parent class, promoting code reusability and reducing duplication. In our system, *Applicant*, *Officer*, and *Manager* classes all extend the abstract *User* class. This inheritance structure allows all common properties, such as NRIC, name, and associated methods, to be reused across user types, while still allowing each subclass to introduce its specialized behavior.

2.2.4 Polymorphism

Polymorphism allows objects of different classes to be treated through a common interface while exhibiting behavior specific to their type. This is evident in our implementation of the *MenuInterface*, which defines the method *displayMenu()*. Each user role (e.g., *ApplicantView, OfficerView,* and *ManagerView*) implements this method to display role-specific menus. Though the method name is the same, the output and functionality are tailored to the needs of the particular user type, demonstrating both interface-based and method overriding polymorphism.

2.3 SOLID Design Principles

2.3.1 Single Responsibility Principle

The Single Responsibility Principle asserts that a class should have only one reason to change, meaning it should focus on a single task or responsibility. In BTOMS, this principle is demonstrated in the *ApplicationController* class. Its sole function is to handle logic related to housing applications, such as submitting applications, checking their status, and processing withdrawals. It does not contain unrelated code for user authentication, project management, or data persistence. This dedicated responsibility ensures that any future changes to application logic can be made in one place without risking unintended side effects on other parts of the system. This separation also simplifies testing and debugging.

2.3.2 Open-Closed Principle (OCP)

The Open-Closed Principle states that software entities (like classes and methods) should be open for extension but closed for modification. This means we should be able to add new functionality without changing existing code. In our system, this is exemplified through the design of the *User* abstract class. It defines core attributes and behaviors (e.g., NRIC, name, password, role) that are shared by all user types. Specialized user roles like *Applicant, Officer,* and *Manager* extend this class and implement additional behaviors without modifying the base class itself. If new user roles (e.g., *Auditor* or *Contractor*) need to be added in the future, they can simply extend the *User* class—no changes to the original logic are necessary. This makes the system both extensible and stable.

2.3.3 Liskov Substitution Principle

The Liskov Substitution Principle requires that subclasses must be substitutable for their parent class without altering the correctness of the program. In BTOMS, all user types (*Applicant*, *Officer*, and *Manager*) inherit from the abstract *User* class and can be passed into methods expecting a *User* object. For example, the *displayMenu(User user)* method can receive any subclass of *User*, and the appropriate menu interface is presented based on the role. Although each subclass may have additional fields or methods specific to its function, they uphold the expected behavior of the base class. This allows polymorphism to work effectively and guarantees that new subclasses won't break existing functionality that depends on the base type.

2.3.4 Interface Segregation Principle

The Interface Segregation Principle emphasizes that classes should not be forced to implement interfaces they do not use. In BTOMS, this principle is demonstrated in the design of our role-specific view classes, such as *OfficerView, ApplicantView*, and *ManagerView*. Each of these classes implements a version of the *MenuInterface* but only includes methods and logic relevant to its specific user type. For instance, *ApplicantView* includes options like applying for a flat or checking application status, while *OfficerView* focuses on handling enquiries or registering for projects. These views are not overloaded with functionalities irrelevant to their roles, resulting in cleaner, more maintainable, and easily testable code.

Beyond *MenuInterface*, we also applied the Interface Segregation Principle through the use of specialized interfaces such as *CheckSecQuesInterface*, *SecQuesChangerInterface*, *PasswordChangerInterface*, *VerificationInterface*, and *ViewProjectInterface*. These interfaces isolate specific responsibilities—for example, handling security questions, password changes, and identity verification—so that classes only implement what they need. This modular design enhances maintainability and testability, while also improving usability and security by ensuring users only interact with operations permitted for their role.

2.3.5 Dependency Inversion Principle

The Dependency Inversion Principle states that high-level modules should not depend on low-level modules, but both should depend on abstractions. This principle is applied in BTOMS through our use of repository classes like *ApplicationRepository*, *ProjectRepository*, and *ApplicantRepository*. These repositories handle all data storage and retrieval logic, such as reading from and writing to CSV files. High-level components like *ApplicationController* and *OfficerView* interact with these repositories through their public interfaces without needing to know how the data is stored. As a result, the underlying data source could be changed (e.g., from CSV files to a database) with minimal changes to the controller or UI logic. This abstraction not only decouples modules but also allows for easier testing, as repositories can be replaced with mock data providers.

