Declaration of Original Work for CE/CZ2002 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.

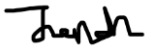
Name	Cours	Lab Group	Signature /Date
Jonah Choon Cai Ming	SC2002	SCMA Group 5	
Lee Jia Xuan	SC2002		
Liew Shan Yuan	SC2002		
Wendy Goh Yu Xin	SC2002		
Ariel Jan Matthew Mudjadi	SC2002		

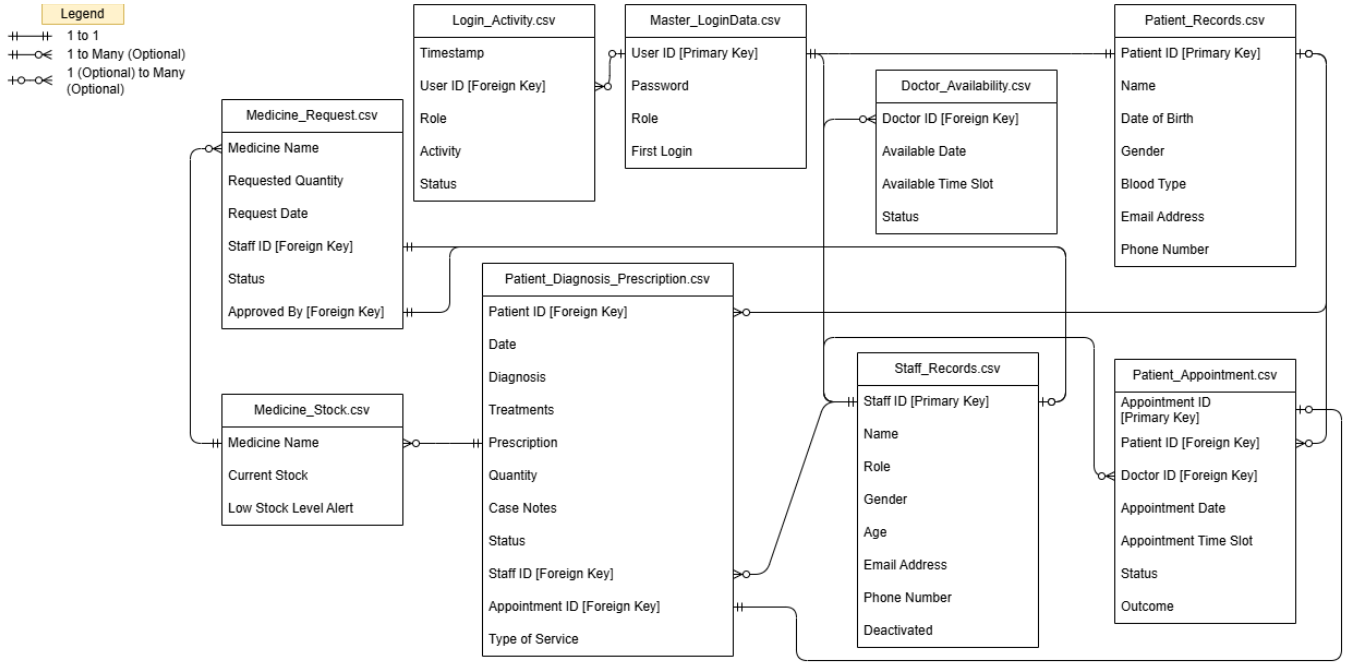
Table of Contents

1. Design Considerations

1.1. Approach Taken

1.1.1. Database Design and Relationship Mapping

A relational database is used to store information about users (Patients, Doctors, Pharmacists, Administrators), appointments, medical records, prescriptions, and medicine stock. Entities are represented as tables, and relationships (such as appointments between patients and doctors) are mapped using AppointmentID as the Primary Key for the system to identify the specific data entry.



