NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

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

2024/2025 SEMESTER 2

Build-To-Order (BTO) Management System

https://github.com/stevennoctavianus/SC2002-BTO-Management-System-New

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**
Hoang Viet Thinh	SC2002	SCE3	[illegible] 22 April 2025
Steven Octavianus Rahmat	SC2002	SCE3	[illegible] 22 April 2025
Tan Jia Wei	SC2002	SCE3	21 April 2025
Tan Jing Zhi, Nigel	SC2002	SCE3	[illegible] 21 April 2025
Wee Keng Kit, Wayne	SC2002	SCE3	[illegible] 21 April 2025

1. Design Considerations

1.1 Design Approach: Overall Architecture and Navigation

For our design approach, we build on the Fundamental OOP Principles of Encapsulation, Inheritance, Polymorphism, and Abstraction. These principles provide a flexible and robust foundation that will serve as the basis for building our code. We began by mapping out our UML class diagrams by identifying what functionality each class should have while adhering to the Fundamental OOP Principles as well as the SOLID Design Principles. We thoroughly refined our class diagram to ensure cohesion and proper abstraction of the necessary classes. This gave us a clear and organized overview of the project.

A Boundary-Control-Entity (BCE) Pattern was implemented to our Build-To-Order (BTO) Management System through separation into 3 layers: boundary (interfaces), control (logic) and entities (data). This allows us to enable role-based access control as different roles (e.g. Applicants, HDB Officers, and HDB Managers) have different access and capabilities. As such, it ensures loose coupling to minimise interdependency between classes and high cohesion as each class can focus on a single set of related functions.

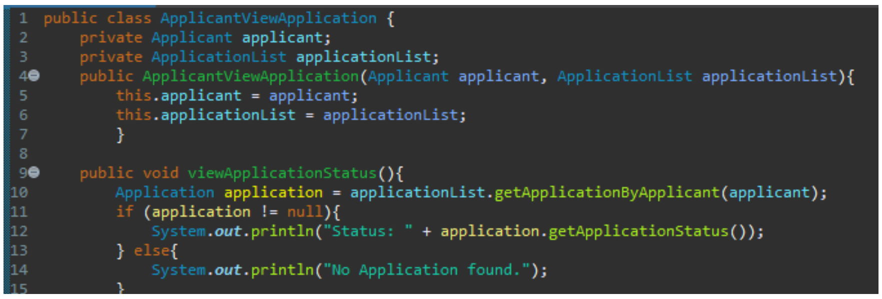
1. Boundary Layer
   The `MainMenu` system serves as the boundary area and implements a hierarchical structure branched to role-specific menus. This allows for users to view menus based on their role after successful login, and as such users are only shown operations and services that are applicable to their role.
2. Control Layer
   The `MainMenu` serves as a central controller to our `OfficerController`, `ApplicantController`, and `ManagerController`, and manages our entities.
3. Entity Layer
   Our entities represent the core data structures and business logic of our BTO Management System, such as `User`, `Enquiry`, `Application`, and others, which represent the core operations of our system.

1.2 Applied Design Principle: Extensibility and Maintainability

To promote modularity and maintainability, we structured our code with **high cohesion and low coupling** by introducing multiple abstraction layers that facilitate interaction between different classes and applying the simple and effective SOLID design principles taught.

1.2.1 Single Responsibility Principle (SRP)

We incorporate SRP into certain classes of our code. These classes perform a single task only to enhance code maintainability, readability, and reusability. For example, the ApplicantViewApplication class is only used when the user wants to view their application status. This makes the code easier to read by clearly separating the responsibilities of each class. It also simplifies maintenance as changes or errors in each function can be isolated without affecting other parts of the code, and the classes can be easily reused when similar functions are required in the program.



```java
public class ApplicantViewApplication {
    private Applicant applicant;
    private ApplicationList applicationList;
    public ApplicantViewApplication(Applicant applicant, ApplicationList applicationList){
        this.applicant = applicant;
        this.applicationList = applicationList;
        }

    public void viewApplicationStatus(){
        Application application = applicationList.getApplicationByApplicant(applicant);
        if (application != null){
            System.out.println("Status: " + application.getApplicationStatus());
        } else{
            System.out.println("No Application found.");
        }
```

Figure 1: SRP Example, `ApplicantViewApplication` *Class*

1.2.2 Open/Closed Principle (OCP)

The Open/Closed Principle states that a class or module should be open for extension but closed for modification. This means that the existing code should not be changed when adding new functionality to the program. Developers can extend existing classes, such as through inheritance or interfaces, to introduce new functions without altering the original implementation. In our case, we created an abstract `User` class that is closed for modification. The flexibility for extension is shown in `Applicant` and `Manager` subclasses, where they inherit the `User` superclass to add new features and for method overriding. Adhering to OCP allows our system to have good extensibility as new user types and roles can easily be implemented without modifying the existing code.

1.2.3 Liskov Substitution Principle (LSP)

The Liskov Substitution Principle suggests that an object of a subclass should be able to replace an object of its superclass without affecting the functionality of the code. For instance, in the PasswordService class, it takes in an object of superclass User type. Essentially, only subclasses (Applicant, Officer, Manager) of the superclass are passed as arguments. These subclasses do not change the expected behavior, hence the changePassword method in the superclass User continues to work as intended, which aligns with LSP.

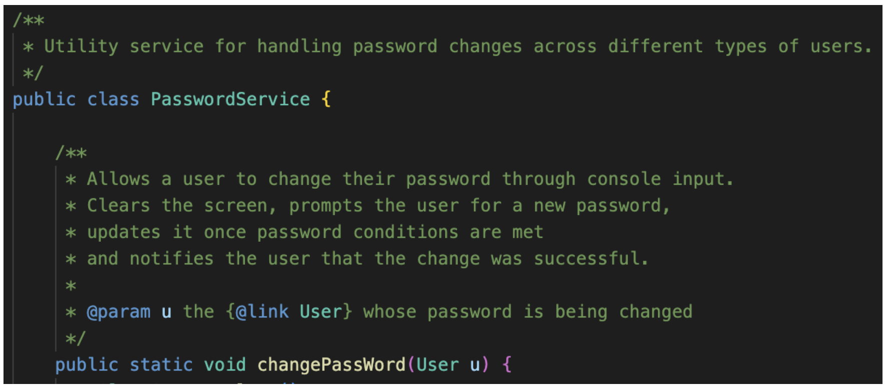

```
/**
 * Utility service for handling password changes across different types of users.
 */
public class PasswordService {

    /**
     * Allows a user to change their password through console input.
     * Clears the screen, prompts the user for a new password,
     * updates it once password conditions are met
     * and notifies the user that the change was successful.
     *
     * @param u the {@link User} whose password is being changed
     */
    public static void changePassWord(User u) {
```

