

BTO MANAGEMENT SYSTEM

PROBLEM

We need to design a system that allows users

- 1. Applicants to book BTO Projects
- 2. **Managers create Projects**
- 3. Officers to approve Booking Request
- 4. Officers to apply to work for a Project
- 5. Managers to approve BTO applications

CONTRIBUTIONS

MODEL

Consists of the

Applicants	1.
Manager	2.
Officer	3.
Request	4.
Enquiry	5.
Project	6.

REPOSITORY PATTERN

Stores the various models

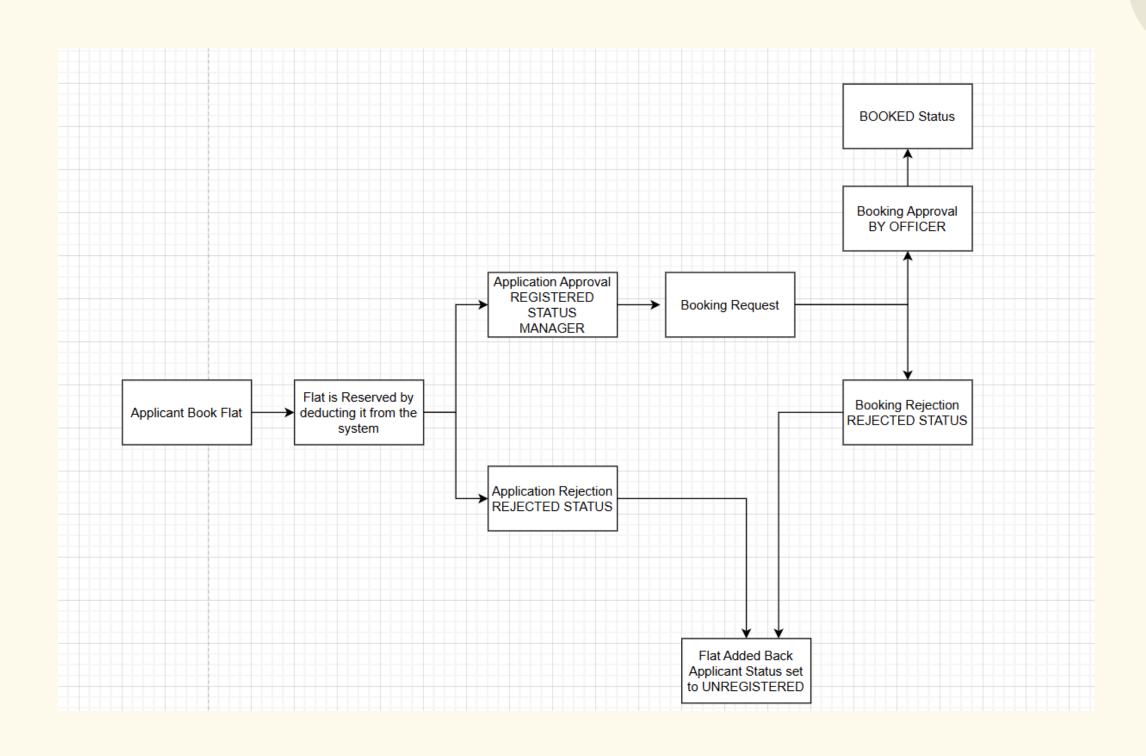


Handles the printing of output and the user interaction flow

CONTROLLER

Handles the business logic, manipulation of the Models and saves them in their repository

BOOKING FLOW





Ensure that Applicants and Officers cannot enter illegal flow by not allowing them to select the option

SOLID - SRP

The Single Responsibility principle recommends that each class should have a clear and singular responsibility, avoiding unrelated tasks.

We ensure SRP by splitting the classes into MVC classes, where the model implements the data structures, the repository classes handle the saving and persistence, the controller classes execute logic to handle what has to be viewed along with updating User, Projects, Enquiry and Request and the boundary class handles only the printing of Objects and interactions.

Example: Applicant Class only defines what a applicant is

SOLID - OCP

The Open/Closed Principle (OCP) states that classes should be open for extension but closed for modification, allowing for the addition of new functionality without changing existing code. OCP can be implemented through abstraction, inheritance, and polymorphism.

In our project, OCP is applied by providing an abstract class, Repository<Model>, which serves as a base for creating specialized repositories like ProjectRepository and RequestRepository. Each subclass provides its own implementation of the getFilePath() method, making it easy to extend the repository system without modifying existing logic.

SOLID - LSP

LSP states that the subtypes can be a replacement for the base types.

In our system, this is used in the Request class.

All subclasses of Request include the different types of requests(eg ProjectApplicationRequest,OfficerApplicationRequest etc). All of these subclasses are interchangeable with the Request interface, ensuring they can be used wherever a Request object is expected while maintaining correct behavior. When determining the type of a request, different request instances can invoke the getRequestType() method.

SOLID - ISP

The interface segregation principle states that we should use many smaller specific interfaces rather than 1 general interface.

In our project, we split the 'Model' into User and Request interfaces so that the different entities can implement the interfaces respectively. For example, Request has a requestType that is different from that of the UserType.

SOLID - DIP

DIP states that both high and low level modules should depend on abstractions.

We implement this by providing a User Interface. Instead of depending on the concrete classes of Applicant, Manager or Officer, we can call User.getID() from the User Interface to get the userID/NRIC.

UserFactory used to create new users during testing

TESTING

Manual + Unit Testing

After implementation, write a small unit test to verify output

TESTING

```
managers = new Manager[3];
// Using the full constructor for clarity
managers[0] = new Manager(
       managerNRIC: "M1234567A", // managerNRIC
       "hashed_password_1", // hashedPassword (placeholder)
       "Alice Wonderland", // managerName
       new ArrayList<>(Arrays.αsList("P101", "P102")) // projectIDsInCharge
);
managers[1] = new Manager(
       managerNRIC: "M7654321B", // managerNRIC
       "hashed_password_2", // hashedPassword (placeholder)
       "Bob The Builder", // managerName
       new ArrayList<>() // Empty project list
);
managers[2] = new Manager(
       managerNRIC: "M9876543C", // managerNRIC
       "hashed_password_3", // hashedPassword (placeholder)
       "Charlie Chaplin", // managerName
       new ArrayList<>(Arrays.αsList("P201")) // projectIDsInCharge
);
```

EDGE CASES + NEW TEST CASES

Officer Applies to project they are already officer of

Officer applies to Project they are incharge of

Data Persistance issues

THANK YOU

