




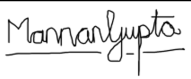
## SC2002 - OBJECT ORIENTED DESIGN AND PROGRAMMING

### 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
Eben Lim	SC2002	SCE3	 20/04/2025
Goh Yi Xiang	SC2002	SCE3	 20/04/2025
Mun Jeongwon	SC2002	SCE3	 20/04/2025
Nicholas Hendra Hartimin	SC2002	SCE3	 20/04/2025
Mannan Gupta	SC2002	SCE3	 20/04/2025

## **0. Problem Analysis**

Before getting started, we parsed the assignment's design requirements line-by-line to identify the user roles: Applicant, HDB Manager and Officer as well as each role's capabilities and responsibilities. Afterwards, we brainstormed possible classes and how we would establish their relationship with one another. We will then further analyze the requirement by looking for key words to identify what was necessary and what was not before we begin to craft our design prototype.

## **1. Design Considerations**

### **1.1 Approach**

The BTO Management System is a system for applicants, HDB officers and HDB managers to view, apply and manage BTO projects. The system follows a layered design, where packages are split into two types: high and low level. Interaction between the user and the system is facilitated through controller classes and as for interaction between classes, interfaces were deployed. All in all, our system achieves loose coupling and tight cohesion, making it easy to maintain and modify.

Additionally, for BTO projects, enquiries, and applications, we decided to store user-related information through their ID or NRIC instead of directly storing the user object instance. This helps promote a more memory-efficient program, which will be essential as the database continues to grow. Despite this, this comes at a price of time as the program needs to continuously search for information through the database. Even so, we believe that this model has a higher opportunity for improvement given the large span of time-efficient search algorithms available.

### **1.2 Assumptions**

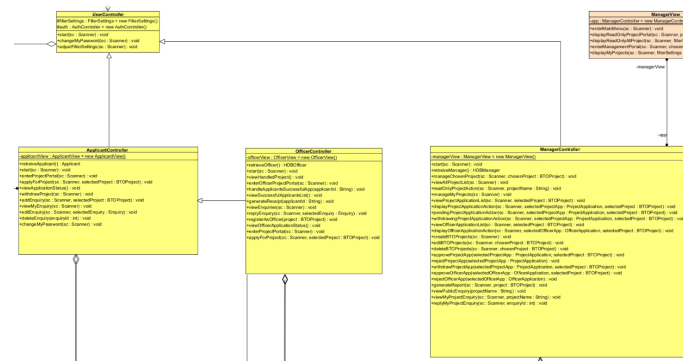
1. HDB officers can register as an officer anytime within the project application period, regardless of whether it's before or after project opening date.
2. All projects have 2 room and 3 room flats.
3. When a HDB manager creates a new project, he/she will be automatically assigned to it. Hence if they try to create a new project with an application period that overlaps with his current assigned one, he would not be able to do so.
4. The deletion of a project during its active period is not allowed. If it is deleted after the application period, all the associated applications will be completely removed regardless of any of the application status.

## 1.3 Design Principles

### 1.3.1 SOLID Design Principle

#### Single Responsibility Principle

The single responsibility principle states that each class should only have one responsibility, keeping the system modular and easier to maintain. For instance, we split the controller classes up according to their respective role. ManagerController for the HDB managers, OfficerController for HDB officers and ApplicantController for applicants. This separation ensures that changes in one logic does not affect the flow of others.



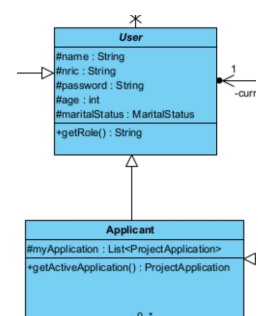
Although this increased the number of classes, we believe that the improvement on the maintainability and testability outweighs the complexity cost.