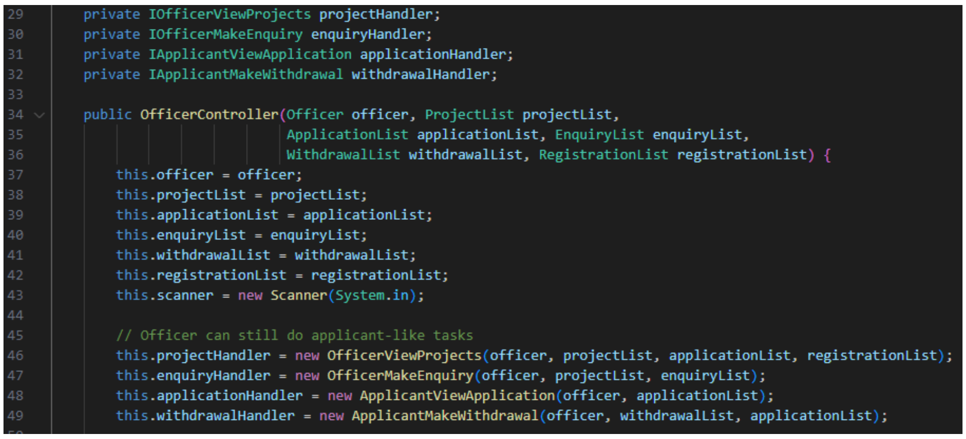
Figure 2. `PasswordService` *Class*

1.2.4 Interface Segregation Principle (ISP)

The Interface Segregation Principle tells us that multiple specific interfaces are better than one general purpose interface. In our project, the interface of each Applicant-related functionality is segregated into its separate interfaces to prevent unintended misuse and to ensure each class only implements what it requires. For example, the `IApplicantMakeEnquiry` interface contains abstract methods related to making an enquiry. When creating the `ApplicantMakeEnquiry` class, it only implements the `IApplicantMakeEnquiry` interface. Other interfaces such as `IApplicantMakeWithdrawal` are not necessary to build the functionality of `ApplicantMakeEnquiry`. This promotes cleaner and more maintainable code by keeping class dependencies minimal.

1.2.5 Dependency Inversion Principle (DIP)

The Dependency Inversion Principle states that high-level modules should not depend on low-level modules, instead both should depend on abstractions. For our project, in the `OfficerController` class, the high-level module does not directly depend on the low-level implementation classes, instead it depends on abstractions (the various interfaces ie. `IOfficerViewProjects`). The actual low-level modules (ie. `projectHandler`) implements its respective interfaces. The controller module assigns an object, `OfficerViewProjects`, to the interface-type reference, projectHandler (line 46), and it will function properly since the object implements the `IOfficerViewProject` interface as well. It does not need to know the details of the low-level modules to function. This aligns with DIP and it allows our code to be more flexible.



```java
    private IOfficerViewProjects projectHandler;
    private IOfficerMakeEnquiry enquiryHandler;
    private IApplicantViewApplication applicationHandler;
    private IApplicantMakeWithdrawal withdrawalHandler;

    public OfficerController(Officer officer, ProjectList projectList,
                             ApplicationList applicationList, EnquiryList enquiryList,
                             WithdrawalList withdrawalList, RegistrationList registrationList) {
        this.officer = officer;
        this.projectList = projectList;
        this.applicationList = applicationList;
        this.enquiryList = enquiryList;
        this.withdrawalList = withdrawalList;
        this.registrationList = registrationList;
        this.scanner = new Scanner(System.in);

        // Officer can still do applicant-like tasks
        this.projectHandler = new OfficerViewProjects(officer, projectList, applicationList, registrationList);
        this.enquiryHandler = new OfficerMakeEnquiry(officer, projectList, enquiryList);
        this.applicationHandler = new ApplicantViewApplication(officer, applicationList);
        this.withdrawalHandler = new ApplicantMakeWithdrawal(officer, withdrawalList, applicationList);
```

Figure 3. `OfficerController` *Class*

1.3 Additional Features

We implemented a few additional features that would help improve system functionality, user experience, and data integrity.

1. "Back" Navigation and UI Improvements (Enhanced CLI User Experience)

   The `BackButton` class allows our code to return to the previous page (Back Navigation) without having to restart the program, ie. the Applicant Dashboard, after viewing the project listings, or return to the Main Menu Login Dashboard after logout. This also aligns with SRP where the BackButton is responsible for doing 1 task only.

   The `ClearScreen` class improves readability and simulates screen transitions when navigating between dashboards.

   The `Colour` class prints messages in distinct colours for better user feedback. For instance, success confirmations are printed in green, improving user interface within the terminal environment.

2. Password Validation

   To ensure user account security, strict password conditions are set. New passwords entered by users must meet the requirements set. After entering a valid password, users will be prompted to re-enter the password again. When both conditions are met, the password is then successfully updated.

3. Shutdown Hook and Crash Recovery Mechanism

   The Shutdown Hook and Uncaught Exception Handler gives our system 3 layers of safety, ensuring data persistence and crash resilience. Whether the user exits normally, accidentally, or the program crashes, user changes and data will be saved.

4. Account Registration for New Applicant

Our system has an added functionality of allowing users who do not have an account to sign up as an applicant only. Through the initial login interface, users can choose to create an account.

2. UML Class Diagram

(Class Diagram is attached and submitted together with this report.)

To reduce coupling, we designed our class diagram to have multiple interfaces to reduce the relationship from association to dependency. This helps our project to be more concise and extendable for future implementations.