2.4 Relationships and Dependencies

2.4.1 Associations

In BTOMS, unidirectional associations are used to ensure the clean separation of responsibilities between system components. Controllers have access to both model and repository classes, but the reverse is not true—models and repositories are not aware of controllers. This design ensures loose coupling and enhances maintainability. For instance, the *OfficerController* interacts with the *ApplicationRepository* and *Project* model classes to manage tasks like viewing projects or submitting applications. This directional flow of logic ensures that the controller acts as an intermediary, coordinating data flow between views and repositories without introducing unnecessary dependencies in the data layer.

2.4.2 Dependencies

The View layer depends on Controllers and Repositories to perform operations and retrieve data. This reflects the core design of the MVC architecture, where the view is responsible for user interaction but delegates all data processing and business logic to the controller. For example, *OfficerView* relies on *OfficerController* to fetch project data or submit applications. The controller, in turn, calls methods from relevant repositories such as *ProjectRepository* or *ApplicationRepository*. This chain of dependency ensures that each layer remains focused on its responsibility—Views handle user interfaces, Controllers manage application logic, and Repositories handle data persistence.

2.4.3 Aggregation/Composition

BTOMS uses both aggregation and composition to model relationships between objects, depending on their lifecycles. Aggregation represents a "has-a" relationship where the child can exist independently of the parent. For example, a *Project* class may aggregate multiple *Officer* objects, but an *Officer* still exists independently even if the *Project* is removed. In contrast, composition represents a stronger "part-of" relationship, where the child object cannot exist without the parent. A clear example of this in BTOMS is found in the *Project* class, which holds two collections: *flatTypeUnits* and *flatTypePrices*, implemented as maps. These collections store flat type information and are owned exclusively by the Project. They have no meaning or use outside the *Project* context. When a *Project* is deleted, these collections are also deleted—demonstrating a strict lifecycle dependency. Furthermore, the *Project* class encapsulates and controls all access to these collections, making it a textbook example of composition in real-world design.This use of composition ensures a tight coupling where needed (i.e., the flat types are an intrinsic part of the project) while maintaining clarity in ownership, data flow, and memory management.

3. Reflection

3.1 Difficulties Encountered and Ways to Conquer

During the initial phases of the project, one of the biggest challenges we encountered was arriving at a cohesive and functional system design. Each of us approached the design process from the perspective of our assigned user roles—officer, applicant, or manager. We each drafted our own version of a UML diagram, thinking this would make development more efficient. However, when we tried to merge our diagrams, we quickly realized the consequences of not aligning our understanding earlier: many classes had overlapping responsibilities, and there was no consistent logic governing how different components should communicate with one another. Additionally, our early attempts at designing interfaces resulted in overly complex, tightly coupled structures. These “fat interfaces” attempted to handle too many unrelated tasks, which made the system fragile and hard to maintain. On top of that, we struggled with deciding where exactly business logic should reside—some of it ended up in the views, some in the models, and much of it was duplicated across controllers. This lack of structure became evident once we started implementing our features, as changes in one part of the system often broke other unrelated parts.

3.2 How We Overcame Them

Recognizing the growing complexity and disjointedness of our system, we decided to pause implementation and regroup as a team. We agreed to rethink our architecture from scratch and began by outlining the core responsibilities of each class based on the Single Responsibility Principle (SRP). This led us to adopt the Model-View-Controller (MVC) architectural pattern, which gave us a clear way to separate concerns across our application. By assigning distinct responsibilities to the Model, View, and Controller layers, we were able to split our workload and avoid stepping into each other’s code unnecessarily. For instance, models handled the internal structure and data for entities like *User*, *Project*, and *Application*; views took charge of user interaction; and controllers managed logic and coordination between views and models. We also revisited our interfaces and refactored them to align with the Interface Segregation Principle (ISP), breaking down large, all-purpose interfaces into smaller, role-specific ones.

This made our system easier to scale and prevented unnecessary dependencies between unrelated modules. To reduce tight coupling, we introduced abstractions between high-level and low-level classes, adhering to the Dependency Inversion Principle (DIP). Once our revised UML diagram reflected this new structure, everything began to fall into place. The application became easier to debug, easier to extend, and far more modular than our earlier version. When we later added optional features like the security question mechanism for password recovery, it was easy to integrate with minimal changes—something that would have been extremely difficult with our earlier design.