#### Open-Close Principle

The open-close principle states that a module should be open for extension but closed for modification. We implemented this principle in our system through the use of interfaces. For instance, within the project, the use of interfaces for all of our services allow new functionalities or changes to be added without having to modify existing classes. This enabled easy extension without modifying core controller logic. Yet again, this increased the total number of classes but we accepted the extra boilerplate to keep our core modes closed for modification.

#### Liskov Substitution Principle

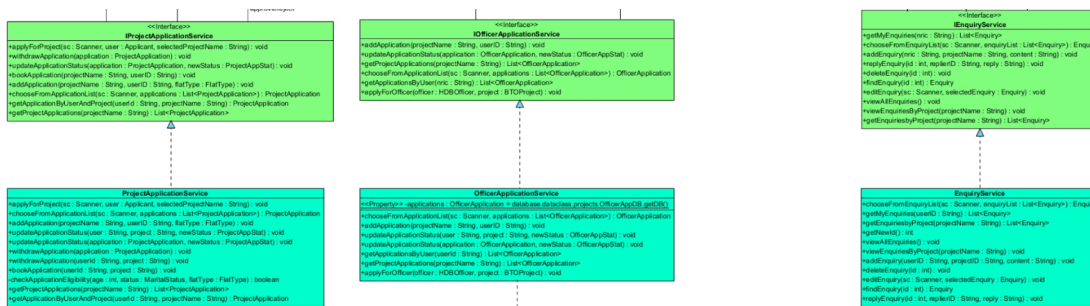
The Liskov substitution principle states that subclasses must perform all the functionalities that their superclasses promise. For instance, anywhere the superclass User is used it should be able to be replaced with Applicant class,



the subclass that inherits User, without disrupting the behavior of the program. This enables flexibility and encourages a reusable design.

## Interface Segregation Principle

The interface segregation principle states that larger interfaces should be separated into smaller, more focused ones. For instance, IEnquiryService interface only focuses on replying enquiries, IOfficerApplicationService only focuses on managing officer registrations and IProjectApplicationService only focuses on the project applications. Particularly, IOfficerApplicationService and IProjectApplicationService were initially created under the same interface IApplicationService. However, due to the conflicting method signatures and different services among the two, it was segregated to cater for the differences. This ensures that the concrete classes implement methods that they actually need and are not forced to depend on methods that they do not use. Ultimately, it improves readability and produces more flexible codes.



## Dependency Injection Principle

The dependency injection principle states that high level modules should not depend on low level modules, instead both should depend on abstractions. Our controllers depend on abstractions and not direct concrete logic. For instance, the ManagerController class calls methods on IProjectApplicationService but not directly on ProjectApplicationService. This way, the high-level module (the controller) and the low-level modules (service class) are not tightly coupled together, but still maintain the cohesion between the two as well as the program extensibility (fulfilling OCP). Take the IReceiptPrintService, for example, if a developer wants to change the formatting or the content arrangement of the receipt, they can directly extend the new class from the interface, without affecting the higher-level module flow (controller) due to the weak coupling. This shows how the program promotes flexibility, maintainability and testability.

### 1.3.2 Additional Design Principles

## Enumeration-Based State Tracking

Enumerations were used in our code, rather than raw strings or integers, which improved readability and made debugging easier. This is because enumerations are type-safe and make switch based logic clear and exhaustive as compile time.

```
package models.enums;

public enum FilterCategory {
    PROJECT_ID,
    PROJECT_NAME,
    OPENING_DATE,
    CLOSING_DATE,
    NEIGHBORHOOD,
    FLAT_TYPE,
}
```

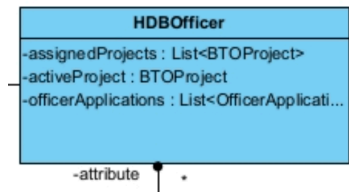
## 1.4 Object-Oriented Programming Principles

### Abstraction

Abstraction was applied in order to simplify interactions between components and hide implementation details, through the use of interfaces. For instance, `IProjectApplicationService`, `IOfficerApplicationService` and `IEnquiryService` are used to allow the `ManagerController` to interact with the system at a high level of abstraction, making the code more maintainable and extendable. This also ensures that the `ManagerController` focuses on its own logic rather than on all the intricate details it does not need to know.

