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
Amos Ang Chee Tiong (U2321010C)	SC2002	SCSI	 17/11/2024
Chong Zhi Yu (U2340412L)	SC2002	SCSI	17/11/2024
Tang Yingjie (U2322462K)	SC2002	SCSI	17/11/2024
Yeo Elijah Xuan-Ye (U2310904A)	SC2002	SCSI	17/11/2024
Ng Yuhang Dilon (U2323060K)	SC2002	SCSI	 17/11/2024

Important notes:

1. Name must **EXACTLY MATCH** the one printed on your Matriculation Card.
2. Student Code of Academic Conduct includes the latest guidelines on usage of Generative AI and any other guidelines as released by NTU.

Design Considerations: Overall Architecture and Navigation

State Machine Design Pattern

A State Machine Design was implemented. This helps to organise our code into states, where each state has their own classes. An example of this pattern is in our menu system. Each of our menus are states which define what options and functionalities are displayed and accessible. The App class is a State Machine which changes what it is currently displaying based on the current state loaded. As such, introducing new states doesn't require changing existing states classes / state machine context. This:

1) Reduces code complexity when handling switching from one state to another.
2) Improved Maintainability for better expandability

An **Entity-Control-Boundary Pattern** (*fig. 1*) was implemented through separation of user contexts, services and data layers. We went with this approach to enable role-based access control as different roles (e.g Doctors and Patients) have access to different kinds of information. This enabled us to use a loose coupling approach to reduce dependencies between classes.

Boundary Layer

In our app's case, the menu system served as the boundary layer and implemented a hierarchical structure with role-specific menus. They extend the structure to show role-specific options and to delegate the operations to appropriate services. In essence, handling menu views for the user based on their role after they login. Please refer to the figures below for reference (*fig. 2*).

Control Layer

Our App class serves as a central controller and manages entities and menu views (*fig. 3*). It coordinates between the UI and data layers to create a service lifestyle. This approach maintains user context and is thus able to handle navigation between the menus for each of our services.

