





Declaration of Original Work for SC2002/CE2002/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.

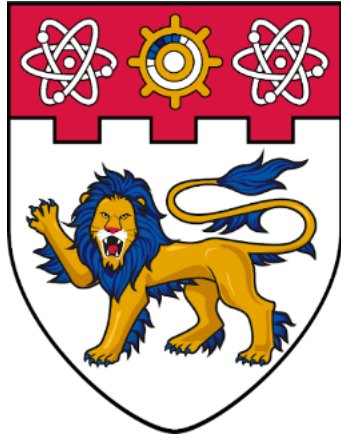
We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work. In addition, disciplinary actions may be taken.

Name	Course (SC2002/CE2002 CZ2002)	Lab Group	Signature /Date
Quek Jun Siong	SC2002	SCSC	 11/21/2014
Tan Yu Xiu	SC2002	SCSC	 11/21/2024
Sun Si Tong	SC2002	SCSC	 11/21/2024
Jiang Zong Zhe	SC2002	SCSC	 11/21/2024

Important notes:

Name must EXACTLY MATCH the one printed on your Matriculation Card.

Student Code of Academic Conduct includes the latest guidelines on usage of Generative AI and any other guidelines as released by NTU.



**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

SC2002: Object-Oriented Design & Programming

Group Assignment Report

Prepared By: Jiang Zong Zhe, Quek Jun Siong, Sun Sitong, Tan Yu Xiu

Design Considerations

Design Approach

Our project was designed with close adherence to the principles of loose coupling. In line with Software Engineering (SWE) Principles, we decided to implement the Model-View-Controller (MVC) pattern to create a scalable application. The Models (src/Models) handle data management through CSV files like Patient_List.csv, while the Views (src/Views) manage the user interface, and Controllers (src/Services) ensures the data flow between them. This model increases maintainability through isolated components that can be modified independently, enhanced testability as components can be tested separately, and improved scalability as new features can be added easily by extending existing patterns. For example, adding new patient fields only requires changes to the Patient model, while UI updates can be made without affecting the business logic.

Following these principles, we started the project by analysing the user requirements and functionalities outlined in the assignment document. This included identifying the key models stated above while mapping out their interactions and system boundaries. We then started to draw our UML class diagram by first listing all the classes, interfaces and enumerations we will need.

Design Principles

In line with object-oriented SOLID principles in Java, we were strongly guided by all 5 principles.

1. Single Responsibility Principle (SRP)

We adhered closely to the SRP in delegating tasks to our classes, ensuring that each class is only responsible for tasks it was assigned to perform, promoting loose coupling. For example, the CSVHandler class is responsible for all CSV-related features, such as reading from and writing to CSV classes.

```
public class CSVHandler {  
    private static final String COMMA_DELIMITER = ",";  
    public static List<List<String>> readCSVLines(String filePath) {  
    public static void writeCSVLines(String[] headers, String[] lines, String filePath) {
```

CSVHandler Class

Each CSV file is also managed by one separate entity class. Thus, each entity class is only responsible for their respective CSV files, fulfilling the Single Responsibility Principle.

2. Open Closed Principle (OCP)

We used OCP to ensure our program is open for extension but closed for modification. For example, we created a "Menu" interface which can be implemented and extended to create different types of menus like "PharmacistMenu", "AdministratorMenu", etc. Each menu subclass overrides operations such as "showMenu" depending on the type of menu. By adhering to OCP, we ensure that the addition of new menu types only requires the creation of new classes or modifications within those subclasses, leaving existing code untouched.

OCP is also demonstrated in our code through the method in which the user controllers use entity class functions. The respective user controllers access certain entity class functions through interfaces, and import only the needed interfaces for controller methods. This way, when more CSV files with their respective entity classes and interfaces are created, the user controllers can import these new interfaces separately from the existing interfaces, thus allowing for extension of newer interfaces, while closing off modification of existing interfaces.

3. Liskov Substitution Principle (LSP)

We used LSP to make sure our subclass can substitute our base classes without causing trouble. For example, we implemented a "User" class and its subclasses like "Doctor", "Patient", etc. Each subclass properly implements the base class methods like "login" and "logout" without changing their behaviour contract, ensuring they can be used interchangeably wherever a User is expected.