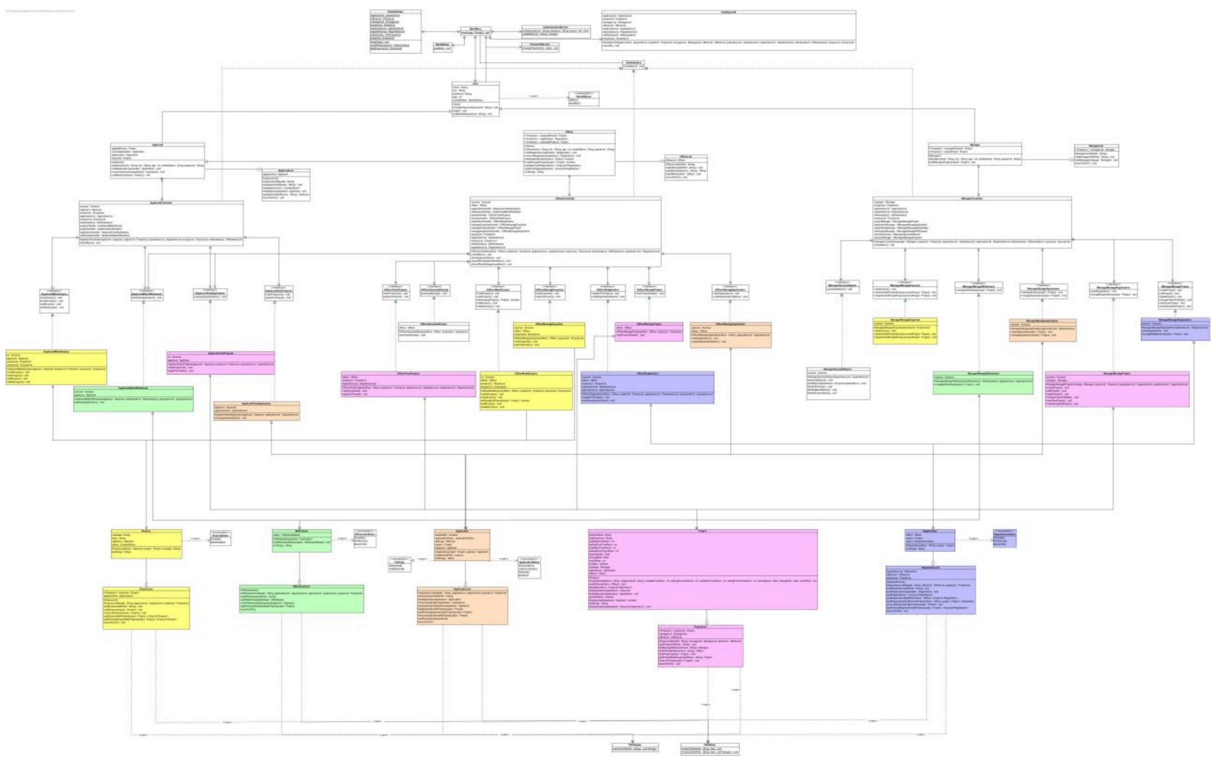


Figure 4. UML Class Diagram

3. UML Sequence Diagram

(Sequence Diagrams are attached and submitted together with this report.)

For simplicity, we assume that the user is already logged in to Officer via the MainMenu class and starts off the Sequence Diagram from the `OfficerController` class onwards. The Sequence Diagram shows the interaction between the group of objects for each use case scenario.

3.1 Officer Applying for a BTO

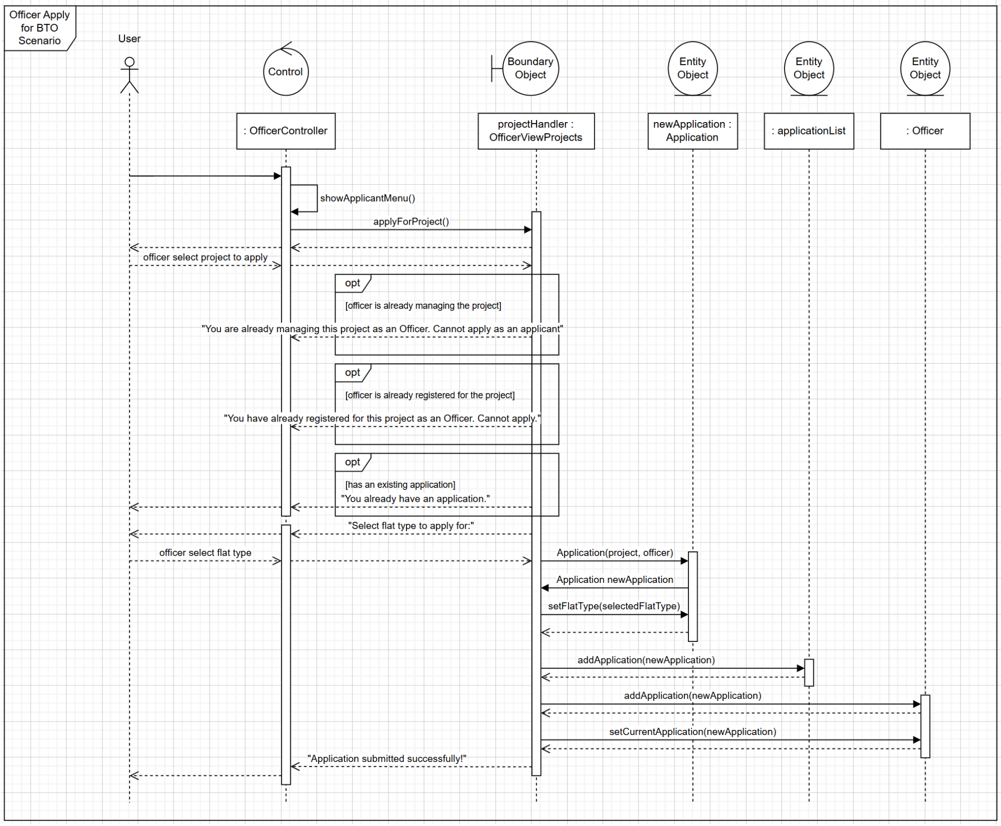

Officer Apply for BTO Scenario
User
Control
Boundary Object
Entity Object
Entity Object
Entity Object
: OfficerController
projectHandler : OfficerViewProjects
newApplication : Application
: applicationList
: Officer
showApplicantMenu()
applyForProject()
officer select project to apply
opt
[officer is already managing the project]
"You are already managing this project as an Officer. Cannot apply as an applicant"
opt
[officer is already registered for the project]
"You have already registered for this project as an Officer. Cannot apply."
opt
[has an existing application]
"You already have an application."
"Select flat type to apply for:"
officer select flat type
Application(project, officer)
Application newApplication
setFlatType(selectedFlatType)
addApplication(newApplication)
addApplication(newApplication)
setCurrentApplication(newApplication)
"Application submitted successfully!"