3.3 Knowledge Learned

This project taught us the importance of having a solid architectural foundation before jumping into implementation. Our initial fragmented design approach led to repeated rework, and we realized that a unified, SOLID-compliant UML diagram would have streamlined development significantly. Through this, we gained a deeper understanding of design patterns—especially the MVC architecture—and how applying principles like SRP and DIP improves modularity and collaboration. We also strengthened our grasp of object-oriented concepts such as inheritance, encapsulation, and polymorphism through hands-on application. On the technical side, we learned to manage data using CSV files and HashMaps, which allowed us to build a lightweight yet functional system without relying on databases. Additionally, documenting our code with Javadoc proved useful in keeping the team aligned and simplifying future maintenance.

3.4 Additional Features

1. **Account Retrieval via Security Questions**

This feature enables users to recover their accounts in the event they forget their passwords. During account creation, users are prompted to set up security questions and answers. During retrieval, the system validates the user based on their responses. This provides a user-friendly recovery mechanism while maintaining authentication integrity. Low coupling and high cohesion are demonstrated through the separation of responsibilities: the *CheckSecQuesInterface* and its related implementations handle all security question logic independently, without interfering with unrelated authentication processes

2. **Password Security with SHA-256 Hashing**

All passwords stored in the system are secured using SHA-256 hashing. This ensures that sensitive data, such as user credentials, is never stored in plaintext, thereby strengthening overall system security and protecting user information from potential breaches. This hashing functionality is cohesively grouped within the authentication logic, and loosely coupled from user account management and other unrelated system modules.

3. **Automatic Role Detection During Login**

The system automatically determines the user's role (e.g., Applicant, Officer, Manager) based on the NRIC entered to streamline the login process and enhance usability. This eliminates the need for manual role selection and reduces the likelihood of user error during authentication. The implementation maintains high cohesion within the login

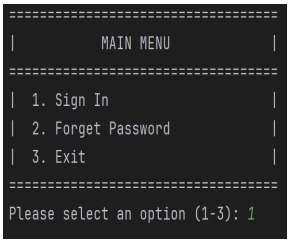
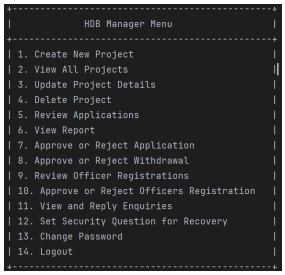
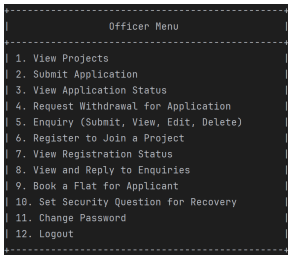
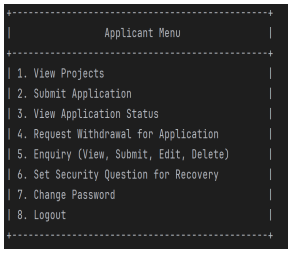
module while remaining loosely coupled from the individual view classes. This ensures changes in role-detection logic do not affect unrelated components.

4. **Improved Data Display through Structured Tables**

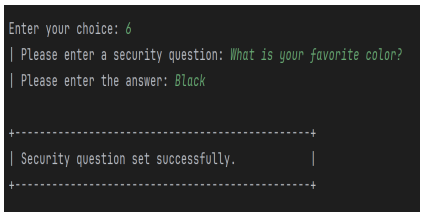
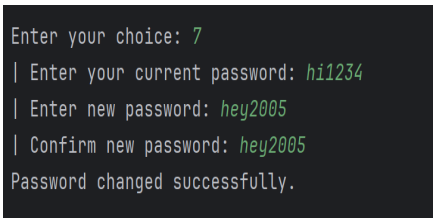
User data, such as enquiry records, is displayed in a clean and structured tabular format. These tables include text wrapping functionality to ensure long entries, such as enquiry messages, remain readable and do not overflow the display. This enhances the clarity and usability of terminal-based interfaces. Display logic is handled independently through utility functions or dedicated output handlers, maintaining low coupling from business logic and ensuring high cohesion in presentation-related components.