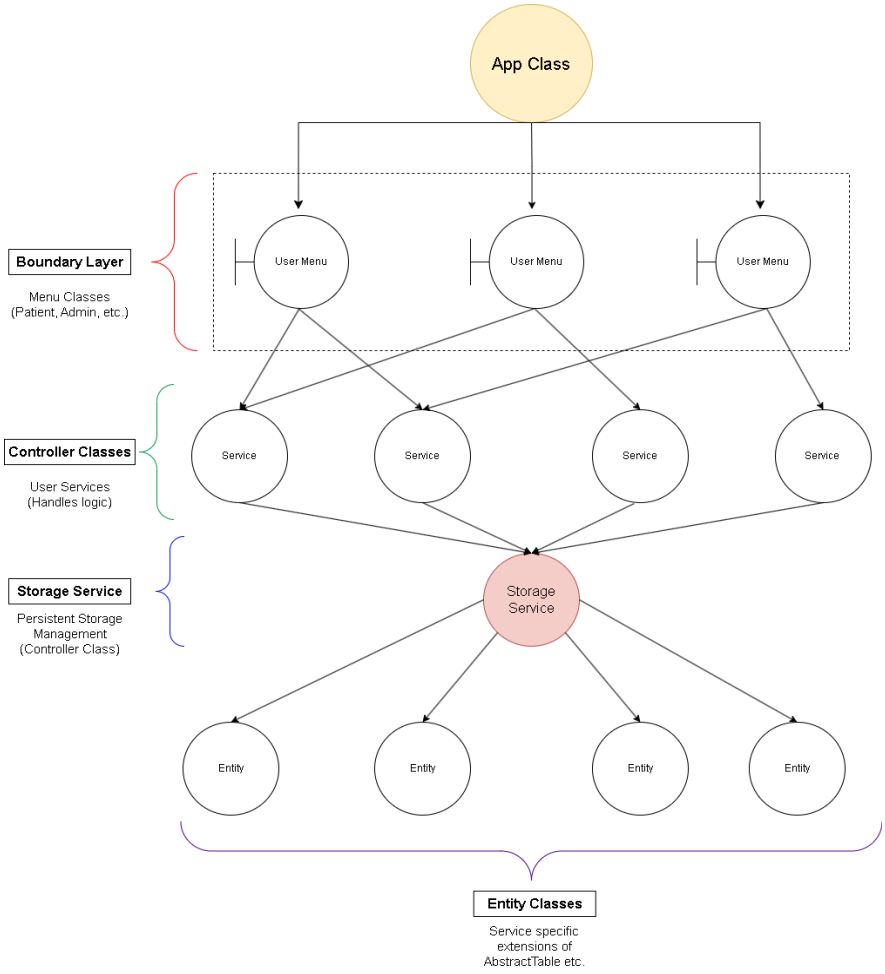


Fig 1. Entity-Control-Boundary Pattern

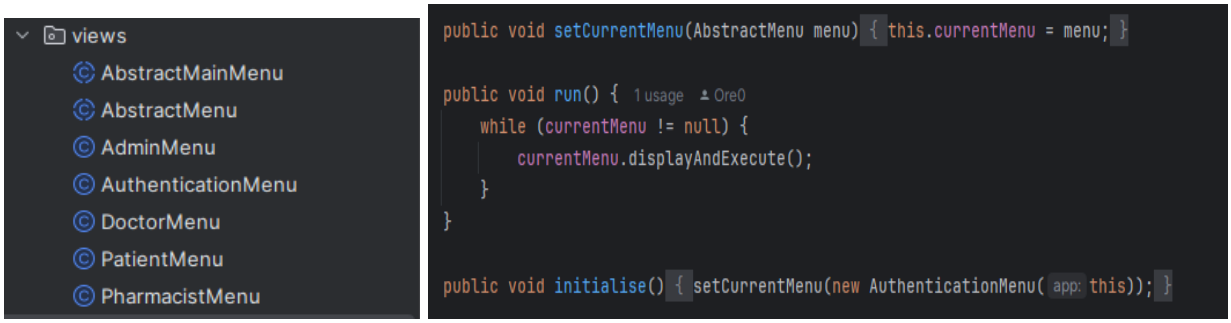


Fig 2. Menu Views

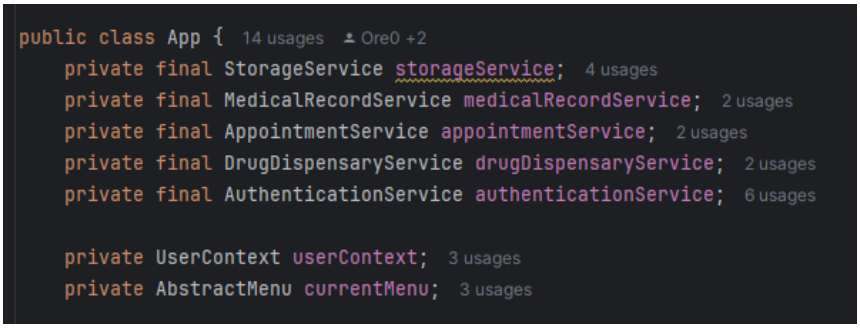


Fig 3. Controller View

Entity Layer

Our entities represent the core domain objects of our Hospital Management System (HMS) and also represent the main services which our app provides. They model the business data relationships present and hold the core functionality for services which our users use. These services include:

1. **Appointment**: Managing appointment scheduling and outcomes such as doctor schedule management and time slot allocation
2. **Authentication**: Handles user authentication and validation for role-based access control
3. **Drug Dispensary:** Manages drug inventory and stock levels as well as requests to dispense and replenish
4. **Medical Record:** Responsible for storing and retrieving patient medical histories + details, also allows patients and doctors to update role-specific details
5. **Staff Management**: Managing manpower related actions like adding, removing, archiving of staff details and assignment of role.

Data Management

AbstractTable

For persistent data management, an AbstractTable data management system was used for CRUD (Create, Read, Update and Delete) operations. The following 2 classes were used for this:

1. **AbstractTable.java:** Manages collections of AbstractTableEntry objects, functioning as a container for persistent data manipulation
2. **AbstractTableEntry.java:** Base class for table entries, focusing on individual rows in a data table

By implementing them as abstract classes, it allows for them to be specialised for varied use cases. This unifies our CSV serialisation for all entities. One example of this would be `filterByAttribute()`. For example when retrieving a patient's medical records from a CSV of medical records, `filterByAttribute()` can be used to access all entries of a specific patientID to find entries corresponding to a particular patient.

Extensibility of AbstractTable

Our system implements a **factory method pattern** in our AbstractTable system where AbstractTable and AbstractTableEntry provide the factory method interface. Due to its extensibility, new data types can be added by extending to base classes, making it very convenient to cater tables to new entities without having to modify a lot of the existing code. AbstractTable provides these extension points. Some of these include:

1. `createValidEntryTemplate()` for custom entry creation
2. `getHeaders()` for column definitions
3. `searchByAttribute()` for custom search functionality

Using `createValidEntryTemplate()` as an example, we can observe how this design pattern is used in DrugInventoryEntry:

```
/**
 * This creates an empty T extends AbstractTableEntry object
 * Only it's ID is set. Other fields to be instantiated.
 * Note that this SHOULD NOT ADD the entry. Call addEntry() for that.
 * @return
 */
protected abstract T createValidEntryTemplate(); 10 usages  4 implementations  Coziyu
```

```
@Override  10 usages  Coziyu
public DrugInventoryEntry createValidEntryTemplate() {
    return new DrugInventoryEntry(getUnusedID(),  name: null,  quantity: -1,  lowStockAlertThreshold: -1);
}
```

Fig 4. DrugInventoryEntry createValidEntryTemplate() Example

In this case, we can observe that AbstractTable provides an interface for `createValidEntryTemplate()` and under DrugInventoryEntry, it is overridden to fit the needs for the Drug Dispensary subsystem. This approach ensures consistency is present throughout different tables with uniform initialisation processes.