1.1.2. 4 Stages of Computational Thinking

- **Decomposition:** Break down the use cases into its individual functions (e.g. Required functions for Patient Schedule Appointment – retrieve doctor availability, view available appointments, book appointment)
- **Pattern Recognition:** Identify repeated functions across all use cases (e.g. log in, view appointments)
- **Abstraction**: Simplified functions that are overlapping with each other (e.g. doctors can view appointment without another scheduling function)
- **Algorithm:** Design algorithms to handle the task efficiently (e.g. appointment scheduling algorithm to check the doctor's availability to ensure that there is no overlap in booking and confirming that the appointment fits within the time slots given.)

1.1.3. Unit Testing

We implemented the project in phases and ensured each section is fully working before proceeding:

1) **User Authentication:** Ensure all 4 user types can log in with the correct roles
2) **Diagnosis & Treatment Entry:** Ensure doctors can enter diagnosis and treatments with and without an active appointmentID

3) **Prescription Management:** Verify that prescriptions are assigned correctly and Pharmacist can mark it as "Dispensed" in the system
4) **Medicine Stock Management:** Test that medicine stock is immediately updated after approval by Administrator and ensure stock is decremented after dispensing medication to Patient
5) **Appointment Scheduling:** Ensure available appointments are based on Doctor's availability and ensure Doctors cannot be double-booked on the same timeslot

1.2. Assumptions

a) No csv files are opened while the HMS is up and running (E.g. Patient_Records.csv)
b) There is at least one Patient, Doctor, Pharmacist and Administrator in the HMS at all times
c) Each Patient's appointment with a Doctor is 1 hour long (E.g. 0900 to 1000 hrs.)
d) Each Patient will only book maximum 1 appointment per day with each Doctor
e) Doctors do not take last-minute MC or leave
f) Patients receive prescription on same day of appointment
g) Low medicine stock is approved on same day of request
h) Any Doctor removed by Administrator do not have any upcoming appointments with any Patient

1.3. Design Principles Applied

1.3.1. 4 Main Concepts of Object-Oriented Programming

- **Abstraction:** The system hides complex internal logic, presenting only essential features (e.g., doctors can see appointments, but not the internal working of scheduling).
- **Encapsulation and Information Hiding:** Critical data such as user credentials (passwords) are protected within the user class, and are not directly accessible from other objects.
- **Inheritance:** A generic User class is created, and specialized classes such as Doctor, Patient, and Administrator inherit from this class, reducing code duplication.
- **Polymorphism:** Methods like ViewAppointments() are implemented in the base User class but behave differently for doctors and patients, providing specialized views.

1.3.2. SOLID Design Principles

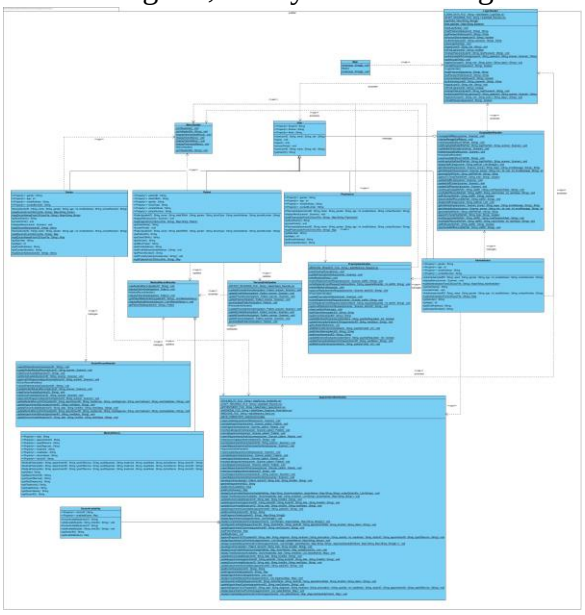
- **Single Responsibility Principle (SRP):** Each class (e.g., Doctor, Appointment, Prescription) has a well-defined responsibility.
- **Open/Closed Principle (OCP):** The system is open for extension (e.g., adding new roles like Nurse), but closed for modification (core functionality remains untouched).
    - New medication can be directly added into Medicine_Stock.csv with the Warning Threshold set for each specific medicine