4. TEST CASES:

4.1 Menus:

Main Menu	Manager Menu	Officer Menu	Applicant Menu
			

Functions shared as User

User 1: Set Security Question for recovery	User 2: Change Password
	

4.2 Applicant

Test Case 1: View Projects

```
Enter your choice: 1
All Available Projects:
+------------+---------------------+---------------+-----------------------+----------------------+
|                                          Project List                                          |
+------------+---------------------+---------------+-----------------------+----------------------+
| Project ID | Project Name        | Neighbourhood | App. Start            | App. End             |
+------------+---------------------+---------------+-----------------------+----------------------+
| P0002      | THE LAKESHORE       | LAKESIDE      | 04/14/2025            | 07/28/2025           |
+------------+---------------------+---------------+-----------------------+----------------------+
| Flat Types:                                                                                     |
|    - THREE_ROOMS : $2000000.00    (150 units available)                                         |
|    - TWO_ROOMS   : $1500000.00    (200 units available)                                         |
+------------+---------------------+---------------+-----------------------+----------------------+
| P0003      | THE GREEN HOUSE     | TAMPINES      | 07/28/2025            | 09/28/2025           |
+------------+---------------------+---------------+-----------------------+----------------------+
| Flat Types:                                                                                     |
|    - THREE_ROOMS : $2500000.00    (100 units available)                                         |
|    - TWO_ROOMS   : $2000000.00    (150 units available)                                         |
+------------+---------------------+---------------+-----------------------+----------------------+
| P0004      | THE YISHUN HAVEN    | YISHUN TOWN   | 09/26/2025            | 12/31/2025           |
+------------+---------------------+---------------+-----------------------+----------------------+
| Flat Types:                                                                                     |
|    - THREE_ROOMS : $3000000.00    (150 units available)                                         |
|    - TWO_ROOMS   : $2500000.00    (200 units available)                                         |
+------------+---------------------+---------------+-----------------------+----------------------+
```

```
Would you like to apply a filter? (yes/no): yes
Enter neighbourhood to filter by (or leave blank): LAKESIDE

Filtered Projects:
+------------+---------------------+---------------+-----------------------+----------------------+
|                                          Project List                                          |
+------------+---------------------+---------------+-----------------------+----------------------+
| Project ID | Project Name        | Neighbourhood | App. Start            | App. End             |
+------------+---------------------+---------------+-----------------------+----------------------+
| P0002      | THE LAKESHORE       | LAKESIDE      | 04/14/2025            | 07/28/2025           |
+------------+---------------------+---------------+-----------------------+----------------------+
| Flat Types:                                                                                     |
|    - THREE_ROOMS : $2000000.00    (150 units available)                                         |
|    - TWO_ROOMS   : $1500000.00    (200 units available)                                         |
+------------+---------------------+---------------+-----------------------+----------------------+
```

Test Case 2: Submit application

```
Enter your choice: 2

Currently applied filters:
Neighbourhood: LAKESIDE
Flat Type: All
+------------+---------------------+---------------+-----------------------+----------------------+
|                                          Project List                                          |
+------------+---------------------+---------------+-----------------------+----------------------+
| Project ID | Project Name        | Neighbourhood | App. Start            | App. End             |
+------------+---------------------+---------------+-----------------------+----------------------+
| P0002      | THE LAKESHORE       | LAKESIDE      | 04/14/2025            | 07/28/2025           |
+------------+---------------------+---------------+-----------------------+----------------------+
| Flat Types:                                                                                     |
|    - THREE_ROOMS : $2000000.00    (150 units available)                                         |
|    - TWO_ROOMS   : $1500000.00    (200 units available)                                         |
+------------+---------------------+---------------+-----------------------+----------------------+
Do you want to apply/change the filters? (yes/no): yes
Enter neighbourhood to filter by (or leave blank): LAKESIDE
+------------+---------------------+---------------+-----------------------+----------------------+
|                                          Project List                                          |
+------------+---------------------+---------------+-----------------------+----------------------+
| Project ID | Project Name        | Neighbourhood | App. Start            | App. End             |
+------------+---------------------+---------------+-----------------------+----------------------+
| P0002      | THE LAKESHORE       | LAKESIDE      | 04/14/2025            | 07/28/2025           |
+------------+---------------------+---------------+-----------------------+----------------------+
| Flat Types:                                                                                     |
|    - THREE_ROOMS : $2000000.00    (150 units available)                                         |
|    - TWO_ROOMS   : $1500000.00    (200 units available)                                         |
+------------+---------------------+---------------+-----------------------+----------------------+
```

```
Enter Project ID to apply for: P0002

====== Available Flat Types ======
THREE_ROOMS - Units available: 150
TWO_ROOMS - Units available: 200

Select Flat Type:
1 - THREE_ROOMS
2 - TWO_ROOMS
Enter your choice (number): 1

Confirm application submission? (Y/N): Y
Application submitted successfully. Application ID: A0003
Application submitted successfully!
```