Extensibility and Maintainability of the Architecture

By adhering to the **SOLID Principles**, our code is both extensible and maintainable. Each of our classes focus on a singular well-defined responsibility and are separated accordingly. Our menus handle user interaction and role specific views while business logic is delegated to our subclasses

Open/Closed Principle (OCP)

Our system has a strong adherence to this principle and as such has good extensibility. Two key areas of implementation where this is seen are:

1. **New User Types**

- Adding new user types is fairly simple with the UserContext which has been implemented
    - Done by creating a new class created extending UserContext
    - Adding role-specific fields and methods
    - System automatically integrating new user types
- In practice, this is quite effective as new hospital roles (e.g Nurses, Neurosurgeons) can be added without modifying the core code of our system

2. **AbstractTable**

- Allows new table entry types to be created
- No changes need to be made to table handling logic
- Easy to add new types of records (e.g Medical Records, Drug Requests)

Liskov Substitution Principle (LSP)

Our derived classes can be used in place of their base classes. This is seen in our PatientContext extending from UserContext, where it maintains all UserContext functionality while adding patient-specific features. Consistent behaviour is thus maintained across inheritance hierarchies.

Interface Segregation Principle (ISP)

Specific interfaces were deployed for different functionalities (e.g IDrugStockDataInterface for drug inventory-related methods) and role-specific menu views only displayed the relevant option for a given role. Each of our services only depend on the specific interfaces they need as well.

```
public AppointmentService(IAppointmentDataInterface dataInterface) {  3 usages  Tyingjie
    this.storageServiceInterface = dataInterface;
    StorageService storageService = new StorageService();
    appointments = storageService.readAppointments();
    appointmentOutcomes = storageService.readAppointmentOutcomesFromCSV();
}
```

Fig 5. Example of ISP with Interface

Dependency Inversion Principle (DIP)

Our services depend on abstract interfaces rather than concrete implementations. Our data manipulation through our Abstract Table system as well as our menu system through our Abstract Menu system provided these interfaces which our subclasses extended.

Maintainability - Core Ideas

Class Hierarchical Organisation: Allows for logical grouping of related functionality.

- Consistent naming conventions (<ServiceName>Service, I<Interface Type>Interface)
- Clear inheritance hierarchies allows for easy navigation

Service Layer Isolation: By keeping the service layer isolated, we can modify each of them independently and add new ones easily. Maintaining service boundaries simplifies the testing and debugging process later on.

Loose Coupling: Aside from these, in our implementation, we had clear separation between functions. We kept user authentication separate from app logic, data storage is separate from data processing, and table operations are separate from entry definitions (AbstractTable system). Issues are easier to locate and isolate, allowing us to work on different components with fewer complications.

Additional Features

Staff Management Subsystem

Archiving Staff (A soft delete system)

- Organised search system
- Allows for inactive staff member records to be preserved
- ListsActiveStaff() and ListsAllStaff() to differentiate between the two

Menu Subsystem

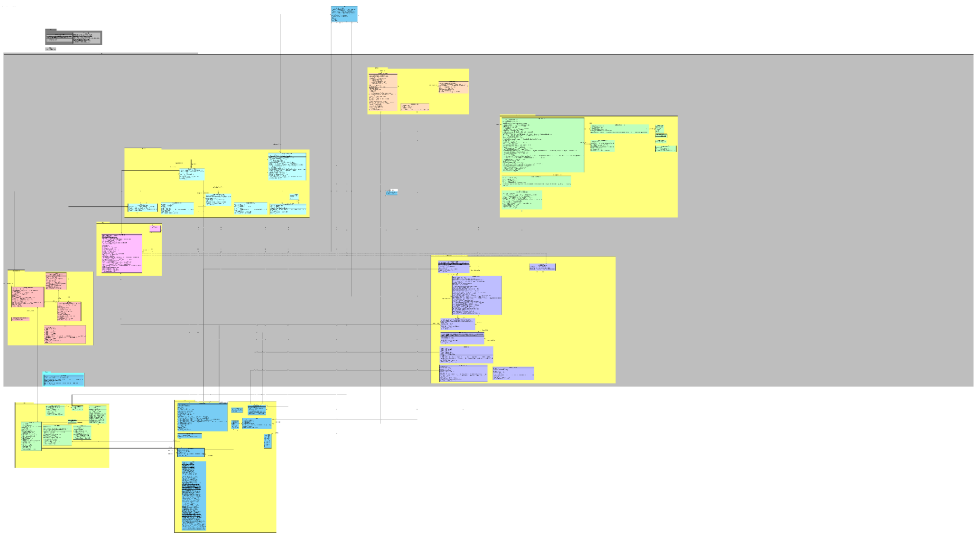
Audit Logging Process

- All actions performed Doctors, Pharmacists and Administrators recorded into a log file ensure that it is compliant with legal and ethical standards
- Data tracked includes Timestamps, Actions Performed, HospitalID for traceability
- Expedites the auditing process as CSV enables easy retrieval, filtering and analysis

Authentication/Login Subsystem

- Implemented Password Hashing to securely store user credentials and prevent unauthorised access
- Introduced Password Validation to enforce the use of strong and secure passwords
- Requires a combination of uppercase letters, lowercase letters, numbers, special characters, and a minimum password length
- Prevent attacks like dictionary or credential-stuffing attacks due to sensitivity of information
- Login Attempt Tracker helps to monitor failed login attempts