- **Liskov Substitution Principle (LSP):** Derived classes like Doctor and Patient can replace the base User class without breaking the system.
- **Interface Segregation Principle (ISP):** Interfaces are designed to be minimal, ensuring that classes only implement what they need (e.g., the Doctor class focuses on appointment and prescription-related interfaces).
- **Dependency Inversion Principle (DIP):** High-level modules, like the Administrator class, do not depend on low-level modules like StockManagement directly but through abstractions.

2. Unified Modeling Language

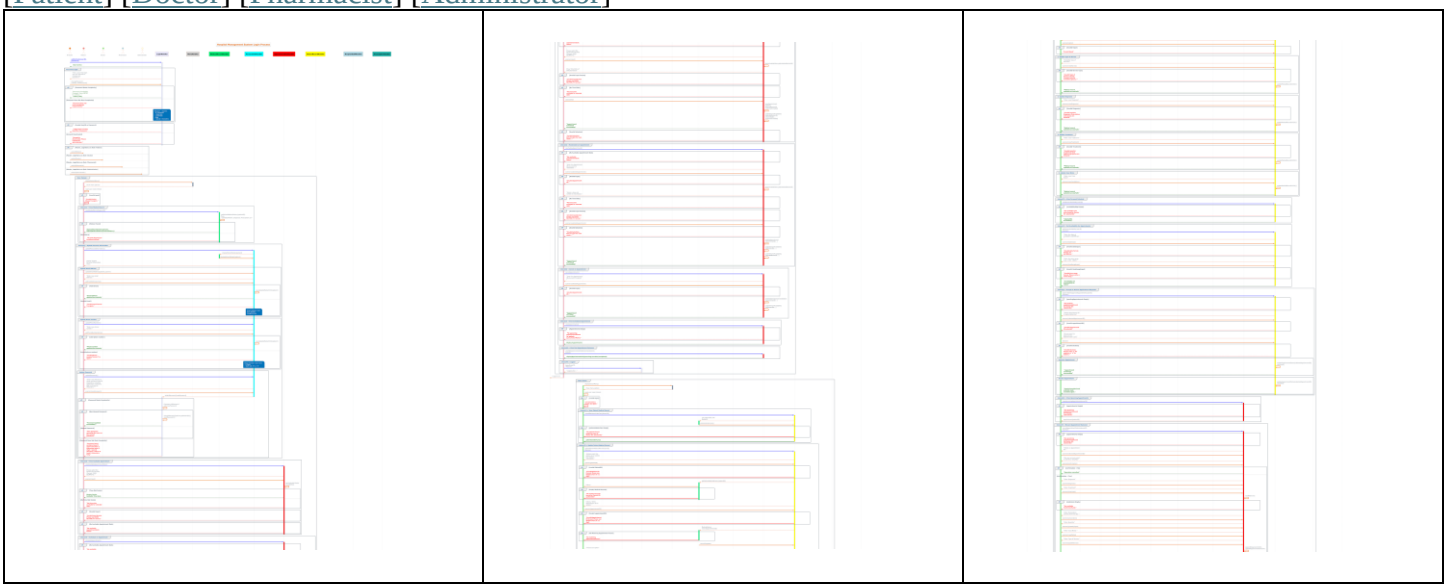