### Encapsulation and Information Hiding

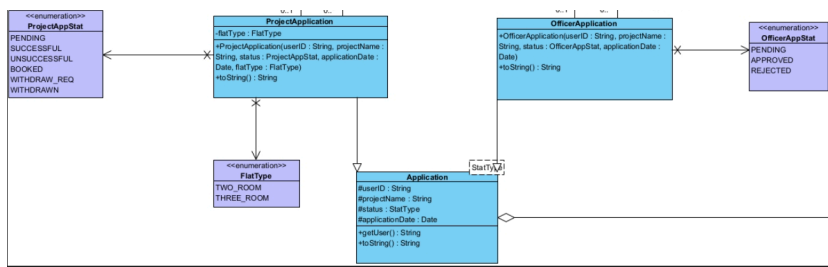
Our BTO Management System demonstrates encapsulation and information hiding to ensure that sensitive attributes cannot be directly accessed or modified. For instance, attributes in the `HDBOfficer` class such as `assignedProjects`, `activeProject` and `officerApplications` are declared as private. Access to these attributes can only be done through public getter and setter methods.



Although there will be more boilerplate due to the getter and setter methods, this design promotes data integrity and is important for handling sensitive data as it prevents unintended data leakage.

### Inheritance

We used inheritance to promote reusability, generalisation and specialisation. For instance, the `Application` class is designed as a generic superclass while `OfficerApplication` and `ProjectApplication` are subclasses, with their own application methods for their respective roles.



This design allows the system to treat both application types uniformly where appropriate, while also supporting specific logic and statuses unique to each. It also promotes extendability. If a new type of application is required, it can just be inserted as a subclass of the Application superclass with minimal duplication.

## Polymorphism

For polymorphism, we enabled different types of objects to be used interchangeably through a shared interface or superclass. For instance, the superclass UserController has subclasses ApplicantController, OfficerController and ManagerController which extends from it. All three subclasses have method overriding for the changeMyPassword method as they all need the function to change password. The concept of polymorphism ensures that a common method does not have to be repeated, hence improving code readability. Since polymorphism can obscure control flow if overused, we limited its implementation to clear, well-documented extension points.

## 1.5 Additional Features

### Registration of new applicant

To support self-service account creation, a feature for new applicant registration was implemented in *AuthController.java*. The method `addApplicant(Scanner sc)` allows users with no account to register by providing their NRIC, name, password, age, and marital status.

The system validates NRIC format and checks for duplicates before creating a new Applicant object and saving it to ApplicantDB. This feature improves usability by allowing new users to join the system without administrator intervention. It also improves the overall completeness of the system by supporting self-service account creation, which is a common and expected functionality in real-world applications.

```

=====
BTO Project Management System
=====
1. Log In
2. Register as Applicant
3. Exit App
=====
Enter your choice (1-3): 2
=====
Enter NRIC:
T2468101L
Enter Name:
Thor
Enter Password:
password
Enter Age:
34
Enter Marital Status (single/married):
Married
Applicant added successfully!
=====
  
```

## 2. UML Class Diagram

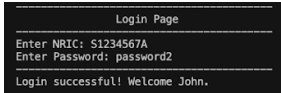
The UML Class Diagram can be found in the GitHub repository. We have divided it into 4 portions, the overall class diagram, one for all the entities, one for view and others for the read-write utility. The reason that the class diagrams are not combined into one diagram is to avoid unnecessary cluttering and messy diagrams, which may have made the diagram even more confusing instead of enhancing its clarity. Therefore through the overall class diagram, readers can focus on the important association or relationship, so they can understand the bigger picture before going onto more details. This thought process utilizes Abstraction, one of the computational thinking skills, to ensure a readable and neat class diagram.