4. Interface Segregation Principle (ISP)

To avoid using a general interface that every low-level module must implement, we separated it into smaller ones specific to the requirements. Each interface is designed with a specific responsibility. For example, "MedicalRecordInterface" handles only medical record operations, while "AppointmentInterface" focuses on appointment management functionality. This segregation ensures that classes implementing these interfaces only need to provide methods that are relevant to their purpose.

5. Dependency Inversion Principle (DIP)

To adhere to DIP, we ensured that both high-level and low-level implementations depend on abstractions/interfaces only. For example, instead of accessing the entity classes directly, the methods are implemented in the respective entity class interfaces. The user controllers then access the methods from the interfaces. This way, when modifications or extensions are made

to the higher-level user controllers, the lower-level entity classes do not have to be modified as the interfaces that they are implementing are unchanged, effectively decoupling the user controllers from our entity classes, and demonstrating the DIP.

```
package Models;

import Services.MedicationInventoryInterface;
import Services.StaffInterface;
```

Importing Interfaces

Project Assumptions

1. Abilities of Users

- 1) The administrator has the right to reset the passwords of patients and staff members.
- 2) The doctor has the right to register (add) or discharge (remove) patients.
- 3) Doctors can set their own appointment availabilities.

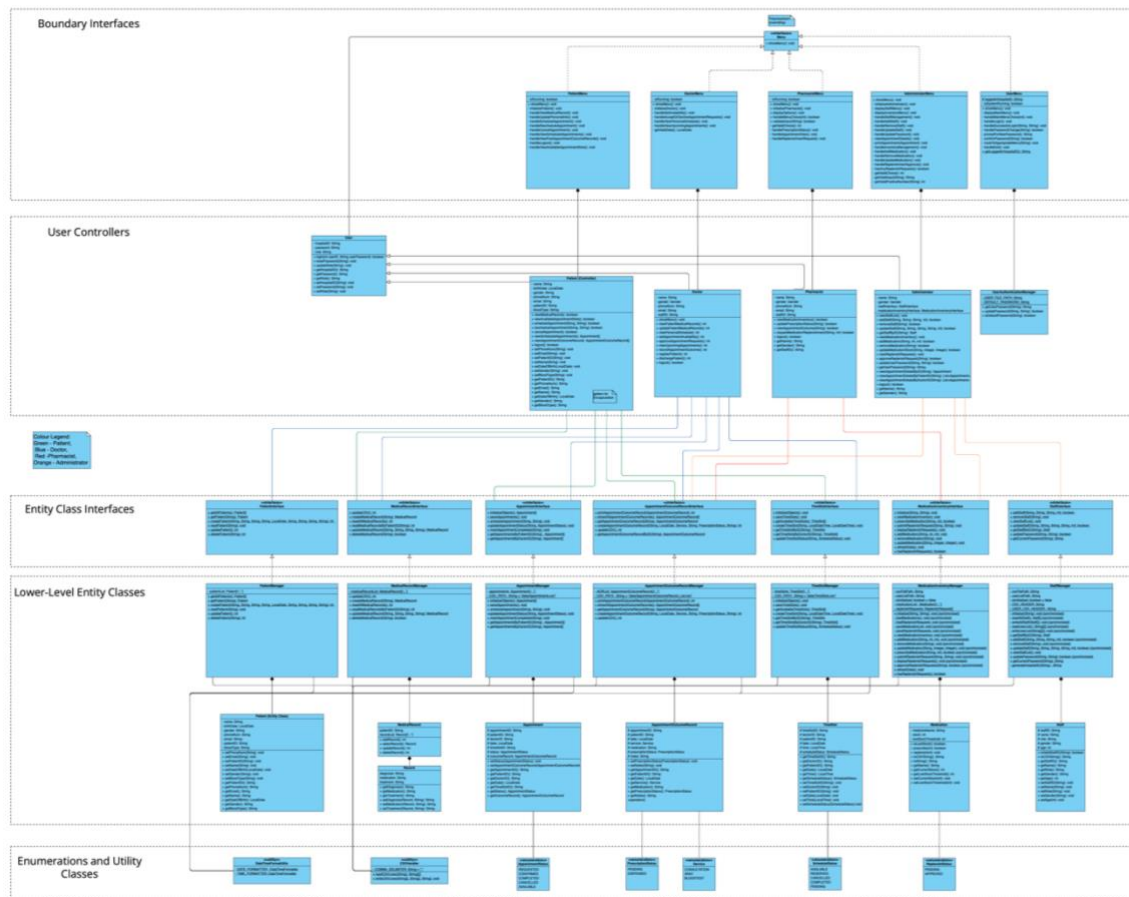
2. Properties of Attributes

- 1) Hospital IDs are unique and immutable.
- 2) Only one type of medication will be prescribed at a time.

3. Constraints on User Inputs

- 1) Number of CSV inputs will stay below 100.
- 2) Medicine names and other data types that are not controlled within our database will not have validity checking.

Detailed UML Class Diagram



Testing

Here are some of the base test cases for different use cases.

1. Authentication

1.1 First Time Login

New users should be able to login with their hospital ID and default password (PWD) “password”.

ID	Input	Expected Output	Actual Output	Result
----	-------	-----------------	---------------	--------

TC01	Correct ID and default PWD	Login successfully, prompt to change password	<pre> ===Welcome to HMS!=== 1: Login 2: Exit System Enter your choice: 1 Enter your Hospital ID: D002 Enter your password: password You must change your default password. Enter new password: █ </pre>	Pass
------	----------------------------	---	--	------