Figure 5. UML Sequence Diagram for Officer Applying for BTO

3.2 Officer Register to Handle a Project

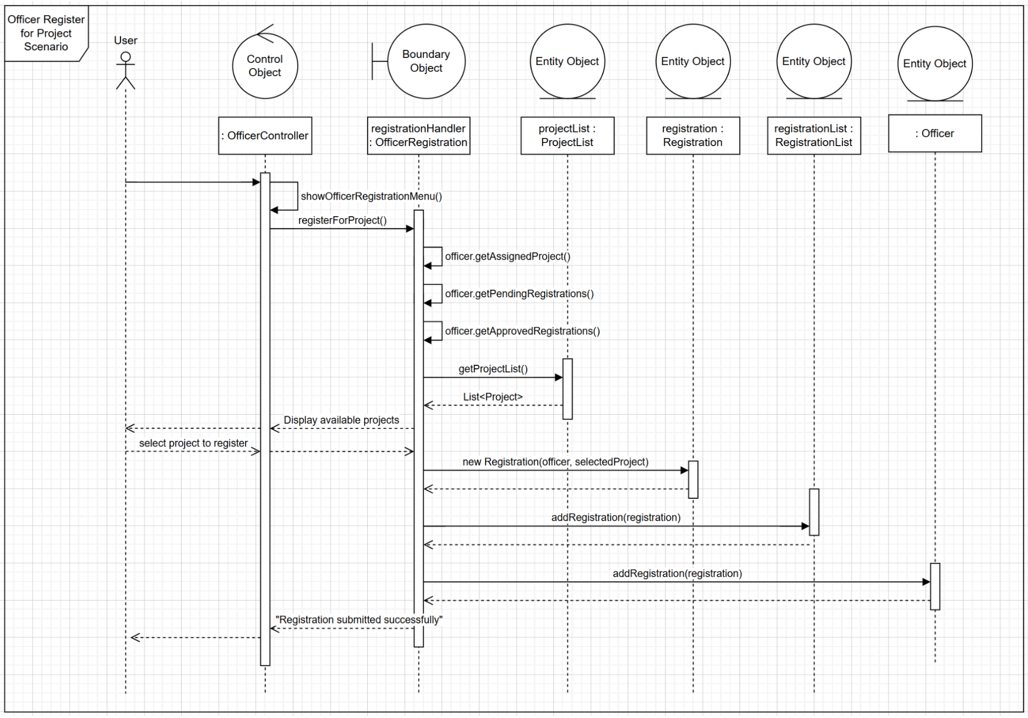
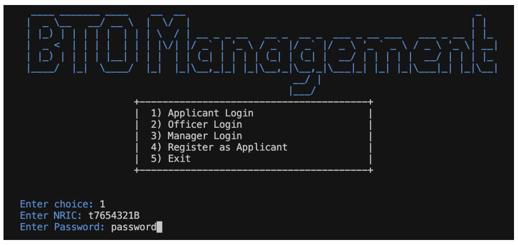

Officer Register for Project Scenario
User
Control Object
Boundary Object
Entity Object
Entity Object
Entity Object
Entity Object
: OfficerController
registrationHandler : OfficerRegistration
projectList : ProjectList
registration : Registration
registrationList : RegistrationList
: Officer
showOfficerRegistrationMenu()
registerForProject()
officer.getAssignedProject()
officer.getPendingRegistrations()
officer.getApprovedRegistrations()
getProjectList()
List<Project>
Display available projects
select project to register
new Registration(officer, selectedProject)
addRegistration(registration)
addRegistration(registration)
"Registration submitted successfully"

Figure 6. UML Sequence Diagram for Officer Registering to Handle a Project

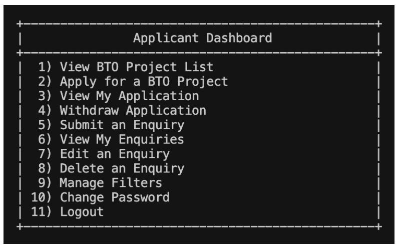
4. Test Cases

4.1 User Login and Password Management

Test Case 1: Valid User Login

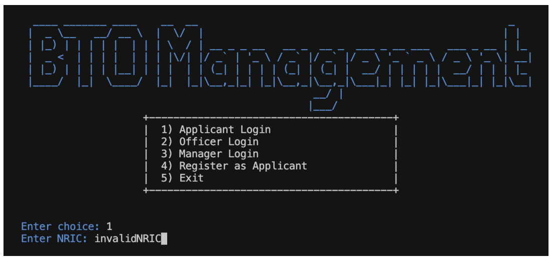


```
+------------------------------------------+
|  1) Applicant Login                      |
|  2) Officer Login                        |
|  3) Manager Login                        |
|  4) Register as Applicant                |
|  5) Exit                                 |
+------------------------------------------+

Enter choice: 1
Enter NRIC: t7654321B
Enter Password: password
```

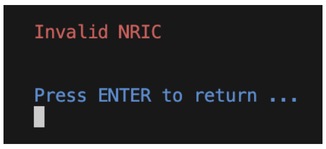


```
+--------------------------------------------------------------+
|                     Applicant Dashboard                      |
+--------------------------------------------------------------+
|  1) View BTO Project List                                    |
|  2) Apply for a BTO Project                                  |
|  3) View My Application                                      |
|  4) Withdraw Application                                     |
|  5) Submit an Enquiry                                        |
|  6) View My Enquiries                                        |
|  7) Edit an Enquiry                                          |
|  8) Delete an Enquiry                                        |
|  9) Manage Filters                                           |
| 10) Change Password                                          |
| 11) Logout                                                   |
+--------------------------------------------------------------+
```

Test Case 2: Invalid NRIC Format

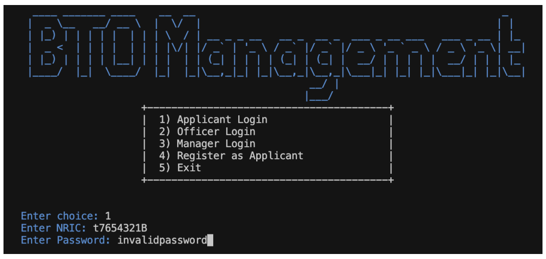


```
+------------------------------------------+
|  1) Applicant Login                      |
|  2) Officer Login                        |
|  3) Manager Login                        |
|  4) Register as Applicant                |
|  5) Exit                                 |
+------------------------------------------+

Enter choice: 1
Enter NRIC: invalidNRIC
```

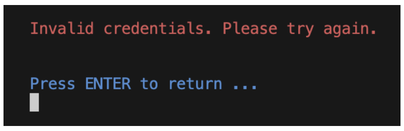


```
Invalid NRIC

Press ENTER to return ...
```

Case 3: Incorrect Password



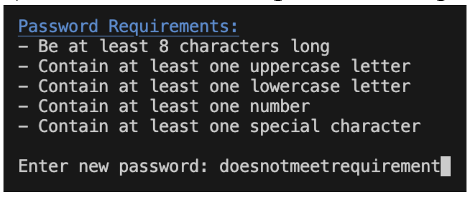
```
+------------------------------------------+
|  1) Applicant Login                      |
|  2) Officer Login                        |
|  3) Manager Login                        |
|  4) Register as Applicant                |
|  5) Exit                                 |
+------------------------------------------+

Enter choice: 1
Enter NRIC: t7654321B
Enter Password: invalidpassword
```



```
Invalid credentials. Please try again.

Press ENTER to return ...
```