2.1. UML Class Diagram

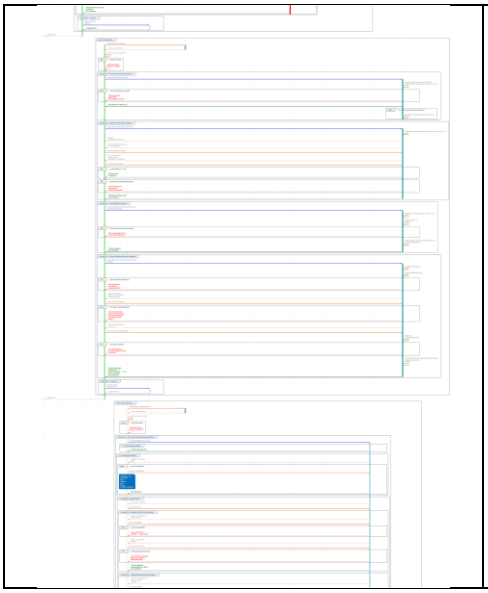
Due to the large scale of the UML Class Diagram, kindly view it through this hyperlink: [GitHub]

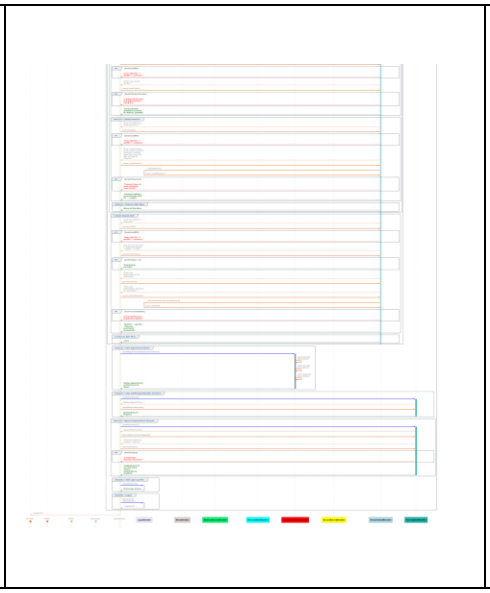


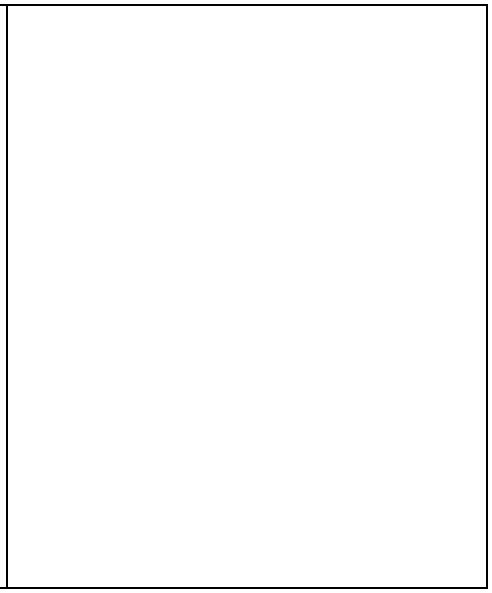
2.2. UML Sequence Diagram

Due to the large scale of the UML Sequence Diagram, kindly view it through these hyperlinks: [GitHub] [Patient] [Doctor] [Pharmacist] [Administrator]



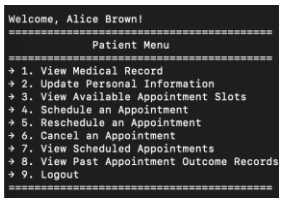
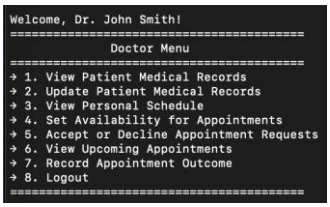
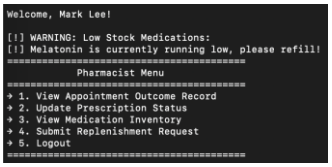
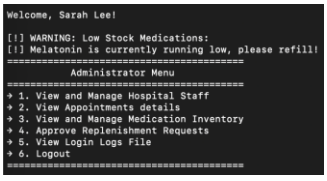