## 3. UML Sequence Diagram

The UML Sequence Diagram can also be found in the GitHub repository. The first sequence diagram shows the overall flow of the HDB officer's role while the second one zooms in on the officer registration. The reason we showcase the overall flow of HDB Officer is because it is the most complex role as it can act as Applicant and Officer yet the eligibility of project differs from that of normal applicant. This also leads us further to the 'zoom-in' version of the application (both processes to clearly show how we decide and filter out. Lastly, we have created another sequence diagram to showcase the Report Generator feature and our new feature of New User Registration

## 4. Test Case Demonstration

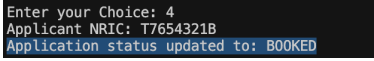
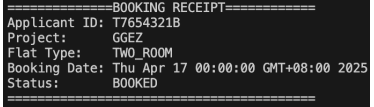
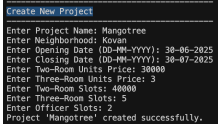
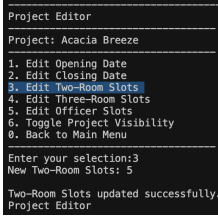
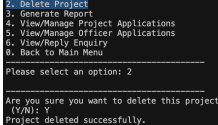
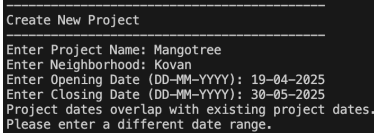
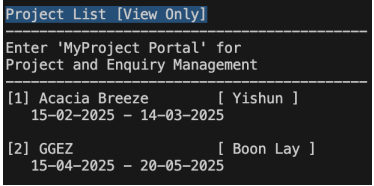

For testing, we used manual functional testing due to the nature of the BTO system which is a CLI, file-driven application.

Test Case	Description	Expected Behaviour	Outcome
User			
1	Successful login:	System allows user to access dashboard based on their roles	

2	Unsuccessful login: a. Invalid NRIC format b. Wrong password	a. System prompts user to re-enter NRIC b. System deny access and prompts user to try again once attempt limit is reached	<p>a.</p> <pre> Login Page Enter NRIC: S12345678A Invalid NRIC format, Please try again. Login Page </pre> <p>b.</p> <pre> Enter NRIC: S1234567A Enter Password: PASSWORD Invalid NRIC or password. Please try again. You have reached your attempt limits! Please try again later </pre>
3	Change Password:	System updates changed password and prompts user to re-login	<pre> Change Your Password Enter Your Old Password: password2 Enter Your New Password: password1 You have logged out successfully. </pre> <pre> Enter NRIC: S1234567A Enter Password: password1 Login successful! Welcome John. </pre>
<b>Applicant</b>			
1	View project list: a. Visibility toggle on b. Visibility toggle off	a. Projects are visible b. Projects are not visible	<p>a.</p> <pre> 1. Project Name: GGEZ Neighborhood: Boon Lay 2-Room Units: 2 (Price: \$350000.0) 3-Room Units: 3 (Price: \$450000.0) Application Opening Date: Tue Apr 15 00:00:00 GMT+08:00 2025 Application Closing Date: Tue May 20 00:00:00 GMT+08:00 2025 Manager: Jessica Officer Slot: 3 Officers: </pre> <p>b.</p> <pre> No projects available based on the current filter settings. </pre>
2	Apply for project: a. Relevant to applicant's group b. Not relevant to applicant's group	a. Project successfully applied b. Project cannot be applied	<p>a.</p> <pre> 1. Apply for this project 2. Ask questions about this project 0. Back to project list Enter your choice: 1 Selected Project: GGEZ Choose room type: 1. 2-Room 2. 3-Room [UNELIGIBLE] 3. Cancel Enter your choice: 1 You have successfully applied for GGEZ. </pre> <p>b.</p> <pre> Choose room type: 1. 2-Room [UNELIGIBLE] 2. 3-Room [UNELIGIBLE] 3. Cancel Enter your choice: 1 You are not Eligible! </pre>
3	View application status: a. Visibility toggle on or off	a. Application status visible	<pre> Enter your choice: 3 Your Applications: &lt;Current Application&gt; ProjectApplication[User=S1234567A, Project= Apr 17 00:00:00 GMT+08:00 2025] </pre>
4	Withdraw application:	Withdraw request sent and application successfully withdrawn	<pre> Are you sure you want to withdraw this application? (yes/no) yes Application status updated to: WITHDRAW_REQ Application withdrawn successfully. </pre>
5	Manage enquiries:	Enquiries can be successfully submitted, displayed, modified and removed	<pre> Enquiries: 1. User =S1234567A [2025-03-07 13:22] Project=Acacia Breeze How many HDB slot are there? Replier =T2109876H [2025-03-07 20:02] There are 2 HDB slot </pre>