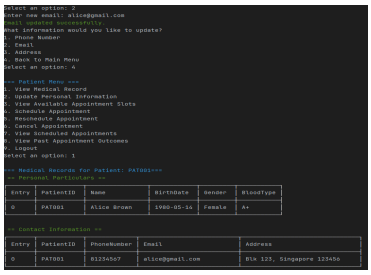
UML Class Diagram (*Please refer to github for clearer images)*



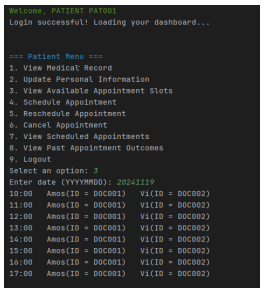
Test Cases (https://github.com/Coziyu/HospitalManagementSystem*)*

Test Case 1: View Medical Record

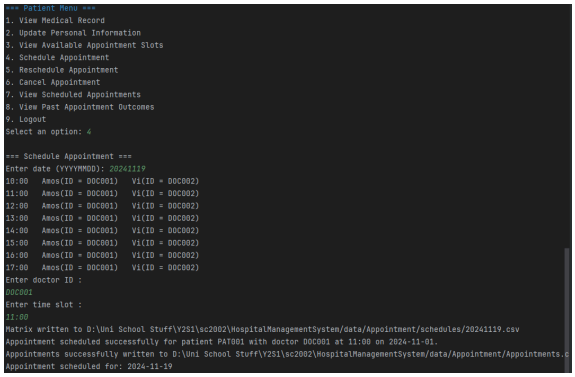
Test Case 2: Update Personal Information



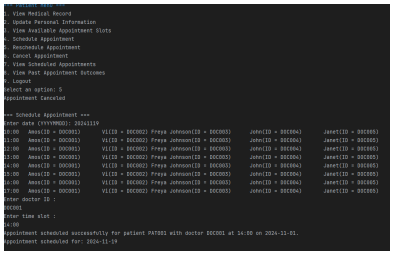
Test Case 3: View Available Appointment Slots

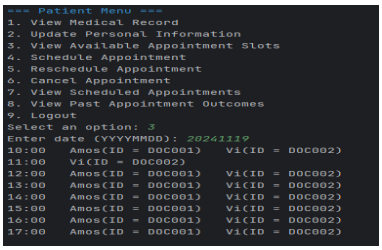


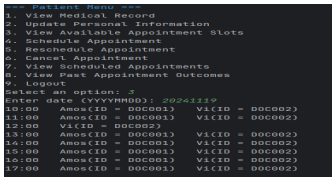
Test Case 4: Schedule an Appointment



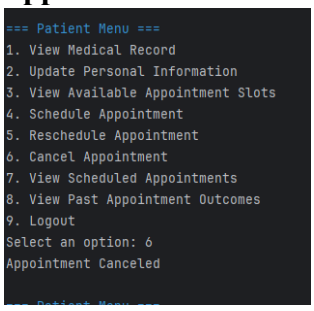
Test Case 5: Reschedule an Appointment



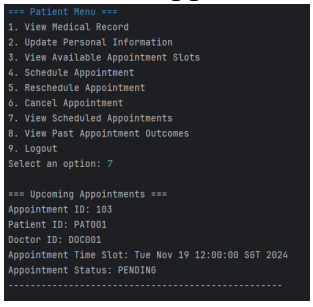




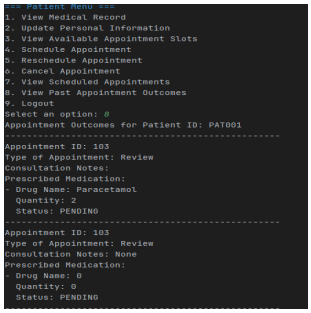
Test Case 6: Cancel an Appointment



Test Case 7: View Scheduled Appointments

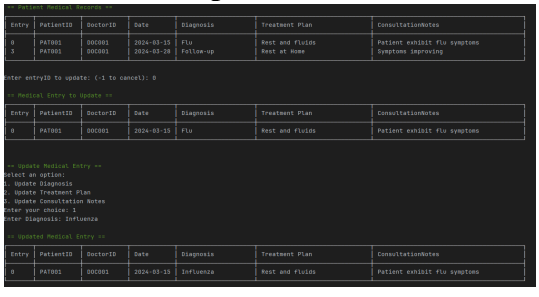


Test Case 8: View Past Appointment Outcome Records



Test Case 9: View Patient Medical Records

Test Case 10: Update Patient Medical Records



Test Case 11: View Personal Schedule

```
=== Doctor Menu ===
Doctor: Dr. DOC001
Hospital ID: DOC001
Date: 2024-11-17
1. View Patient Medical Records
2. Update Patient Medical Records
3. View Personal Schedule
4. Set Appointment Slot Availability
5. Handle Appointment Requests
6. View Upcoming Appointments
7. Record Appointment Outcome
8. Logout
Select an option: 3
Enter date (YYYYMMDD): 20241119

=== Daily Schedule ===
Schedule for: Dr. DOC001
Hospital ID: DOC001
LOG: 2024-11-17 13:58:54 - DOC001 (DOCTOR) - Viewed daily schedule
10:00   available
11:00   available
12:00   PAT001
13:00   available
14:00   available
15:00   available
16:00   available
17:00   available
```