Test Case 3: View Application Status

```
Enter your choice: 3
Application ID: A0003
Project ID: P0002
Flat Type: THREE_ROOMS
Application Status: PENDING
Withdrawal Status: NULL
```

Test Case 4: Request Withdrawal for Application

```
Enter your choice: 4
Application ID: A0003
Project ID: P0002
Flat Type: THREE_ROOMS
Withdrawal Status: NULL
Enter 1 to confirm withdrawal, 0 to cancel:
1
Processing withdrawal for applicant: T0000006A
Updated application saved successfully.
Withdrawal request submitted successfully.
```

Test Case 5: Enquiry (View, Submit, Edit, Delete)

```
+-------------------------------------------+
|              Enquiry Menu                 |
+-------------------------------------------+
| 1. View Enquiries                         |
| 2. Submit Enquiry                         |
| 3. Edit Enquiry                           |
| 4. Delete Enquiry                         |
| 5. Back to Previous Menu                  |
+-------------------------------------------+
Enter your choice: 2
Enter Project ID: P0002
Enter your enquiry: Is the MRT Station nearby?
Enquiry submitted successfully.
```

4.3 Officer:

Functions As Applicant

Test Case 1: View Projects

Test Case 2: Submit Application

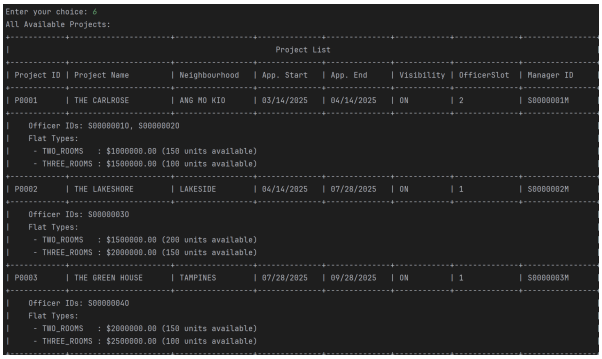
Test Case 3: View Application Status

Test Case 4: Request Withdrawal for Application

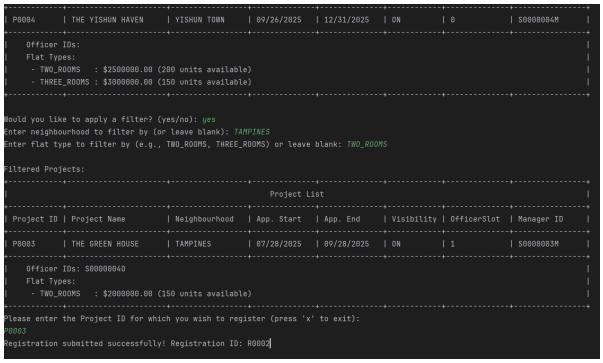
Test Case 5: Enquiry (View, Submit, Edit, Delete)

Functions As an Officer

Test Case 6: Register for a Project



```
Enter your choice: 6
All Available Projects:
+------------+-------------------------+-----------------+-------------+-------------+------------+-----------------+
|                                                  Project List                                                      |
+------------+-------------------------+-----------------+-------------+-------------+------------+-----------------+
| Project ID | Project Name            | Neighbourhood   | App. Start  | App. End    | Visibility | OfficerSlot     | Manager ID      |
+------------+-------------------------+-----------------+-------------+-------------+------------+-----------------+
| P0001      | THE CARLROSE            | ANG MO KIO      | 03/14/2025  | 04/14/2025  | ON         | 2               | S0000001M       |
+------------+-------------------------+-----------------+-------------+-------------+------------+-----------------+
|    Officer IDs: S00000010, S00000020                                                                               |
|    Flat Types:                                                                                                     |
|     - TWO_ROOMS   : $1000000.00 (150 units available)                                                              |
|     - THREE_ROOMS : $1500000.00 (100 units available)                                                              |
+------------+-------------------------+-----------------+-------------+-------------+------------+-----------------+
| P0002      | THE LAKESHORE           | LAKESIDE        | 04/14/2025  | 07/28/2025  | ON         | 1               | S0000002M       |
+------------+-------------------------+-----------------+-------------+-------------+------------+-----------------+
|    Officer IDs: S00000030                                                                                          |
|    Flat Types:                                                                                                     |
|     - TWO_ROOMS   : $1500000.00 (200 units available)                                                              |
|     - THREE_ROOMS : $2000000.00 (150 units available)                                                              |
+------------+-------------------------+-----------------+-------------+-------------+------------+-----------------+
| P0003      | THE GREEN HOUSE         | TAMPINES        | 07/28/2025  | 09/28/2025  | ON         | 1               | S0000003M       |
+------------+-------------------------+-----------------+-------------+-------------+------------+-----------------+
|    Officer IDs: S00000040                                                                                          |
|    Flat Types:                                                                                                     |
|     - TWO_ROOMS   : $2000000.00 (150 units available)                                                              |
|     - THREE_ROOMS : $2500000.00 (100 units available)                                                              |
+------------+-------------------------+-----------------+-------------+-------------+------------+-----------------+
```