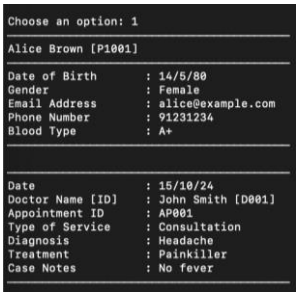
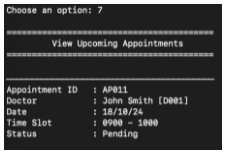
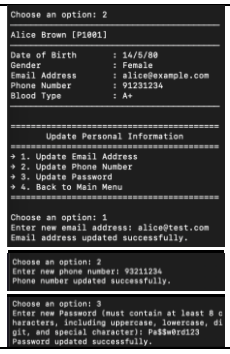
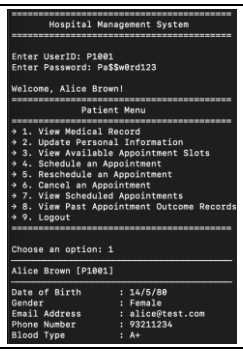
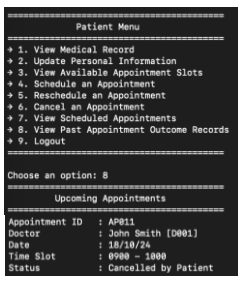


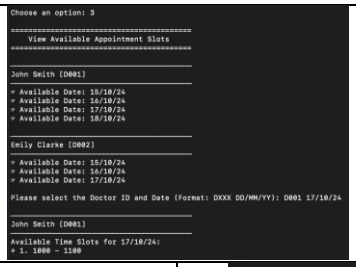
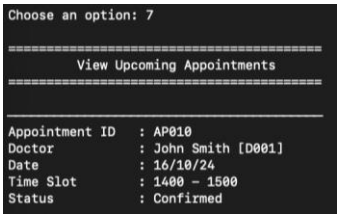
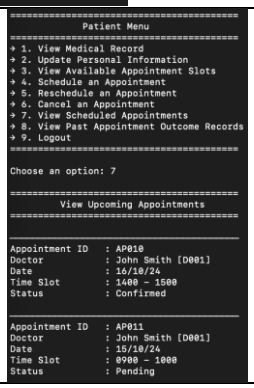
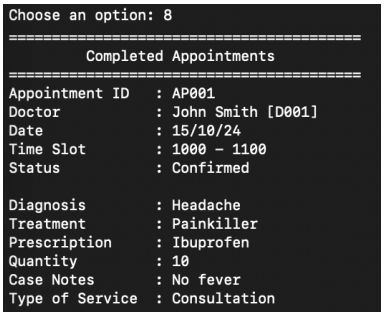
3. Testing Results

3.1. Main Menu

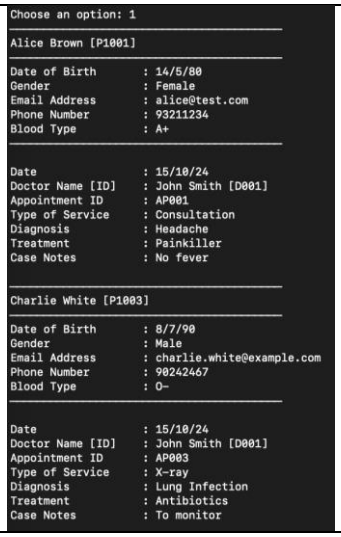
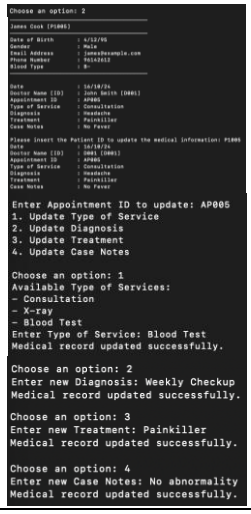
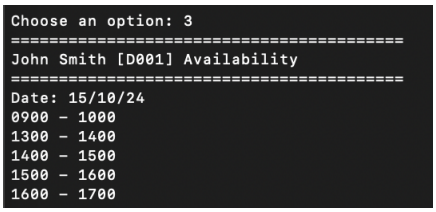
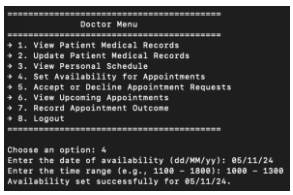
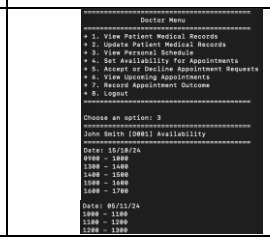
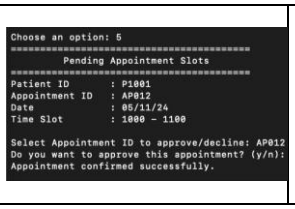
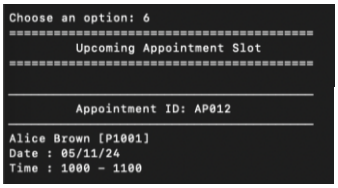
Patient Menu	Doctor Menu	Pharmacist Menu	Administrator Menu
			