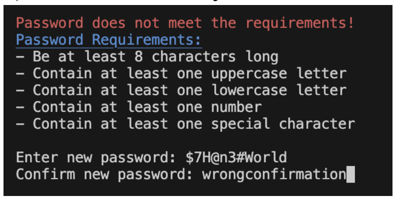
Test Case 4: Password Change Functionality

1) Does not meet the password requirements



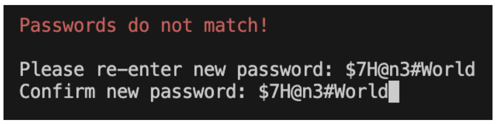
```
Password Requirements:
- Be at least 8 characters long
- Contain at least one uppercase letter
- Contain at least one lowercase letter
- Contain at least one number
- Contain at least one special character

Enter new password: doesnotmeetrequirement
```

2) Is not correctly entered twice

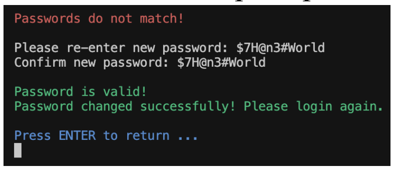


```
Password does not meet the requirements!
Password Requirements:
- Be at least 8 characters long
- Contain at least one uppercase letter
- Contain at least one lowercase letter
- Contain at least one number
- Contain at least one special character

Enter new password: $7H@n3#World
Confirm new password: wrongconfirmation
```

System updates password when both conditions are met, prompts users to re login.



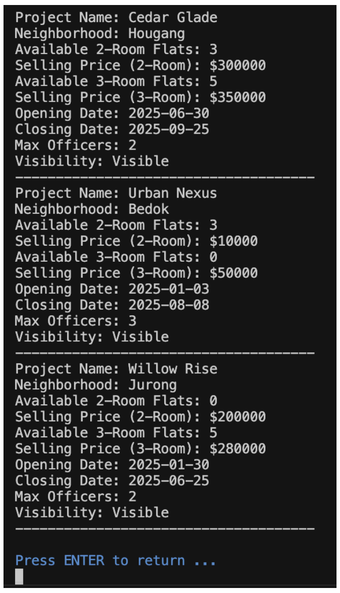
```
Passwords do not match!

Please re-enter new password: $7H@n3#World
Confirm new password: $7H@n3#World
```



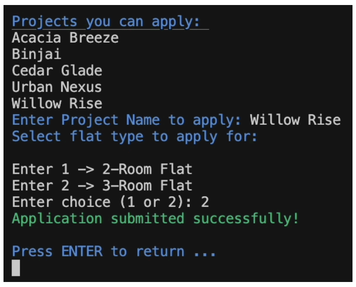
```
Passwords do not match!

Please re-enter new password: $7H@n3#World
Confirm new password: $7H@n3#World

Password is valid!
Password changed successfully! Please login again.

Press ENTER to return ...
```

4.2 Applicant

Test Case 5: Project Visibility Based on User Group and Toggle	Test Case 6: Project Application

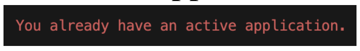


```
Project Name: Cedar Glade
Neighborhood: Hougang
Available 2-Room Flats: 3
Selling Price (2-Room): $300000
Available 3-Room Flats: 5
Selling Price (3-Room): $350000
Opening Date: 2025-06-30
Closing Date: 2025-09-25
Max Officers: 2
Visibility: Visible
----------------------------------------
Project Name: Urban Nexus
Neighborhood: Bedok
Available 2-Room Flats: 3
Selling Price (2-Room): $10000
Available 3-Room Flats: 0
Selling Price (3-Room): $50000
Opening Date: 2025-01-03
Closing Date: 2025-08-08
Max Officers: 3
Visibility: Visible
----------------------------------------
Project Name: Willow Rise
Neighborhood: Jurong
Available 2-Room Flats: 0
Selling Price (2-Room): $200000
Available 3-Room Flats: 5
Selling Price (3-Room): $280000
Opening Date: 2025-01-30
Closing Date: 2025-06-25
Max Officers: 2
Visibility: Visible
----------------------------------------

Press ENTER to return ...
```



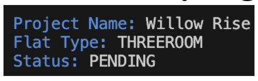
```
Projects you can apply:
Acacia Breeze
Binjai
Cedar Glade
Urban Nexus
Willow Rise
Enter Project Name to apply: Willow Rise
Select flat type to apply for:

Enter 1 -> 2-Room Flat
Enter 2 -> 3-Room Flat
Enter choice (1 or 2): 2
Application submitted successfully!

Press ENTER to return ...
```

Test Case 8: Single Flat Booking per Successful Application



```
You already have an active application.
```

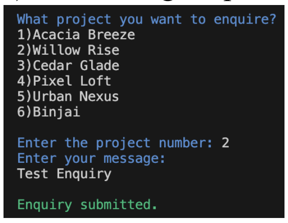
Test Case 7: Viewing Application Status after Visibility Toggle Off



```
Project Name: Willow Rise
Flat Type: THREEROOM
Status: PENDING
```

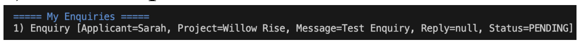
Test Case 9: Applicant's Enquiries Management

1) Submitting enquiries



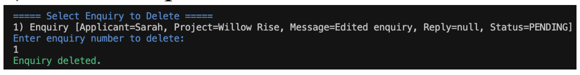
```
What project you want to enquire?
1)Acacia Breeze
2)Willow Rise
3)Cedar Glade
4)Pixel Loft
5)Urban Nexus
6)Binjai

Enter the project number: 2
Enter your message:
Test Enquiry

Enquiry submitted.
```

2) View enquiries

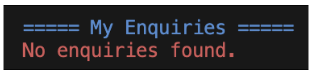


```
===== My Enquiries =====
1) Enquiry [Applicant=Sarah, Project=Willow Rise, Message=Test Enquiry, Reply=null, Status=PENDING]
```

4) Delete enquiries

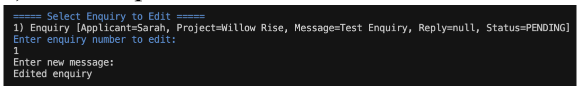


```
===== Select Enquiry to Delete =====
1) Enquiry [Applicant=Sarah, Project=Willow Rise, Message=Edited enquiry, Reply=null, Status=PENDING]
Enter enquiry number to delete:
1
Enquiry deleted.
```