Test Case 12: Set Availability for Appointments

```
=== Doctor Menu ===
Doctor: Dr. DOC001
Hospital ID: DOC001
Date: 2024-11-17
1. View Patient Medical Records
2. Update Patient Medical Records
3. View Personal Schedule
4. Set Appointment Slot Availability
5. Handle Appointment Requests
6. View Upcoming Appointments
7. Record Appointment Outcome
8. Logout
Select an option: 4
Do you want to set the schedule as:
1. Available
2. Unavailable
2
Enter the date (YYYYMMDD): 20241119
Enter the time slot (e.g., 12:00): 13:00
Schedule set to 'Unavailable' successfully.
```

```
Welcome, PATIENT PAT001
Login successful! Loading your dashboard...


=== Patient Menu ===
1. View Medical Record
2. Update Personal Information
3. View Available Appointment Slots
4. Schedule Appointment
5. Reschedule Appointment
6. Cancel Appointment
7. View Scheduled Appointments
8. View Past Appointment Outcomes
9. Logout
Select an option: 3
Enter date (YYYYMMDD): 20241119
10:00   Amos(ID = DOC001)   Vi(ID = DOC002)
11:00   Amos(ID = DOC001)   Vi(ID = DOC002)
12:00   Vi(ID = DOC002)
13:00   Vi(ID = DOC002)
14:00   Amos(ID = DOC001)   Vi(ID = DOC002)
15:00   Amos(ID = DOC001)   Vi(ID = DOC002)
16:00   Amos(ID = DOC001)   Vi(ID = DOC002)
17:00   Amos(ID = DOC001)   Vi(ID = DOC002)
```

Test Case 13: Accept or Decline Appointment Req

```
Doctor: Dr. DOC001
Hospital ID: DOC001
Date: 2024-11-17
1. View Patient Medical Records
2. Update Patient Medical Records
3. View Personal Schedule
4. Set Appointment Slot Availability
5. Handle Appointment Requests
6. View Upcoming Appointments
7. Record Appointment Outcome
8. Logout
Select an option: 5
Appointment ID: 104
Patient ID: PAT001
Doctor ID: DOC001
Appointment Time Slot: Tue Nov 19 14:00:00 SGT 2024
Appointment Status: PENDING
--------------------------------------------------
Enter the appointment ID
104
Appointment ID: 104
Patient ID: PAT001
Doctor ID: DOC001
Appointment Time Slot: Tue Nov 19 14:00:00 SGT 2024
Appointment Status: PENDING
--------------------------------------------------
Enter new status (CONFIRMED/CANCELLED): CONFIRMED
Appointment status updated successfully to CONFIRMED.
```

Test Case 14: View Upcoming Appointments

```
=== Doctor Menu ===
Doctor: Dr. DOC001
Hospital ID: DOC001
Date: 2024-11-17
1. View Patient Medical Records
2. Update Patient Medical Records
3. View Personal Schedule
4. Set Appointment Slot Availability
5. Handle Appointment Requests
6. View Upcoming Appointments
7. Record Appointment Outcome
8. Logout
Select an option: 6

--- Upcoming Appointments ---
Doctor: Dr. DOC001
All Confirmed Appointments for Doctor ID: DOC001
--------------------------------------------------
Appointment ID: 101
Patient ID: PAT001
Doctor ID: DOC001
Appointment Time Slot: Mon Nov 18 12:00:00 SGT 2024
Appointment Status: CONFIRMED
--------------------------------------------------
```

Test Case 15: Record Appointment Outcome

```
Select an option: 7
All Confirmed Appointments for Doctor ID: DOC001
--------------------------------------------------
Appointment ID: 101
Patient ID: PAT001
Doctor ID: DOC001
Appointment Time Slot: Mon Nov 18 12:00:00 SGT 2024
Appointment Status: CONFIRMED
--------------------------------------------------
Enter Appointment ID: 101
Enter number of medications to prescribe: 1
 == Available Drugs ==
```

ID	Name	Qty	lowStockQty
0	Paracetamol	175	80
1	Ibuprofen	47	50
2	Amoxicillin	75	15
3	Adderall	42	45
4	Ozempic	39	30

```
Enter name of drug 1: Paracetamol
Enter quantity for Paracetamol: 10
Enter Type of Appointment: Consultation
Enter Consultation Notes (use ',' and '/' if needed): NA
Appointment ID: 101 for Doctor ID: DOC001 has been completed.
```

```
Appointment Outcomes for Patient ID: PAT001
--------------------------------------------------
Appointment ID: 100
Type of Appointment: consult
Consultation Notes: Patient is aggresssive
Prescribed Medication:
- Drug Name: Paracetamol
  Quantity: 25
  Status: DISPENSED
- Drug Name: Ibuprofen
  Quantity: 50
  Status: PENDING
- Drug Name: Ozempic
  Quantity: 1
  Status: DISPENSED
--------------------------------------------------
Appointment ID: 101
Type of Appointment: Consultation
Consultation Notes: NA
Prescribed Medication:
- Drug Name: Paracetamol
  Quantity: 10
  Status: PENDING
--------------------------------------------------
```