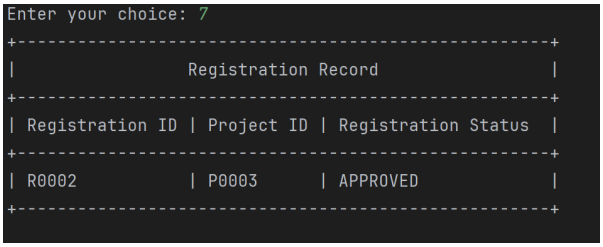


```
| P0004      | THE YISHUN HAVEN        | YISHUN TOWN     | 09/26/2025  | 12/31/2025  | ON         | 0               | S0000004M       |
+------------+-------------------------+-----------------+-------------+-------------+------------+-----------------+
|    Officer IDs:                                                                                                    |
|    Flat Types:                                                                                                     |
|     - TWO_ROOMS   : $2500000.00 (200 units available)                                                              |
|     - THREE_ROOMS : $3000000.00 (150 units available)                                                              |
+------------+-------------------------+-----------------+-------------+-------------+------------+-----------------+

Would you like to apply a filter? (yes/no): yes
Enter neighbourhood to filter by (or leave blank): TAMPINES
Enter flat type to filter by (e.g., TWO_ROOMS, THREE_ROOMS) or leave blank: TWO_ROOMS

Filtered Projects:
+------------+-------------------------+-----------------+-------------+-------------+------------+-----------------+
|                                                  Project List                                                      |
+------------+-------------------------+-----------------+-------------+-------------+------------+-----------------+
| Project ID | Project Name            | Neighbourhood   | App. Start  | App. End    | Visibility | OfficerSlot     | Manager ID      |
+------------+-------------------------+-----------------+-------------+-------------+------------+-----------------+
| P0003      | THE GREEN HOUSE         | TAMPINES        | 07/28/2025  | 09/28/2025  | ON         | 1               | S0000003M       |
+------------+-------------------------+-----------------+-------------+-------------+------------+-----------------+
|    Officer IDs: S00000040                                                                                          |
|    Flat Types:                                                                                                     |
|     - TWO_ROOMS   : $2000000.00 (150 units available)                                                              |
+------------+-------------------------+-----------------+-------------+-------------+------------+-----------------+
Please enter the Project ID for which you wish to register (press 'x' to exit):
P0003
Registration submitted successfully! Registration ID: R0002
```

Test Case 7: View Registration Status

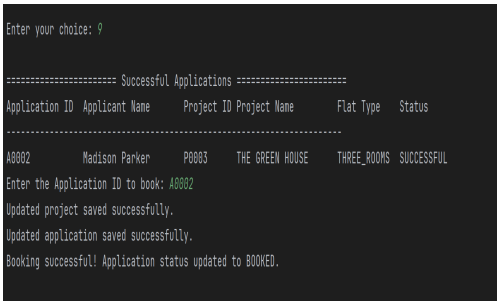


```
Enter your choice: 7
+-----------------------------------------------------+
|                  Registration Record                |
+-----------------------------------------------------+
| Registration ID | Project ID | Registration Status  |
+-----------------------------------------------------+
| R0002           | P0003      | APPROVED             |
+-----------------------------------------------------+
```