3.2. Patient Actions

Test Case	Input & Output	Test Case	Input & Output
1		5	 
2	 	6	 

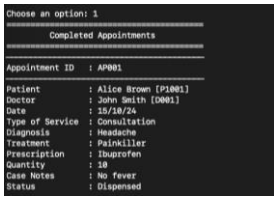
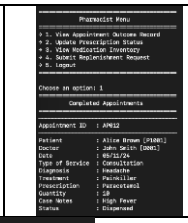
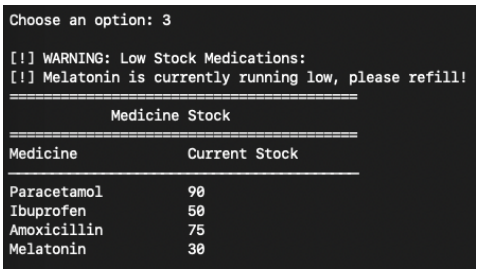
3		7	
4	 	8	

3.3. Doctor Actions

Test Case	Input & Output	Test Case	Input & Output
9		10	 
11		12	 
13	 	14	

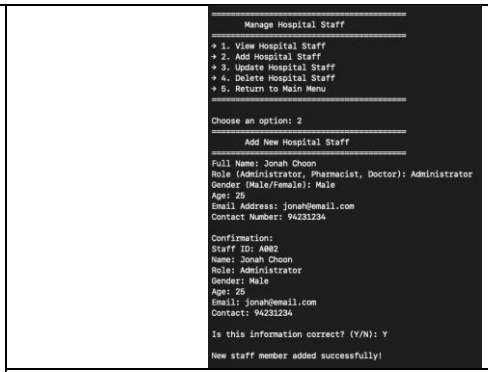
15		

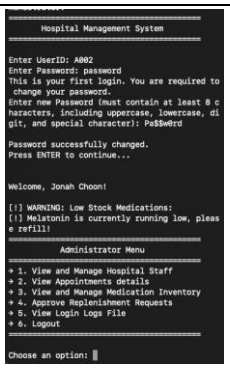
3.4. Pharmacist Actions

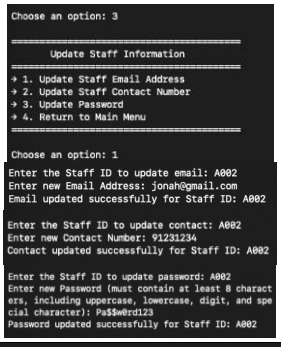
Test Case	Input & Output	Test Case	Input & Output
16		17	 
18		19	