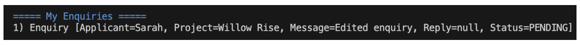


```
===== My Enquiries =====
No enquiries found.
```

3) Edit enquiries



```
===== Select Enquiry to Edit =====
1) Enquiry [Applicant=Sarah, Project=Willow Rise, Message=Test Enquiry, Reply=null, Status=PENDING]
Enter enquiry number to edit:
1
Enter new message:
Edited enquiry
```

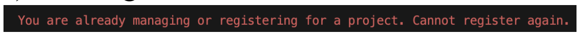


```
===== My Enquiries =====
1) Enquiry [Applicant=Sarah, Project=Willow Rise, Message=Edited enquiry, Reply=null, Status=PENDING]
```

4.3 Officer

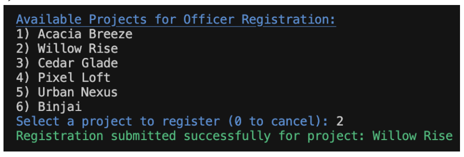
Test Case 10: HDB Officer Registration Eligibility

1) Not eligible



```
You are already managing or registering for a project. Cannot register again.
```

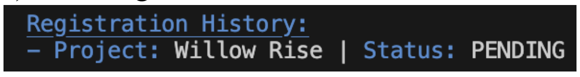
2)



```
Available Projects for Officer Registration:
1) Acacia Breeze
2) Willow Rise
3) Cedar Glade
4) Pixel Loft
5) Urban Nexus
6) Binjai
Select a project to register (0 to cancel): 2
Registration submitted successfully for project: Willow Rise
```

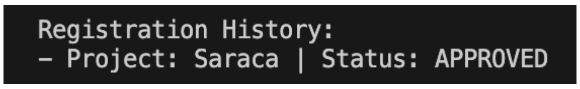
Test Case 11: HDB Officer Registration Status

1) After registration



```
Registration History:
- Project: Willow Rise | Status: PENDING
```

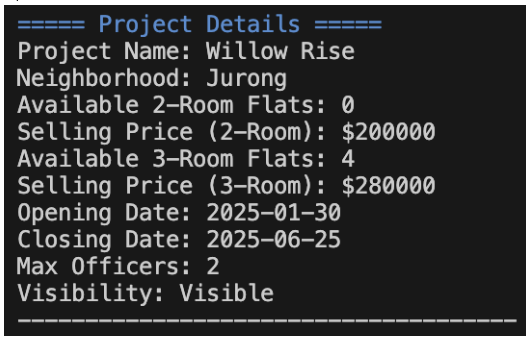
2) After HDB Manager approval



```
Registration History:
- Project: Saraca | Status: APPROVED
```

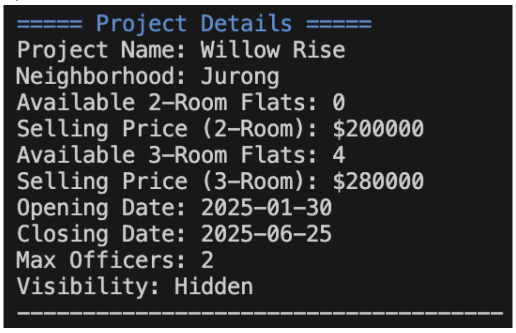
Test Case 12: Project Detail Access for HDB Officer

1) Visible



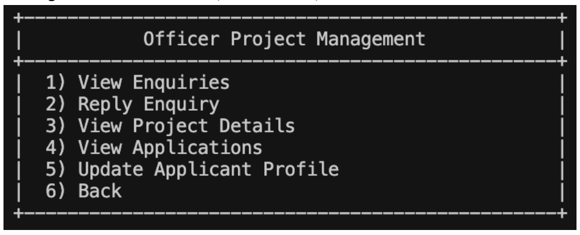
```
===== Project Details =====
Project Name: Willow Rise
Neighborhood: Jurong
Available 2-Room Flats: 0
Selling Price (2-Room): $200000
Available 3-Room Flats: 4
Selling Price (3-Room): $280000
Opening Date: 2025-01-30
Closing Date: 2025-06-25
Max Officers: 2
Visibility: Visible
----------------------------------------
```

2) Not Visible



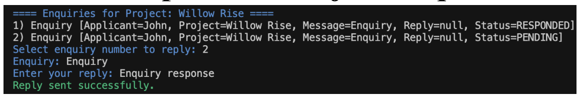
```
===== Project Details =====
Project Name: Willow Rise
Neighborhood: Jurong
Available 2-Room Flats: 0
Selling Price (2-Room): $200000
Available 3-Room Flats: 4
Selling Price (3-Room): $280000
Opening Date: 2025-01-30
Closing Date: 2025-06-25
Max Officers: 2
Visibility: Hidden
----------------------------------------
```

Test Case 13: Restriction on Editing Project Details (Absent)



```
+--------------------------------------------------+
|             Officer Project Management           |
+--------------------------------------------------+
|  1) View Enquiries                               |
|  2) Reply Enquiry                                |
|  3) View Project Details                         |
|  4) View Applications                            |
|  5) Update Applicant Profile                     |
|  6) Back                                         |
+--------------------------------------------------+
```

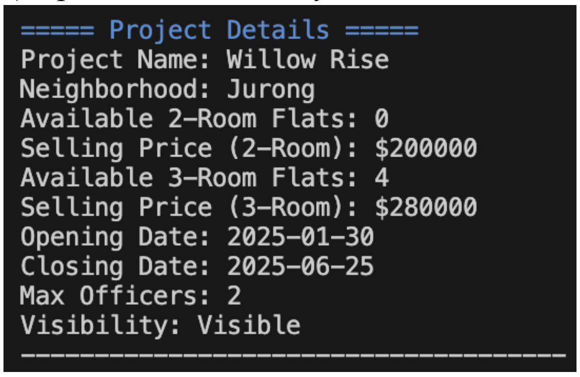
Case 14: Response to Project Enquiries



```
==== Enquiries for Project: Willow Rise ====
1) Enquiry [Applicant=John, Project=Willow Rise, Message=Enquiry, Reply=null, Status=RESPONDED]
2) Enquiry [Applicant=John, Project=Willow Rise, Message=Enquiry, Reply=null, Status=PENDING]
Select enquiry number to reply: 2
Enquiry: Enquiry
Enter your reply: Enquiry response
Reply sent successfully.
```