1.2 Change Password

When a new user login with default password, they should be prompted to change it.
(Follow up from TC01)

ID	Input	Expected Output	Actual Output	Result
TC02	TC01, new valid PWD	Verify that the PWD change is successful, and the user can log in with the new PWD.	<pre> You must change your default password. Enter new password: password2 Confirm new password: password2 ==Login successful!== Welcome, D002 Perform the following methods: 1: View Patient Medical Records 2: Update Patient Medical Records 3: View Personal Schedule 4: Set Availability for Appointments 5: Accept or Decline Appointment Requests 6: View Upcoming Appointments 7: Record Appointment Outcome 8: Register Patient 9: Discharge Patient 10: Logout </pre>	Pass
TC03	TC01, same as default PWD	PWD change unsuccessful, prompted to change to another PWD	<pre> ===Welcome to HMS!=== 1: Login 2: Exit System Enter your choice: 1 Enter your Hospital ID: D002 Enter your password: password You must change your default password. Enter new password: password New password cannot be the default password. Enter new password: █ </pre>	Pass

1.3 Login Error Handling

When users enter incorrect credentials.

ID	Input	Expected Output	Actual Output	Result
TC04	Correct credentials	Login successful with relevant menu display	<pre> ===Welcome to HMS!=== 1: Login 2: Exit System Enter your choice: 1 Enter your Hospital ID: D001 Enter your password: password1 ==Login successful!== Welcome, D001 Perform the following methods: 1: View Patient Medical Records 2: Update Patient Medical Records 3: View Personal Schedule 4: Set Availability for Appointments 5: Accept or Decline Appointment Requests 6: View Upcoming Appointments 7: Record Appointment Outcome 8: Register Patient 9: Discharge Patient 10: Logout </pre>	Pass
TC05	Incorrect credentials (either ID or PWD)	Login unsuccessful	<pre> ===Welcome to HMS!=== 1: Login 2: Exit System Enter your choice: 1 Enter your Hospital ID: D001 Enter your password: password Invalid Hospital ID or Password. Please try again. ===Welcome to HMS!=== 1: Login 2: Exit System Enter your choice: █ </pre>	Pass

2. Patient Modules

2.1 View Medical Record

ID	Input	Expected Output	Actual Output	Result
TC06	TC04	Displays the patient's medical record with all relevant information	<pre>===== Patient Profile ===== Patient ID: P1001 Name: Alice Brown Date of Birth: 1980-05-14 Gender: Female Blood Type: +65 9876 5432 Contact Information: A+ Diagnosis: Diagnosis1 Medication: Medication1 Treatment: Treatment1 Diagnosis: Diagnosis2 Medication: Medication2 Treatment: Treatment2 =====</pre>	Pass