HDB Officer			
1	Register for project as an HDB officer: a. Register for one project b. Register for more than one project c. Register for project that the HDB officer has applied for as an applicant	a. Registration is successful b. Registration is unsuccessful c. Registration is unsuccessful	<p>a.</p> <pre>You have selected: GGEZ 1. Register as officer for this project 2. Back to project list Enter your choice: 1 Registration PENDING approval.</pre> <p>b.</p> <pre>You have selected: Acacia Breeze 1. Register as officer for this project 2. Back to project list Enter your choice: 1 This project overlaps with your current application. Current application: GGEZ   Status: PENDING</pre> <p>c.</p> <pre>Cannot register as an officer for "GGEZ" because you've already applied for it as an applicant.</pre>
2	View registration status:	Pending or approved registration status can be viewed	<pre>Your Applications: &lt;Current Application&gt; OfficerApplication[User=T2109876H, Project=G Apr 17 00:00:00 GMT+08:00 2025]</pre>
3	Apply for project as applicant: a. Apply for project that the HDB officer is not handling b. Apply for project that the HDB officer is handling	a. Application is successful b. Application is unsuccessful	<p>a.</p> <pre>You have selected: GGEZ 1. Apply for this project 2. Ask questions about this project 3. Back to project list Enter your choice: 1 Selected Project: GGEZ Choose room type: 1. 2-Room 2. 3-Room [INELIGIBLE] 3. Cancel Enter your choice: 1 You have successfully applied for GGEZ.</pre> <p>b.</p> <pre>You have already applied as an officer for this project. Please check your application status.</pre>
4	View application status:	Pending or approved application status can be viewed	<pre>Enter your choice: 3 Your Applications: &lt;Current Application&gt; ProjectApplication[User=T2109876H, Project=Mangotree, Status=PENDING, flatType=TWO_ROOM, ApplicationDate=Sun Apr 20 00:52:19 GMT+08:00 2025]</pre>
5	View project details: a. Visibility toggle is on/off	a. Full project details are visible	<pre>Enter your Choice: 2 Project Name: GGEZ Neighborhood: Boon Lay 2-Room Units: 2 (Price: \$350000.0) 3-Room Units: 3 (Price: \$450000.0) Application Opening Date: Tue Apr 15 00:00:00 GMT+08:00 2025 Application Closing Date: Tue May 20 00:00:00 GMT+08:00 2025 Manager: S5678901G Officer Slot: 3 Officers: S6543210I</pre>
6	Edit project details:	Option to edit projects is not provided	
7	Manage enquiries:	Enquiries can be successfully accessed and responded to	<pre>Enquiries: 1. User =T7654321B [2025-04-18 15:54] Project=GGEZ How many slot are there? No Reply 0. Back to menu Please select an enquiry (1 - 1): 1 This enquiry has not been replied to yet. 1. Reply to this enquiry 2. Back to menu Enter your choice: 1 Enter your reply content: 3! Reply sent successfully.</pre>
8	Retrieve applicant's BTO application with applicant's NRIC:	Successfully retrieved the correct application	<pre>Enter your Choice: 3 userID='T7654321B' projectName='GGEZ' flatType=TWO_ROOM status=SUCCESSFUL applicationDate=17-04-2025</pre>