Test Case 16: View Appointment Outcome

```
=== Pharmacist Menu ===
Pharmacist: PHARM001
Hospital ID: PHARM001
Date: 2024-11-17
1. View Appointment Outcome Records
2. Update Prescription Status
3. View Medication Inventory
4. Submit Replenishment Request
5. Logout
Select an option: 1

=== View Appointment Outcome Records ===
Recent Appointment Outcomes with Prescriptions:

Appointment ID: APT001
Date: 15/11/2024
Patient: P001
Doctor: Dr. Smith
Prescribed Medications:
1. Amoxicillin 500mg - Status: Pending
2. Paracetamol 500mg - Status: Dispensed
LOG: 2024-11-17 14:19:50 - PHARM001 (PHARMACIST) - Viewed appointment outcomes
```

Test Case 17: Update Prescription Status

```
Recent patients with pending Prescriptions:
0. PAT001
Select a patient (Enter the entry number):
0
 == Patient Particulars ==
```

Entry	PatientID	Name	BirthDate	Gender	BloodType
0	PAT001	Alice Brown	1980-05-14	Female	A+

```
 == Medical History ==
```

Entry	PatientID	DoctorID	Date	Diagnosis	Treatment Plan	ConsultationNotes
0	PAT001	DOC001	2024-03-15	Influenza	Rest and fluids	Patient exhibit flu symptoms
1	PAT001	DOC002	2024-03-20	Follow-up	Continue rest	Symptoms improving
3	PAT001	DOC001	2024-03-28	Follow-up	Rest at Home	Symptoms improving

```
 == Appointment Outcome ==
Appointment ID: 100
Type of Appointment: consult
Consultation Notes: Patient is aggresssive
 == Prescription Details ==
Drug Requested: Paracetamol, Quantity: 25,Status: PENDING
Dispense this prescription? (Y/N): y
Prescription dispensed successfully.
 == Prescription Details ==
Drug Requested: Ibuprofen, Quantity: 50,Status: PENDING
Dispense this prescription? (Y/N): y
Not enough stock for Ibuprofen.
Failed to dispense prescription.
 == Prescription Details ==
Drug Requested: Ozempic, Quantity: 1,Status: PENDING
Dispense this prescription? (Y/N): y
Prescription dispensed successfully.
```

Test Case 18: View Medication Inventory

```
Welcome, PHARMACIST PHARM001
Login successful! Loading your dashboard...


=== Pharmacist Menu ===
Pharmacist: PHARM001
Hospital ID: PHARM001
Date: 2024-11-17
1. View Appointment Outcome Records
2. Update Prescription Status
3. View Medication Inventory
4. Submit Replenishment Request
5. Logout
Select an option: 3

=== Medication Inventory ===
```

ID	Name	Qty	lowStockQty
0	Paracetamol	100	80
1	Ibuprofen	47	50
2	Amoxicillin	75	15
3	Adderall	42	45
4	Ozempic	20	30

Test Case 19: Submit Replenishment Request

```
 == Pharmacist Menu ==
Pharmacist: PHARM001
Hospital ID: PHARM001
Date: 2024-11-17
1. View Appointment Outcome Records
2. Update Prescription Status
3. View Medication Inventory
4. Submit Replenishment Request
5. Logout
Select an option: 4

=== Submit Replenishment Request ===

=== Medication Inventory ===
```

ID	Name	Qty	lowStockQty
0	Paracetamol	100	80
1	Ibuprofen	47	50
2	Amoxicillin	75	15
3	Adderall	42	45
4	Ozempic	20	30

```
Enter the ID of the medication to replenish:
1
Enter the quantity to request:
50
Enter replenishment notes (reason for request):
Low Stock
Successfully submitted replenishment request.
```

```
=== Administrator Menu ===
Logged in as: ADMIN001
Hospital ID: ADMIN001
Date: 2024-11-17
1. View and Manage Hospital Staff
2. View Scheduled Appointments Details
3. View and Manage Medication Inventory
4. Approve/Reject Replenishment Requests
5. Logout
Select an option: 4
=== Replenishment Requests ===
```

ID	Name	Qty	Notes
0	Ibuprofen	50	Flu season, demand for Ibuprofen is projected to increase.
1	Paracetamol	100	Paracetamol, the classic drug, is "almost always" in demand!
2	Ozempic	30	Low Stock

```
Select the replenishment request EntryID to approve: (-1 to go back) (a to approve all)
2
You have selected EntryID: 2. Approve or Reject?
1. Approve
2. Reject
Enter your choice: 1
Replenishment request approved successfully.
```