2.2 Update Personal Information

ID	Input	Expected Output	Actual Output	Result
TC07	New email and contact number	Verify that patient's contact information is updated successfully	<pre>Patient Menu: 1: View Medical Record 2: Update Personal Information 3: View Available Appointment Slots 4: Schedule an Appointment 5: Reschedule an Appointment 6: Cancel an Appointment 7: View Scheduled Appointments 8: View Past Appointment Outcome Records 9: Logout 2 Enter field to update: 1: Phone number 2: Email 1 Enter your new phone number: 9000 9002 Phone number changed successfully</pre>	Pass

2.3 View Available Appointment Slots

ID	Input	Expected Output	Actual Output	Result
TC08	TC04	Displays a list of available appointment slots	<pre>Available Appointment Slots: Slot ID: TS1, Doctor: D001, Date: 2024-11-22, Time: 09:00-09:30 Slot ID: TS2, Doctor: D001, Date: 2024-11-22, Time: 10:00-10:30 Slot ID: TS3, Doctor: D001, Date: 2024-11-22, Time: 11:00-11:30 Slot ID: TS4, Doctor: D001, Date: 2024-11-22, Time: 12:00-12:30</pre>	Pass

2.4 Appointment Handling

ID	Input	Expected Output	Actual Output	Result
TC09	Schedule Appointment	Outputs a list of available slots to choose from	<pre>Appointment ID, Patient ID, Doctor ID, Date, Time Slot ID, Status, Service APT1, P1001, D001, 2024-11-22, TS1, PENDING, CONSULTATION Available Appointment Slots: Slot ID: TS1, Doctor: D001, Date: 2024-11-22, Time: 09:00-09:30 Slot ID: TS2, Doctor: D001, Date: 2024-11-22, Time: 10:00-10:30 Slot ID: TS3, Doctor: D001, Date: 2024-11-22, Time: 11:00-11:30 Slot ID: TS4, Doctor: D001, Date: 2024-11-22, Time: 12:00-12:30 Enter slot ID to book (or 'back' to return): TS1 Available services: 1: CONSULTATION 2: XRAY 3: BLOODTEST Choose service (1-3): 1 Appointment request sent successfully!</pre>	Pass
TC10	View AOR	Displays list of AOR		Pass

3. Doctor Modules

3.1 View Patient Medical Record

ID	Input	Expected Output	Actual Output	Result
TC11	TC04	Displays relevant patients' medical record and medical history	<pre>===== Patient Profile ===== Patient ID: P1002 Name: Bob Stone Date of Birth: 1975-11-22 Gender: Male Blood Type: +65 9000 0000 Contact Information: B+ [Models.Patient@59851f0b, Models.Patient@4528ebad, Models.Patient@5419f379] Patient profile exists. ID: P1002, Diagnosis: Diagnosis1, Medication: Medication1, Treatment: Treatment1 Medical Records: 1 =====</pre>	Pass

3.2 Update Patient Medical Record

ID	Input	Expected Output	Actual Output	Result
TC12	Patient ID, new diagnosis and treatment plans	Verify that the medical record is updated successfully, reflecting new information.	<pre>1 ID,Diagnosis,Medication,Treatment 2 PatientID, Diagnosis, Medication, Treatment 3 P1001, Diagnosis1, Medication1, Treatment1 4 P1001, Diagnosis2, Medication2, Treatment2 5 P1001,Diagnosis1,Medication1,Treatment10 6 P1002, Diagnosis1, Medication1, Treatment1 7 Welcome, D001 Perform the following methods: 1: View Patient Medical Records 2: Update Patient Medical Records 3: View Personal Schedule 4: Set Availability for Appointments 5: Accept or Decline Appointment Requests 6: View Upcoming Appointments 7: Record Appointment Outcome 8: Register Patient 9: Discharge Patient 10: Logout 2 Enter Patient ID: P1001 [Models.Patient@5f375610, Models.Patient@1810399e, Models.Patient@32d992b2] Patient profile exists. Enter Diagnosis: Diagnosis1 Enter Medication: Medication1 Enter Treatment: Treatment10 Task completed successfully.</pre>	Pass

3.3 View Personal Schedule

Doctors can view their personal appointment schedule.