Test Case 8: View and reply to Enquiries

```
Enter your choice: 8
+-----------------------------------------------------------------------------------------------------------------+
|                                                  Enquiry List                                                   |
+-------------+--------------+------------+-------------------------------------+-------------------------------+-----------+
| Enquiry ID  | Applicant ID | Project ID | Enquiry                             | Reply                         | Status    |
+-------------+--------------+------------+-------------------------------------+-------------------------------+-----------+
| E0001       | T0800004A    | P0001      | Can I apply again? I missed the     | null                          | PENDING   |
|             |              |            | dateline.                           |                               |           |
+-------------+--------------+------------+-------------------------------------+-------------------------------+-----------+
Do you want to reply to an enquiry? (yes/no): yes
Enter Enquiry ID: E0001
Enter your reply: I am sorry, but the application has closed
Reply submitted successfully.
```

Test Case 9: Book a flat for an applicant



```
Enter your choice: 9


======================== Successful Applications ========================
Application ID  Applicant Name      Project ID Project Name        Flat Type    Status
-----------------------------------------------------------------------------
A0002           Madison Parker      P0003      THE GREEN HOUSE     THREE_ROOMS  SUCCESSFUL
Enter the Application ID to book: A0002
Updated project saved successfully.
Updated application saved successfully.
Booking successful! Application status updated to BOOKED.
```

```
                                        GENERATING FLAT BOOKING RECEIPT
Officer: Natalie Walker (ID: S0000004O)
                                    FLAT BOOKING RECEIPT
APPLICANT INFORMATION:
Name          : Madison Parker      NRIC          : S0000008O
Age           : 35                  Marital Status : MARRIED
PROPERTY DETAILS:
Project ID    : P0003               Project Name   : THE GREEN HOUSE
Neighbourhood : TAMPINES            Flat Type      : THREE_ROOMS
FINANCIAL SUMMARY:
Base Price:                                                    $2500000.00
IMPORTANT INFORMATION:
1. This receipt confirms your flat booking under Application ID: A0002
2. Please keep this receipt for your records.
3. You will be contacted by HDB for the next steps in the purchase process.
4. Failure to make payments by due dates may result in cancellation of your booking.


Receipt generated successfully. The above receipt has been provided to the applicant.
```

4.4 Manager:

Test Case 1: Create, Edit, and Delete BTO Project Listings

Create Project

```
Enter your choice: 1
Auto-generated Project ID: P0004
Enter Project Name: BLUE HOUSE
Enter Neighborhood: LAKESIDE

TWO_ROOMS Flat Type
Enter Number of Units: 100
Enter Price: 1000000
200

THREE_ROOMS Flat Type
Enter Number of Units: Enter Price: 1500000
Enter Opening Date (MM/dd/yyyy): 11/11/2011
✅ Parsed Opening Date: 11/11/2011
Enter Closing Date (MM/dd/yyyy): 11/11/2024
✅ Parsed Closing Date: 11/11/2024
Manager updated with new project successfully.
Project created successfully!
```

Edit Project

```
Would you like to apply a filter? (yes/no): no

===== Update Project Details =====
Enter the Project ID to update: P0004
Updating Project: BLUE HOUSE

What would you like to update?
1. Project Name
2. Two-Room Units
3. Two-Room Price
4. Three-Room Units
5. Three-Room Price
6. Application Opening Date
7. Application Closing Date
8. Visibility
Enter your choice: 1
Enter new Project Name: NEW BLUE HOUSE
Updated project saved successfully.
Project updated successfully!
```

Delete Project

```
Would you like to apply a filter? (yes/no): no

===== Delete Project =====
Enter the Project ID to delete: P0004
Are you sure you want to delete project: NEW BLUE HOUSE? (yes/no)
yes
Project deleted successfully.
```

Test Case 2: Manage Only One Active Project

```
Enter your choice: 1
Auto-generated Project ID: P0005
Enter Project Name: New Project 1
Enter Neighborhood: Nanyang Road

TWO_ROOMS Flat Type
Enter Number of Units: 2
Enter Price: 1000

THREE_ROOMS Flat Type
Enter Number of Units: 3
Enter Price: 1500
Enter Opening Date (MM/dd/yyyy): 11/11/2011
✅ Parsed Opening Date: 11/11/2011
Enter Closing Date (MM/dd/yyyy): 11/11/2025
✅ Parsed Closing Date: 11/11/2025
You are already managing a project with an open application period.
Failed to create project.
```