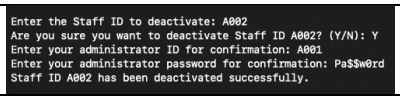
3.5. Administrator Actions

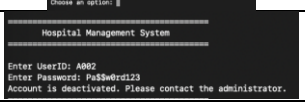
Test Case	Input	Output
20		





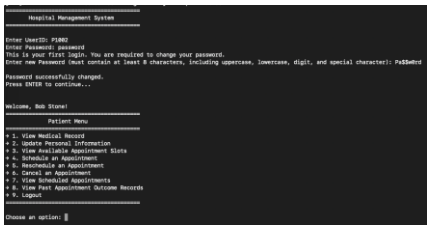
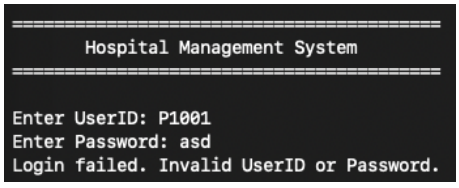






Test Case	Input & Output	Test Case	Input & Output
21		22	
23			

3.6. Login System and Password Management

Test Case	Input & Output	Test Case	Input & Output
25		26	

3.7. Additional Functions

S/N	Function

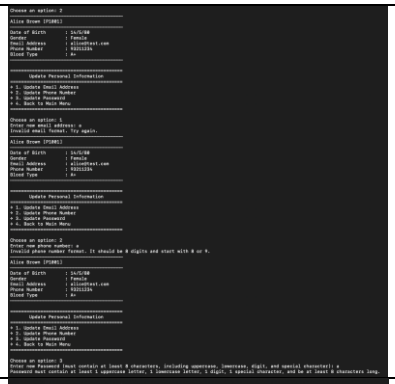
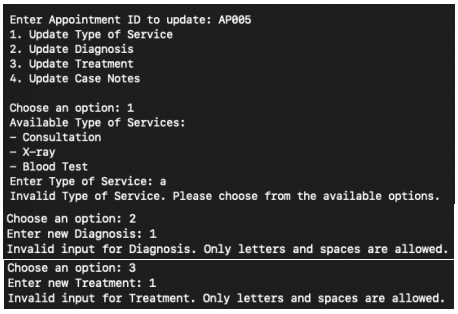
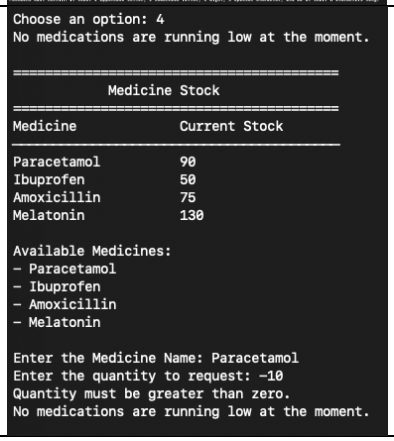
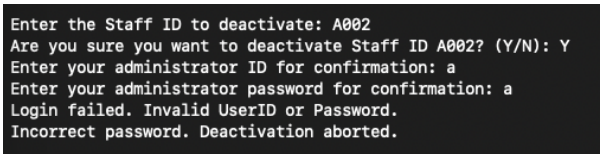
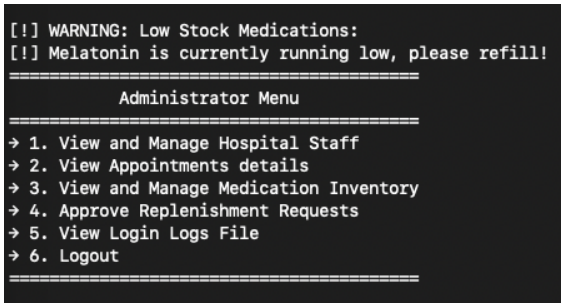
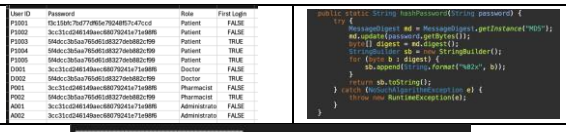
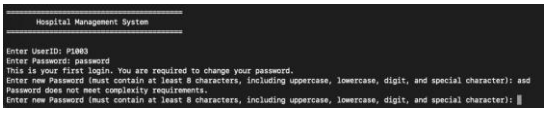
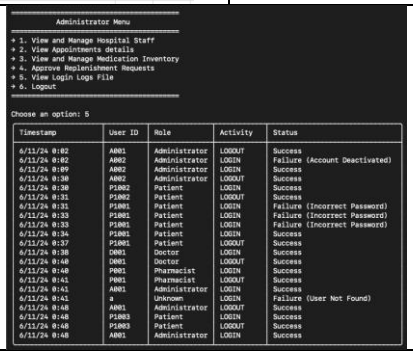
1	Regex on Update Personal Information
	• Updated email address must be in the following format "XXX@XXX.XXX"
	• Updated phone number must have 8 digits and must start with "8" or "9"
	• Ability to update password, new password must fulfil the following password policy of:
	  ○ 1 Uppercase letter
	  ○ 1 Lowercase letter
	  ○ 1 Digit
	  ○ 1 Special character
	  ○ At least 8 characters long