ID	Input	Expected Output	Actual Output	Result
TC13	Doctors must input their available appointment slots.	Displays a list of schedules for next 7 days	<pre>Your Schedule: Slot ID: TS1 Date: 2024-11-22 Time: 09:00 - 09:30 Status: AVAILABLE Patient: null Slot ID: TS2 Date: 2024-11-22 Time: 10:00 - 10:30 Status: AVAILABLE Patient: null Slot ID: TS3 Date: 2024-11-22 Time: 11:00 - 11:30 Status: AVAILABLE Patient: null</pre>	Pass

3.4 Set Appointment Availability

ID	Input	Expected Output	Actual Output	Result
TC14	TC04, timeslot	Verify that the doctor's availability is updated, and patients can see the new slots when scheduling appointments.	<pre>TimeSlot ID,Start Time,End Time,Doctor ID,Patient ID,Status TS1,2024-11-22T09:00,2024-11-22T09:30,D001,null,AVAILABLE TS2,2024-11-22T10:00,2024-11-22T10:30,D001,null,AVAILABLE TS3,2024-11-22T11:00,2024-11-22T11:30,D001,null,AVAILABLE TS4,2024-11-22T12:00,2024-11-22T12:30,D001,null,AVAILABLE Set Appointment Availability Enter date for slots (YYYY-MM-DD): 2024-11-22 Enter number of slots to create: 4 Enter starting hour (9-17): 9 Time slots created successfully.</pre>	Pass

3.5 Manage Appointment Requests

ID	Input	Expected Output	Actual Output	Result
TC15	Appointment ID	Appointment status changes to "confirmed" when accepted/ "cancelled" when declined;	<pre>APT3,P1001,D001,2024-11-22,TS4,CONFIRMED,BLOODTEST</pre> <pre>Pending Appointment Requests: ----- Appointment ID: APT3 Patient ID: P1001 Date: 2024-11-22 Time: 12:00 - 12:30 Accept appointment? (Y/N): y Appointment confirmed. -----</pre>	Pass

3.6 View Upcoming Appointments

ID	Input	Expected Output	Actual Output	Result
TC16	TC04	Displays upcoming appointments	<pre>Upcoming Appointments: ----- Appointment ID: APT3 Patient ID: P1001 Date: 2024-11-22 Time: 12:00 - 12:30 Status: CONFIRMED -----</pre>	Pass

3.7 Record Appointment Outcome

ID	Input	Expected Output	Actual Output	Result
TC17	Appointment ID, outcome of appointment	Verify that the appointment outcome is recorded, and relevant updates are visible to the patient	<pre>Appointment ID,Date,Service,Medicine,Prescription Status,Notes APT3,2024-11-22,BLOODTEST,Ibuprofen,PENDING,need further checks</pre> <pre>Confirmed Appointments: ID: APT3, Patient: P1001, Date: 2024-11-22, Service: BLOODTEST Enter Appointment ID to record outcome: APT3 Enter medication prescribed (or 'NONE'): Ibuprofen Enter notes for appointment (or 'NIL'): need further checks Appointment outcome record created successfully. Appointment outcome recorded successfully.</pre>	Pass

4. Pharmacist Modules

4.1 View AOR

ID	Input	Expected Output	Actual Output	Result
TC18	TC04	Displays the appointment outcome details with prescribed medications.	<pre>Pending Prescriptions: Appointment Outcome Records: ----- Appointment ID: APT3 Date: 2024-11-22 Service: BLOODTEST Prescribed Medicine: Ibuprofen Status: PENDING Notes: need further checks -----</pre>	Pass