9	Update number of flats remaining:	Upon successful booking, remaining number of flats are updated in the CSV file	
10	Update applicant's project status from successful to booked:	Status updated correctly	 <pre> Enter your Choice: 4 Applicant NRIC: T7654321B Application status updated to: BOOKED </pre>
11	Update applicant's profile:	Upon successful booking, applicant's profile is updated in the CSV file	
12	Generate receipt for flat booking:	Receipts generated for each successful booking	 <pre> =====BOOKING RECEIPT===== Applicant ID: T7654321B Project: GGEZ Flat Type: TWO_ROOM Booking Date: Thu Apr 17 00:00:00 GMT+08:00 2025 Status: BOOKED ===== </pre>
<b>HDB Manager</b>			
1	Create, edit and delete BTO projects: a. Create project b. Edit project c. Delete project	a. New BTO project successfully created and saved b. Projects can be edited and changes are saved c. Projects can be deleted	a.  <pre> Create New Project Enter Project Name: Mangotree Enter Neighborhood: Kovan Enter Opening Date (DD-MM-YYYY): 30-06-2025 Enter Closing Date (DD-MM-YYYY): 30-07-2025 Enter Two-Room Units Price: 30000 Enter Three-Room Units Price: 3 Enter Two-Room Slots: 4000 Enter Three-Room Slots: 5 Enter Officer Slots: 2 Project 'Mangotree' created successfully. </pre> b.  <pre> Project Editor Project: Acacia Breeze 1. Edit Opening Date 2. Edit Closing Date 3. Edit Two-Room Slots 4. Edit Three-Room Slots 5. Edit Officer Slots 6. Toggle Project Visibility 0. Back to Main Menu  Enter your selection:3 New Two-Room Slots: 5 Two-Room Slots updated successfully. Project Editor </pre> c.  <pre> 1. Delete Project 2. Generate Report 4. View/Manage Project Applications 5. View/Manage Officer Applications 6. View/Reply Enquiry 0. Back to Main Menu  Please select an option: 2  Are you sure you want to delete this project? (Y/N): Y Project deleted successfully. </pre>
2	Assignment to projects: a. Single project management per application period	a. System does not allow HDB managers to be assigned to more than one project within the same application period	a.  <pre> Create New Project Enter Project Name: Mangotree Enter Neighborhood: Kovan Enter Opening Date (DD-MM-YYYY): 19-04-2025 Enter Closing Date (DD-MM-YYYY): 30-05-2025 Project dates overlap with existing project dates. Please enter a different date range. </pre>
3	Toggle project visibility:	Changes in the visibility are updated in the CSV directly upon exiting the edit project menu	
4	View all projects: a. Visibility toggle on/off	a. Projects are visible	 <pre> Project List [View Only] Enter 'MyProject Portal' for Project and Enquiry Management  [1] Acacia Breeze [ Yishun ] 15-02-2025 - 14-03-2025  [2] GGEZ [ Boon Lay ] 15-04-2025 - 20-05-2025 </pre>
5	Apply filter and view projects	Projects are displayed according to the filter	 <pre> Choose a filter to edit (1-8): 7 Sort ascending? (true/false): false </pre>