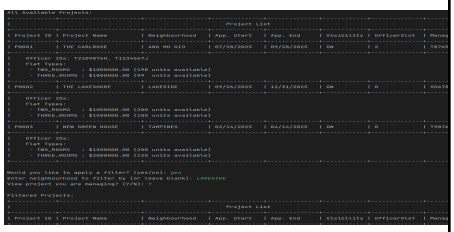
Test Case 3: Toggle Project Visibility

```
Would you like to apply a filter? (yes/no): no

===== Update Project Details =====
Enter the Project ID to update: P0001
Updating Project: THE CARLROSE

What would you like to update?
1. Project Name
2. Two-Room Units
3. Two-Room Price
4. Three-Room Units
5. Three-Room Price
6. Application Opening Date
7. Application Closing Date
8. Visibility
Enter your choice: 8
Current project VisibilityONEnter new Visibility (ON/OFF):
OFF
Updated project saved successfully.
Project updated successfully!
```

Test Case 4: View All & Filter Project



Test Case 5: Generate and Filter Report

```
Enter your choice: 6

+-------------------------------------------------+
|             HDB REPORT GENERATION               |
+-------------------------------------------------+
| 1. Application Status Report                    |
| 2. Booked Applications Report                   |
| 3. Project Summary Report                       |
| 0. Back to Main Menu                            |
+-------------------------------------------------+
Enter your choice: 1

=== APPLICATION STATUS REPORT FILTER OPTIONS ===
1. Filter by Project
2. Filter by Application Status
3. Filter by Withdrawal Status
4. No filter (show all)
Enter your choice: 1
Enter Project ID: P0004

+-----------------------------------------------------------------------------------------------------------+
|                                    APPLICATION STATUS REPORT                                              |
+-----------------------------------------------------------------------------------------------------------+
| Generated on: 2025-04-22 22:03:57                                                                         |
+-----------------------------------------------------------------------------------------------------------+
| APP ID          | APPLICANT       | PROJECT             | FLAT TYPE       | APP STATUS   | WITHDRAWAL   |
+-----------------------------------------------------------------------------------------------------------+
+-----------------------------------------------------------------------------------------------------------+
| Total Applications: 0                                                                                     |
+-----------------------------------------------------------------------------------------------------------+

Generate another report? (Y/N): N

Press Enter to continue...
```

Test Case 6: Manage HDB Officer Registrations

```
Enter your choice: 10
Found project: P0004 - BLUE HOUSE

====== PENDING OFFICER REGISTRATIONS ======
ID  REGISTRATION ID OFFICER NRIC    OFFICER NAME    PROJECT ID
------------------------------------------------------------
1   R0005   S6543210I   Emily   P0004

Enter the ID of the registration to approve/reject (or 0 to cancel): 1
Enter 1 to APPROVE or 0 to REJECT this officer registration: 0
Updated project saved successfully.
Registration rejected.
```

Test Case 7: Approve or Reject BTO Applications and Withdrawals

Approve Applications

```
Enter your choice: 8
Found project: P0001 - THE CARLROSE
Looking for pending withdrawal applications in project: P0001

====== PENDING WITHDRAWAL APPLICATIONS ======
Application ID: A0004
Applicant NRIC: S1111111B
Applicant Name: CK
Withdrawal Status: PENDING
Project ID: P0001
Project Name: THE CARLROSE
Flat Type: TWO_ROOMS
Application Status: PENDING
------------------------------------------------------------

Enter the ID of the application to approve/reject (or 0 to cancel):
A0004
Enter 1 to approve or 0 to reject the withdrawal:
1
Withdrawal approved for Application ID: A0004
Updated application saved successfully.
```

Approve Withdrawals

```
Enter your choice: 7

===== Approve Application =====
Manager: T8765432F
Found project: P0001 - THE CARLROSE
Looking for pending applications in project: P0001

======== PENDING APPLICATIONS ========
+-----------------------------------------------+
| APPLICATION ID  | NRIC         | NAME         | FLAT TYPE    |
+-----------------------------------------------+
| A0002           | T2109876H    | Daniel       | TWO_ROOMS    |
| A0003           | T2109876H    | Daniel       | TWO_ROOMS    |
+-----------------------------------------------+

Enter the ID of the application to approve or reject (or 0 to cancel): A0002
Enter 1 to approve or 0 to reject the application:
1
Updated application saved successfully.
Application approved successfully.
```