4.2 Update Prescription Status

ID	Input	Expected Output	Actual Output	Result
TC19	TC18	Updates status in relevant data files	<pre>Appointment ID,Date,Service,Medicine,Prescription Status,Notes APT3,2024-11-22,BLOODTEST,Ibuprofen,DISPENSED,need further checks Medicine Name,Initial Stock,Low Stock Level Alert Paracetamol,100,20 Ibuprofen,68,10 Amoxicillin,75,15 Appointment Outcome Records: ----- Appointment ID: APT3 Date: 2024-11-22 Service: BLOODTEST Prescribed Medicine: Ibuprofen Status: PENDING Notes: need further checks ----- Enter Appointment ID to dispense medication (or 'back' to return): APT3 Prescribed 1 units of Ibuprofen Prescription status updated to DISPENSED. Successfully dispensed medication and updated records.</pre>	Pass

4.3 View Medical Inventory

ID	Input	Expected Output	Actual Output	Result
TC20	TC04	Displays list of medication with their current stock levels and low stock threshold	<pre>Pharmacist Menu: 1: View Appointment Outcome Record 2: Update Prescription Status 3: View Medication Inventory 4: Submit Replenishment Request 5: Logout 3 Current Inventory ----- Medicine: Paracetamol Current Stock: 100 Low Stock Alert Level: 20 Medicine: Ibuprofen Current Stock: 70 Low Stock Alert Level: 10 Medicine: Amoxicillin Current Stock: 75 Low Stock Alert Level: 15</pre>	Pass

4.4 Submit Medication Replenish Requests

ID	Input	Expected Output	Actual Output	Result
TC21	Medication name	Update medication requests document visible to admins	<pre>Medicine Name,Requested Quantity,Status Ibuprofen,4,PENDING Paracetamol,200,PENDING Amoxicillin,10,PENDING Pharmacist Menu: 1: View Appointment Outcome Record 2: Update Prescription Status 3: View Medication Inventory 4: Submit Replenishment Request 5: Logout 4 Current Inventory ----- Medicine: Paracetamol Current Stock: 100 Low Stock Alert Level: 20 Medicine: Ibuprofen Current Stock: 70 Low Stock Alert Level: 10 Medicine: Amoxicillin Current Stock: 75 Low Stock Alert Level: 15 Enter medicine name to replenish (or 'exit' to return to main menu): Ibuprofen Enter quantity to replenish: 4 Replenish request submitted to administrator.</pre>	Pass

5. Administrator Modules

5.1 View and Manage Staff

ID	Input	Expected Output	Actual Output	Result
TC22	TC04	Display list of staff and menu for staff management	<pre>==Login successful== Welcome, A001 Administrator Menu: 1: View and Manage Hospital Staff 2: View Appointments Details 3: View and Manage Medication Inventory 4: Approve Replenishment Requests 5: Reset User Password 6: Logout Enter your choice (1-6): 1 Staff List ----- ID: D001, Name: John Smith, Role: Doctor, Gender: Male, Age: 50 ID: D002, Name: Emily Clarke, Role: Doctor, Gender: Female, Age: 38 ID: P001, Name: Mark Lee, Role: Pharmacist, Gender: Male, Age: 29 ID: A001, Name: Sarah Lee, Role: Administrator, Gender: Female, Age: 40 Staff Management Options: 1: Add new staff 2: Remove staff 3: Update staff details 4: Return to main menu Enter your choice (1-4):</pre>	Pass

5.2 View Appointment Details

ID	Input	Expected Output	Actual Output	Result
TC23	TC04	Display entire list of appointments with details	<pre>View Appointments By: 1: View All Appointments 2: View by Doctor ID 3: View by Patient ID 4: View by Status 5: Return to Main Menu Enter your choice (1-5): 1 All Appointments in System: Appointment ID: APT1 Patient ID: P1001 Doctor ID: D001 Date: 2024-11-22 Service: CONSULTATION Status: CANCELLED Time: 09:00 - 09:30 Appointment ID: APT2 Patient ID: P1001 Doctor ID: D001 Date: 2024-11-22 Service: CONSULTATION Status: CANCELLED Time: 11:00 - 11:30</pre>	Pass