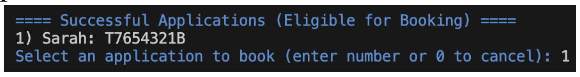
Test Case 15: Flat Selection and Booking Management

1) Update flat availability



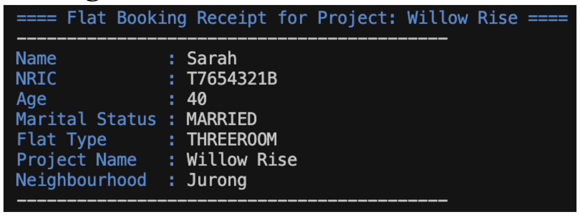
```
===== Project Details =====
Project Name: Willow Rise
Neighborhood: Jurong
Available 2-Room Flats: 0
Selling Price (2-Room): $200000
Available 3-Room Flats: 4
Selling Price (3-Room): $280000
Opening Date: 2025-01-30
Closing Date: 2025-06-25
Max Officers: 2
Visibility: Visible
----------------------------------------
```

2) Log booking details in the applicant's profile



```
==== Successful Applications (Eligible for Booking) ====
1) Sarah: T7654321B
Select an application to book (enter number or 0 to cancel): 1
```

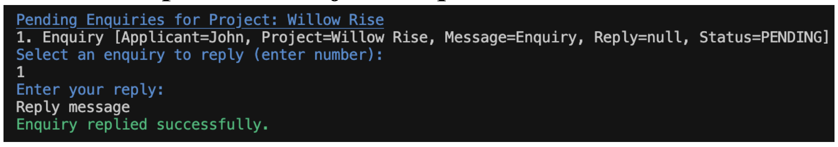
Case 16: Receipt Generation for Booking



```
==== Flat Booking Receipt for Project: Willow Rise ====
-----------------------------------------------------
Name           : Sarah
NRIC           : T7654321B
Age            : 40
Marital Status : MARRIED
Flat Type      : THREEROOM
Project Name   : Willow Rise
Neighbourhood  : Jurong
-----------------------------------------------------
```

4.4 Manager

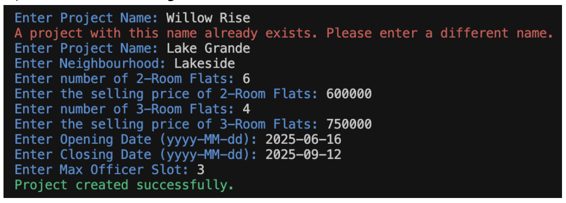
Case 14: Response to Project Enquiries



```
Pending Enquiries for Project: Willow Rise
1. Enquiry [Applicant=John, Project=Willow Rise, Message=Enquiry, Reply=null, Status=PENDING]
Select an enquiry to reply (enter number):
1
Enter your reply:
Reply message
Enquiry replied successfully.
```

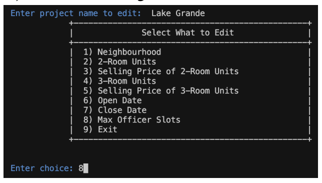
Test Case 17: Create, Edit, and Delete BTO Project Listings

1) Create Project

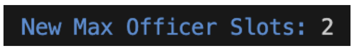


```
Enter Project Name: Willow Rise
A project with this name already exists. Please enter a different name.
Enter Project Name: Lake Grande
Enter Neighbourhood: Lakeside
Enter number of 2-Room Flats: 6
Enter the selling price of 2-Room Flats: 600000
Enter number of 3-Room Flats: 4
Enter the selling price of 3-Room Flats: 750000
Enter Opening Date (yyyy-MM-dd): 2025-06-16
Enter Closing Date (yyyy-MM-dd): 2025-09-12
Enter Max Officer Slot: 3
Project created successfully.
```

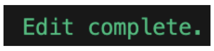
2) Edit Project



```
Enter project name to edit:  Lake Grande
            +------------------------------------------+
            |              Select What to Edit          |
            +------------------------------------------+
            |  1) Neighbourhood                         |
            |  2) 2-Room Units                          |
            |  3) Selling Price of 2-Room Units         |
            |  4) 3-Room Units                          |
            |  5) Selling Price of 3-Room Units         |
            |  6) Open Date                             |
            |  7) Close Date                            |
            |  8) Max Officer Slots                     |
            |  9) Exit                                  |
            +------------------------------------------+


Enter choice: 8
```

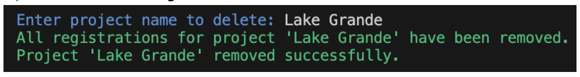


```
New Max Officer Slots: 2
```



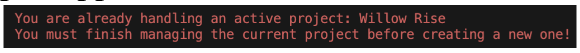
```
Edit complete.
```

3) Delete Project



```
Enter project name to delete: Lake Grande
All registrations for project 'Lake Grande' have been removed.
Project 'Lake Grande' removed successfully.
```

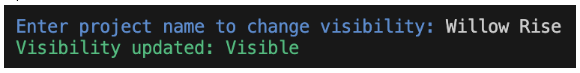
Test Case 18: Single Project Management per Application Period



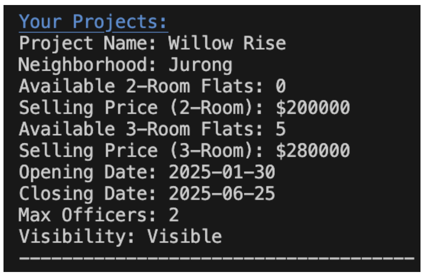
```
You are already handling an active project: Willow Rise
You must finish managing the current project before creating a new one!
```

Test Case 19: Toggle Project Visibility

1) Visible

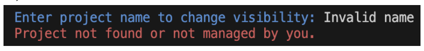


```
Enter project name to change visibility: Willow Rise
Visibility updated: Visible
```

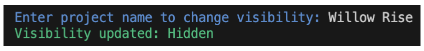


```
Your Projects:
Project Name: Willow Rise
Neighborhood: Jurong
Available 2-Room Flats: 0
Selling Price (2-Room): $200000
Available 3-Room Flats: 5
Selling Price (3-Room): $280000
Opening Date: 2025-01-30
Closing Date: 2025-06-25
Max Officers: 2
Visibility: Visible
------------------------------------
```

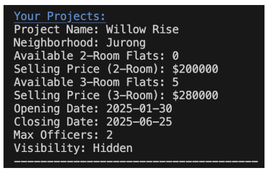
2) Hidden



```
Enter project name to change visibility: Invalid name
Project not found or not managed by you.
```