Test Case 20: View and Manage Hospital Staff

```
=== Add New Staff Member ===
Enter staff ID (eg. DOC001 or PHARM001): DOC010
Enter full name: Zac
Enter age: 30
Select staff role:
1. Doctor
2. Pharmacist
Enter choice (1 or 2): 1
Enter status (active or inactive): active
Enter gender (M/F): M
LOG: 2024-11-17 14:32:01 - ADMIN001 (ADMINISTRATOR) - Added new staff member: DOC010
Staff member added successfully!
```

```
=== Hospital Staff Management ===
Managing staff for Hospital: ADMIN001
1. Add New Staff Member
2. Update Staff Information
3. Remove Staff Member
4. View Staff List
5. Search for Staff
6. Back to Main Menu
Select an option: 3
=== Remove Staff Member ===
Enter staff ID to remove: DOC010
Y to mark as inactive or N to hard delete: N
LOG: 2024-11-17 14:32:21 - ADMIN001 (ADMINISTRATOR) - Removed staff member: DOC010
Staff member removed successfully!
```

```
=== Hospital Staff Management ===
Managing staff for Hospital: ADMIN001
1. Add New Staff Member
2. Update Staff Information
3. Remove Staff Member
4. View Staff List
5. Search for Staff
6. Back to Main Menu
Select an option: 2
=== Update Staff Information ===
Enter staff ID to update: DOC001
Enter new name (leave blank to keep current): Zac
Enter new age (leave blank to keep current):
Enter new role (Doctor/Pharmacist, leave blank to keep current):
LOG: 2024-11-17 14:32:45 - ADMIN001 (ADMINISTRATOR) - Updated staff member: DOC001
Staff information updated successfully!
```

```
=== Update Staff Information ===
Enter staff ID to update: DOC001
Enter new name (leave blank to keep current): Zac
Enter new age (leave blank to keep current):
Enter new role (Doctor/Pharmacist, leave blank to keep current):
LOG: 2024-11-17 14:32:45 - ADMIN001 (ADMINISTRATOR) - Updated staff member: DOC001
Staff information updated successfully!

=== Hospital Staff Management ===
Managing staff for Hospital: ADMIN001
1. Add New Staff Member
2. Update Staff Information
3. Remove Staff Member
4. View Staff List
5. Search for Staff
6. Back to Main Menu
Select an option: 4
=== Staff List ===
ID          Name                Age   Gender      Role        Status
DOC001      Zac                 23    M           Doctor      active
DOC002      Vi                  49    F           Doctor      active
PHARM001    John                33    M           Pharmacist  active
DOC003      Freya Johnson       23    F           Doctor      active
DOC004      John                27    M           Doctor      active
DOC005      Janet               10    F           Doctor      active
DOC006      Chris               66    M           Doctor      active
LOG: 2024-11-17 14:33:06 - ADMIN001 (ADMINISTRATOR) - Viewed staff list
```

```
=== Hospital Staff Management ===
Managing staff for Hospital: ADMIN001
1. Add New Staff Member
2. Update Staff Information
3. Remove Staff Member
4. View Staff List
5. Filter Staff
6. Back to Main Menu
Select an option: 5

=== Filter Staff ===
Filter by:
1. Role
2. Gender
3. Age Range
Enter your choice: 3
Enter minimum age: 21
Enter maximum age: 50
ID          Name                Age   Gender      Role        Status
DOC001      Amos                23    M           Doctor      active
DOC002      Vi                  49    F           Doctor      active
PHARM001    John                33    M           Pharmacist  active
DOC003      Freya Johnson       23    F           Doctor      active
DOC004      John                27    M           Doctor      active
```

Test Case 21: View Appointments Details

```
=== Administrator Menu ===
Logged in as: ADMIN001
Hospital ID: ADMIN001
Date: 2024-11-17
1. View and Manage Hospital Staff
2. View Scheduled Appointments Details
3. View and Manage Medication Inventory
4. Approve/Reject Replenishment Requests
5. Logout
Select an option: 2
Appointment ID: 100
Patient ID: PAT009
Doctor ID: DOC001
Appointment Time Slot: Sun Dec 01 12:00:00 SGT 2024
Appointment Status: CANCELLED
-----------------------------------------------
Appointment ID: 101
Patient ID: PAT001
Doctor ID: DOC001
Appointment Time Slot: Tue Nov 19 11:00:00 SGT 2024
Appointment Status: CANCELLED
-----------------------------------------------
Appointment ID: 102
Patient ID: PAT001
Doctor ID: DOC001
Appointment Time Slot: Tue Nov 19 12:00:00 SGT 2024
Appointment Status: CANCELLED
```