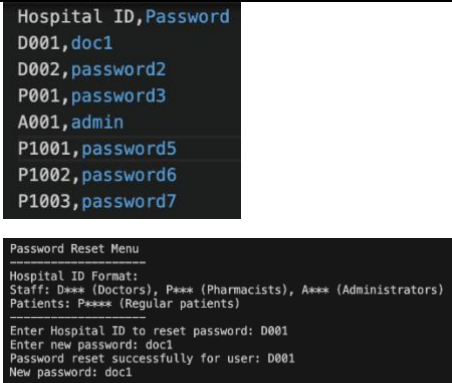
5.3 View and Manage Medication Inventory

ID	Input	Expected Output	Actual Output	Result
TC24	Add new medication to inventory, "Vitamin C", "20", "10"	Display updated medication inventory	<pre>Inventory Management Options: 1: Add new medication to inventory 2: Remove medication from inventory 3: Edit inventory quantities 4: Return to main menu 1 Enter name of new medicine: Vitamin C Enter initial stock quantity: 20 Set low stock threshold: 10 Successfully added new medication: Vitamin C Medication added successfully! Current Inventory Medicine: Paracetamol Current Stock: 100 Low Stock Alert Level: 20 Medicine: Ibuprofen Current Stock: 70 Low Stock Alert Level: 10 Medicine: Amoxicillin Current Stock: 10 Low Stock Alert Level: 15 (LOW STOCK!) Medicine: Vitamin C Current Stock: 20 Low Stock Alert Level: 10</pre>	Pass

5.4 Approve Medication Replenish Requests

ID	Input	Expected Output	Actual Output	Result
TC25	TC04, "Amoxicillin"	Changes status of request, updates medication inventory	<pre>Administrator Menu: 1: View and Manage Hospital Staff 2: View Appointments Details 3: View and Manage Medication Inventory 4: Approve Replenishment Requests 5: Logout Enter your choice (1-5): 4 Replenishment Requests: Ibuprofen: 20 units (Status: APPROVED) Paracetamol: 200 units (Status: PENDING) Amoxicillin: 10 units (Status: PENDING) Enter medicine name to approve (or 'exit' to return): Amoxicillin Approved replenishment of 10 units for Amoxicillin Replenishment request approved successfully!</pre>	Pass

5.5 Update User Password

ID	Input	Expected Output	Actual Output	Result
TC26	"D001", "doc1"	Changes user password on user_list.csv	 <pre>Hospital ID,Password D001,doc1 D002,password2 P001,password3 A001,admin P1001,password5 P1002,password6 P1003,password7 Password Reset Menu Hospital ID Format: Staff: D*** (Doctors), P*** (Pharmacists), A*** (Administrators) Patients: P***** (Regular patients) Enter Hospital ID to reset password: D001 Enter new password: doc1 Password reset successfully for user: D001 New password: doc1</pre>	Pass

Reflection

We initially planned to have a simple file structure where each class directly handled its own data access, UI elements, and business logic. However, this resulted in very messy code and file presentation, with many overlapping functionalities in different classes, and with classes working in silos. The code became difficult to maintain as changes in one area required updates across multiple files, and testing was challenging since functionalities were tightly coupled. As a result, we decided to adopt the Model-View-Controller (MVC) Architecture.

Furthermore, to incorporate design principles into our code, we intentionally created classes we felt would best exhibit the use of principles. For example, in order to demonstrate the Open-Closed Principle (OCP), we initially created a Person class extends our User class, and would be further extended by Patient and Pharmacist classes. However, this created redundancies as our Person class functionally served no purpose apart from forcing the design principle. As such, we pivoted back to the drawing board, and came up with an optimal class diagram that was clean and efficient, with appropriate usage of design principles.

This project was a memorable experience for us as it served to enhance our understanding and application of course content in object-oriented programming (OOP) in Java. Furthermore, the usage of external libraries such as CSV file-reading and writing allowed us to develop practical skills in development. We also appreciate the purpose of UML diagrams more, as it helped us stay organised with a clearer end goal in mind while coding the application. As a result, we have gained a deeper appreciation for the principles of OOP.