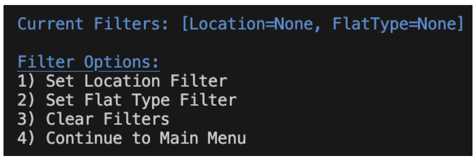


```
Enter project name to change visibility: Willow Rise
Visibility updated: Hidden
```

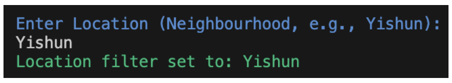


```
Your Projects:
Project Name: Willow Rise
Neighborhood: Jurong
Available 2-Room Flats: 0
Selling Price (2-Room): $200000
Available 3-Room Flats: 5
Selling Price (3-Room): $280000
Opening Date: 2025-01-30
Closing Date: 2025-06-25
Max Officers: 2
Visibility: Hidden
------------------------------------
```

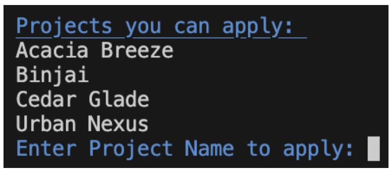
Test Case 20: View All and Filtered Project Listings



```
Current Filters: [Location=None, FlatType=None]

Filter Options:
1) Set Location Filter
2) Set Flat Type Filter
3) Clear Filters
4) Continue to Main Menu
```

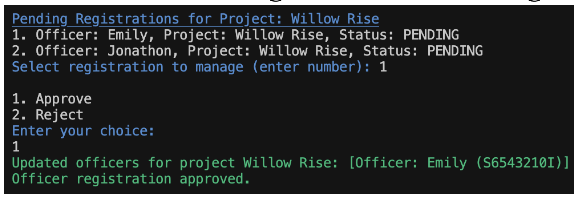


```
Enter Location (Neighbourhood, e.g., Yishun):
Yishun
Location filter set to: Yishun
```



```
Projects you can apply:
Acacia Breeze
Binjai
Cedar Glade
Urban Nexus
Enter Project Name to apply:
```

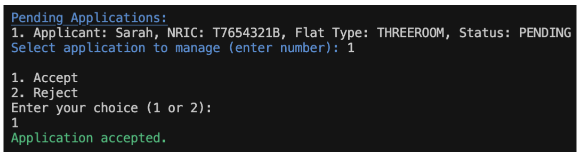
Test Case 21: Manage HDB Officer Registrations



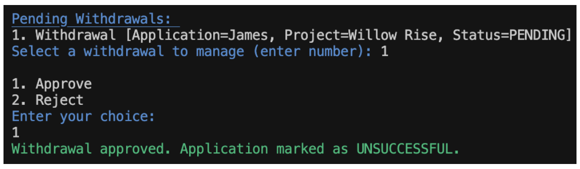
```
Pending Registrations for Project: Willow Rise
1. Officer: Emily, Project: Willow Rise, Status: PENDING
2. Officer: Jonathon, Project: Willow Rise, Status: PENDING
Select registration to manage (enter number): 1

1. Approve
2. Reject
Enter your choice:
1
Updated officers for project Willow Rise: [Officer: Emily (S6543210I)]
Officer registration approved.
```

Test Case 22: Approve or Reject BTO Applications and Withdrawals

1) Managing Applications	2) Managing Withdrawals

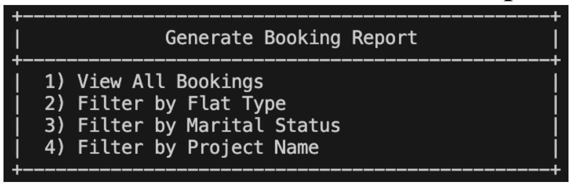


```
Pending Applications:
1. Applicant: Sarah, NRIC: T7654321B, Flat Type: THREEROOM, Status: PENDING
Select application to manage (enter number): 1

1. Accept
2. Reject
Enter your choice (1 or 2):
1
Application accepted.
```

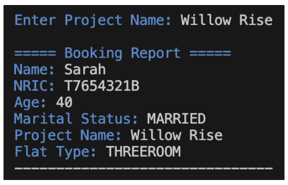


```
Pending Withdrawals:
1. Withdrawal [Application=James, Project=Willow Rise, Status=PENDING]
Select a withdrawal to manage (enter number): 1

1. Approve
2. Reject
Enter your choice:
1
Withdrawal approved. Application marked as UNSUCCESSFUL.
```

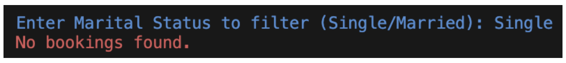
Test Case 23: Generate and Filter Reports



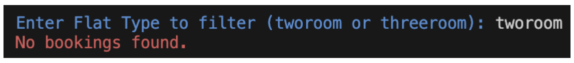
```
+--------------------------------------------------+
|               Generate Booking Report            |
+--------------------------------------------------+
|  1) View All Bookings                            |
|  2) Filter by Flat Type                          |
|  3) Filter by Marital Status                     |
|  4) Filter by Project Name                       |
+--------------------------------------------------+
```



```
Enter Project Name: Willow Rise

===== Booking Report =====
Name: Sarah
NRIC: T7654321B
Age: 40
Marital Status: MARRIED
Project Name: Willow Rise
Flat Type: THREEROOM
-----------------------------
```



```
Enter Marital Status to filter (Single/Married): Single
No bookings found.
```



```
Enter Flat Type to filter (tworoom or threeroom): tworoom
No bookings found.
```

5. Reflection

During the initial drafting of the UML Class Diagram, we encountered challenges in structuring the classes efficiently. Our first few drafts revealed redundancies and design issues that did not align well with what we required. While going through multiple iterations of the Class Diagrams, we identified overlapping functionalities and consolidated similar responsibilities into single, reusable classes. This helped promote code reusability and modularity, aligning with the key principles of OOP.

We also practiced loose coupling wherever possible to minimize the dependencies between each class, making the system more flexible and easier to maintain. While developing the code, we also had to go through a self-learning process to read/write the data from the CSV files (Applicant, Manager, Officer and Project Lists), and how to retain the data after initializing and exiting the system, so that data such as the user's password and enquiries made for certain projects are saved properly for the next system initializing.

5.1 Future Improvement Suggestions

Given more time, we see several areas for improvement. For instance, we will require users to change their password on their first login to ensure that users establish their own unique, strong credentials. Additionally, we would implement a Multi-Factor Authentication, password hashing, and limit the number of login attempts before an account is temporarily locked out during login.This improves data security by preventing unauthorized access and mitigates risks associated with password breaches.