Test Case 22: View and Manage Medication Inventory

```
=== Administrator Menu ===
Logged in as: ADMIN001
Hospital ID: ADMIN001
Date: 2024-11-17
1. View and Manage Hospital Staff
2. View Scheduled Appointments Details
3. View and Manage Medication Inventory
4. Approve/Reject Replenishment Requests
5. Logout
Select an option: 3
=== Manage Drug Inventory ===
1. View Drug Stock
2. Add New Drug
3. Delete Drug
4. Update Drug Quantity
5. Update Low Stock Threshold
6. Back to Main Menu
Select an option: 1
=== Drug Inventory ===
| ID | Name | Qty | lowStockQty |
|---|---|---|---|
| 0 | Paracetamol | 100 | 80 |
| 1 | Ibuprofen | 47 | 50 |
| 2 | Amoxicillin | 75 | 15 |
| 3 | Adderall | 42 | 45 |
| 4 | Ozempic | 50 | 30 |
```

```
=== Update Drug Quantity ===
| ID | Name | Qty | lowStockQty |
|---|---|---|---|
| 0 | Paracetamol | 100 | 80 |
| 1 | Ibuprofen | 47 | 50 |
| 2 | Amoxicillin | 75 | 15 |
| 3 | Adderall | 42 | 45 |
| 4 | Ozempic | 50 | 30 |

Select the drug EntryID to update: (-1 to go back)
4
Enter the new quantity:
40
Quantity updated successfully.
=== Manage Drug Inventory ===
1. View Drug Stock
2. Add New Drug
3. Delete Drug
4. Update Drug Quantity
5. Update Low Stock Threshold
6. Back to Main Menu
Select an option: 1
=== Drug Inventory ===
| ID | Name | Qty | lowStockQty |
|---|---|---|---|
| 0 | Paracetamol | 100 | 80 |
| 1 | Ibuprofen | 47 | 50 |
| 2 | Amoxicillin | 75 | 15 |
| 3 | Adderall | 42 | 45 |
| 4 | Ozempic | 40 | 30 |
```

Test Case 23: Approve Replenishment Requests

```
=== Replenishment Requests ===
| ID | Name | Qty | Notes |
|---|---|---|---|
| 0 | Ibuprofen | 50 | Flu season, demand for Ibuprofen is projected to increase. |
| 1 | Paracetamol | 100 | Paracetamol, the classic drug, is *almost always* in demand! |

Select the replenishment request EntryID to approve: (-1 to go back) (a to approve all)
0
You have selected EntryID: 1. Approve or Reject?
1. Approve
2. Reject
Enter your choice: 1
Replenishment request approved successfully.
=== Replenishment Requests ===
| ID | Name | Qty | Notes |
|---|---|---|---|
| 0 | Ibuprofen | 50 | Flu season, demand for Ibuprofen is projected to increase. |
```

```
=== Administrator Menu ===
Logged in as: ADMIN001
Hospital ID: ADMIN001
Date: 2024-11-17
1. View and Manage Hospital Staff
2. View Scheduled Appointments Details
3. View and Manage Medication Inventory
4. Approve/Reject Replenishment Requests
5. Logout
Select an option: 3
=== Manage Drug Inventory ===
1. View Drug Stock
2. Add New Drug
3. Delete Drug
4. Update Drug Quantity
5. Update Low Stock Threshold
6. Back to Main Menu
Select an option: 1
=== Drug Inventory ===
| ID | Name | Qty | lowStockQty |
|---|---|---|---|
| 0 | Paracetamol | 200 | 80 |
| 1 | Ibuprofen | 47 | 50 |
| 2 | Amoxicillin | 75 | 15 |
| 3 | Adderall | 42 | 45 |
| 4 | Ozempic | 40 | 30 |
```

Reflection: Difficulties

As a group, we didn't communicate our intentions clearly enough and assumed we could tackle individual tasks and merge our work later. This led to issues like inconsistent use of variables (e.g., patientID and hospitalID as String or Int), requiring refactoring to ensure uniformity. Better communication could have avoided this. Our time allocation was also unrealistic—we initially set an optimistic timeline, believing we could stick to it without issues. In hindsight, we should've allowed more buffer time and anticipated the challenges typical of group projects.

This project deepened our understanding of Java concepts like classes, methods, and object-oriented programming, and gave us first-hand experience in collaborative programming. We learnt to use software engineering practices such as Github branching workflows, CI/CD, automated testing - including unit testing.

Before this course, coding in labs felt like an individual endeavour. However, this project highlighted the challenges of developing a larger application, stressing the need for long-term planning, modular design, and code maintainability.

Rather than relying on quick fixes, we learned to consider the broader impact of our design choices, embracing practices like modularity, maintainability, and clear documentation. Ultimately, this experience reshaped our view of software development —not just as a technical skill, but as a collaborative and strategic process.

Immediate improvements for the project include: **Multi-Factor Authentication**: Add an extra security layer during login, especially for high-privilege roles like doctors or administrators, to prevent unauthorised access. **Enhanced Audit Logging**: Expand logs to track not only actions but also anomalies or unauthorised access attempts, with real-time monitoring to alert administrators of critical actions. **Tiered Access Control for Staff**: Implement more granular access levels for different staff roles (e.g., nurses vs. administrators), ensuring staff access only the information they need, improving data security. Beyond the project, we could consider: **Centralised Database**: Replace the CSV storage with a relational database like MySQL to improve query performance, ensure data integrity, and enable complex reporting and analytics. **Machine Learning Integration for Predictive Analysis**: Leverage patient data to predict health complications, resource needs, and disease risk based on patient history.