2	Regex on Update Patient Medical Records • In the update of diagnosis and treatment plan, no digit input is allowed • Type of Service must be chosen from the list
3	Error Handling in Submit Replenishment Request • Implemented Error Handling, User can only input positive integer
4	Extra Security Step in View and Manage Hospital Staff • When deleting a hospital staff, Administrator must enter ID and Password again for confirmation to avoid accidental deletion
5	Warning System in View and Manage Medication Inventory • Implemented a warning when medicine stock is low. • The warning value is saved directly in "column C" of Medicine_Stock.csv (E.g. Melatonin alert value is 30)
6	Logging System in View and Manage Medication Inventory • Implemented logging to keep track of Request Date, Medicine, Quantity, Staff ID (Requestor & Approver) and Status
7	Login User Password Storing • For security, passwords are not stored in plain-text, they are hashed with MD5 in Master_LoginData.csv

8	Regex on New Password Login
	• New password must fulfil the following password policy of:
	  ○ 1 Uppercase letter
	  ○ 1 Lowercase letter
	  ○ 1 Digit
	  ○ 1 Special character
	  ○ At least 8 characters long

9	Administrator View Login Logs
	• Allow Administrator to review successful and failed user login attempts on HMS
	• Types of Status
	  ○ Success
	  ○ Failure (User Not Found)
	  ○ Failure (Incorrect Password)
	  ○ Failure (Account Deactivated)

3.8. Additional Function Test Case

Test Case	Input & Output	Test Case	Input & Output
1		2	
3		4	
5		6	
7	 	8	
9			

4. Reflection

Database Design

Since only the basic database structure was provided, we had to brainstorm on how to expand the existing database for our HMS implementation. The database had to effectively store the information in a clear and organized manner so that we can avoid re-designing the database during the code implementation phase. Hence, we drew the UML and ER diagram to help visualize the class, attributes, methods and relationships for all use cases.

UML Diagram

Multiple classes were used, creating a complex system of dependencies. Scrutiny of each class was required to accurately establish their relationships with one another. With more relationships learnt in the 2nd half of the course (dependency, association, etc.), it provides the UML class diagram with a clearer and more comprehensive picture of how each class plays a role in the outcome of the design. Abstraction, encapsulation, polymorphism and inheritance, critical concepts of object-oriented design and programming are evident through the display of the UML class diagram in a visual, easy to understand diagram.

For the UML sequence diagram, it was challenging to ensure that every function that each role was able to perform was covered, complete with every error handling functions.

Code Implementation

Although the tutorial and lab examples have sufficiently prepared us for this project, we have not implemented on a system of this scale before. Instead of looking the project as one big system, we applied the 4 stages of computational thinking to break up the project into smaller individual deliverables

5. GitHub Repository Link

https://github.com/JonahChoon/SC2002-OOP-Assignment