			<div> <div>[1] GGEZ [ Boon Lay ] 15-04-2025 - 20-05-2025</div> <div>[2] Acacia Breeze [ Yishun ] 15-02-2025 - 14-03-2025</div> </div>
6	Manage HDB officer registrations: a. View pending and approved registrations b. Approve registration c. Reject registration d. Update project's remaining officer slots	a. Pending and approved registrations can be viewed b. Approvals are processed correctly c. Rejections are processed correctly d. The number of officer slots remaining are updated and changes are saved	a. <div>You have selected: GGEZ Display only 'pending' applications? (yes/no): yes Choose an application to view details: 1. David - GGEZ - PENDING</div> b. <div>Application status updated to: APPROVED You have successfully approved the officer application for GGEZ.</div> c. <div>Application status updated to: REJECTED You have successfully rejected the officer application for GGEZ.</div> d. <div>Enter your selection: 5 New Officer Slots: 2 Officer Slots updated successfully.</div>
7	Manage applicant's BTO application: a. Approve application b. Reject application	a. Approvals are processed correctly b. Rejections are processed correctly	a. <div>1. Approve Project Application 2. Reject Project Application Enter your choice: 1 Application status updated to: SUCCESSFUL You have successfully approved the project application for GGEZ.</div> b. <div>2. Reject Project Application Enter your choice: 2 Application status updated to: UNSUCCESSFUL You have successfully rejected the project application for GGEZ.</div>
8	Manage withdrawal requests: a. Approve withdrawal	a. Withdrawal approved successfully	<div>This application is requesting to withdraw. Do you want to approve the withdrawal request? (yes/no): yes Application status updated to: WITHDRAWN You have successfully approved the withdrawal request for the project application for GGEZ.</div>
9	Manage enquiries: a. View and reply to enquiries:	Enquiries can be successfully accessed and responded to	<div>           Viewing enquiry for project: GGEZ            Enquiries:            1. User =T7654321B [2025-04-18 15:54]            Project=GGEZ            How many slot are there?            No Reply            0. Back to menu            Please select an enquiry [1 - 1]:            1            Enter your reply content: 3!            Reply sent successfully.         </div>
10	Generate report for flat booking: a. Option to filter by categories	Accurate report generated with filter applied	<div>           Choose a filter category: 1. Project Name            2. Flat Type            3. Marital Status            4. Age            0. Done            Enter your choice: 1            Enter project name: GGEZ            A report of the list of applicants based on the filter category is as follows:            Applicant Name: Sarah            Age: 40            Marital Status: MARRIED            Project Name: GGEZ            Flat Type: TWO_ROOM            Report generated successfully.         </div>

## 5. Reflection

### 5.1 Difficulties Encountered and Conquering It

Initially our group encountered difficulties as we tried to come up with additional features before completing the fundamental code. Not to mention, it was challenging to work on the delegated code segments without knowing each other's codes, resulting in overlap of some methods, varying class names, attributes, enums, etc. We were also hit with yet another hardship after the lecture on

SOLID designs as we had to redesign our class diagram and scrap most of the codes. We continuously refine our design prototype even after we begin to develop the program by adding new properties or perform further abstraction as needed. Another difficulty which we faced throughout this project is that we continuously perform changes and refinement of the class diagram as we are writing the code, making the coding process a bit prolonged because of numerous conversions as well as trial and error.

## **5.2 Knowledge learned**

Through this project, we learned how to build a well-designed system using Java through the help of a class diagram and implementing both SOLID and object-oriented design principles. We initially did not understand the need for the SOLID design principles but after facing lots of errors and failed test cases, we realised how crucial the concept of loose coupling was for debugging. Furthermore, we learned the power of platforms like Git-hub when working as a group as well as the importance of documenting our code and progress.

## **5.3 Further improvements**

Since the design of the program focuses more on memory efficiency in terms of time and memory trade-off, the program often needs to linearly traverse through the database repeatedly to search for the ID or name that it stored to obtain the needed data. Such design consideration may save a significant amount of memory when the database is huge, yet it may become slow as a result. To combat this trade-off, various efficient algorithms can be considered, such as binary search. By making sure that the database is always written in sorted manner, we can use binary search, which has  $O(\log N)$  time complexity, to search for required information, saving a significant amount of time without sacrificing too much memory.

Additionally, an interactive and mobile-friendly GUI can also be considered to enhance the user experience as CLI interface may often look overwhelming and daunting and hence not as user-friendly as it was intended to be. Moreover, sophisticated hashing and password management algorithms can also be implemented to create a robust and secure application that respects its user privacy.