



**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

SC2002 OBJECT ORIENTED DESIGN & PROGRAMMING

HOSPITAL MANAGEMENT SYSTEM (HMS)





AY 24/25 SEM 1 | SCSH GROUP 6

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 honoured 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 action may be taken.

Name	Course	Lab Group	Signature/Date
Heng Ziyang (U2322439H)	SC2002	SCSH	 14/11/24
Keagan Kong Kai Yi (Kang Kaiyi) (U2321702K)	SC2002	SCSH	 14/11/24
Lee Heng Sheng, Brandon (U2322900C)	SC2002	SCSH	 14/11/24
Lim Zhen Rong (U2320961J)	SC2002	SCSH	 14/11/24

1. Design Considerations

The Hospital Management System (HMS) is an application aimed at automating the management of hospital operations. The system is expected to facilitate efficient management of hospital resources, enhance patient care, and streamline administrative processes.

1.1 Assumptions Made

The following are the assumptions made while designing the HMS:

1. Only one user is using the HMS at one time.
2. There already exists a repository of all staff and Patient users, and inventory of Medicine.
3. All handling of patients are Appointment-based and there are no impromptu prescriptions and treatment plans given without a prior Appointment.
4. Doctor follows a standard 9am to 5pm schedule and Patient is not allowed to schedule Appointment outside of the time period, unless Doctor explicitly sets his availability to allow so. Anything outside those times will be considered an emergency and not fall under a regular Appointment.
5. Doctor will not have to prescribe a Medicine that the hospital does not have in its inventory.
6. The hospital has records and stock of all Medicine available. If the hospital currently does not own any of a particular Medicine, stock will be 0. Any new types of medicine to be added to the MedicineInventory will be manually added to the CSV file.

1.2 Design Approach

Extensibility and modularity was a key consideration in our design of HMS. We wanted our system to be easily extended with new features, such as adding new staff roles (Receptionist) as well as easily integrated with external systems. such as E-Payment software, or existing user databases. To achieve this, we employed the Entity-Control-Boundary (ECB) pattern. Entity objects encapsulate the business logic and data of our application. For example, a Patient may have attributes such as a **name**, **gender**, and **phoneNumber** to store Patient-related information for the application to function. Boundary objects act as the interface between the system and the users, similar to a User Interface (UI). It handles input/output operations, formats data for user interaction and is responsible for how the user interacts with the system. Control objects coordinate the flow of data between the Entity and Boundary objects. They manage navigational logic (choices that the user selects) and workflow.

The ECB pattern achieves loose decoupling via modularity. The Entity and Boundary objects are unaware of each other; communication is strictly performed through the Control object. This improves both extensibility and scalability: each layer adheres to the principle of Single Responsibility, and changes or additions to a Boundary object should not affect our business logic in the Entity object. It also allows for better code-reuse and code-organisation, which allows the system to be easily scaled effectively, as debugging and testing is straightforward. Loose coupling also helps us to adhere to the Open-Closed principle by extending existing code instead of modifying old code. Integration is now achieved more easily—we could entirely substitute our Boundary and replace it with a new one, provided it conforms to the existing API of the Control. In this manner, we achieve both our key considerations of extensibility and modularity.

1.2.1 Entity/Model Classes

Our Entity classes are the lowest layer of the HMS, where the core business logic of our system lies, encapsulating the attributes and behaviours of our domain objects. They are the only classes with the ability to retrieve, manage and modify these attributes. This allows the Entity classes to work independently of other layers or classes in our System. Some Entity classes in our code are the user classes of Patient, Doctor, Pharmacist and Administrator, or the record classes of MedicalRecord and AppointmentOutcomeRecord. These classes have the ability to return the data encapsulated in their attributes through their public methods.

1.2.2 Controller Classes

Our Controller classes act as the intermediary between our Entity/Model classes and our Boundary/View classes. They process inputs from the user received via Boundary/View and either **(a)** call another Controller class if more in-depth processing is needed; **(b)** call a Boundary/View class to display another prompt and receive more input; or **(c)** call the appropriate method in an Entity/Model class to retrieve, store, or edit data in its attributes.

All Controller classes inherit from a base Controller class in our code, which holds various Repositories as static final attributes and an abstract method **navigate()**, which Controllers must override. Some examples of Controller classes in our code include: **(a)**

DoctorMenuController and PatientMenuController for user menus; **(b)**

CompleteAppointmentController to mark an Appointment as completed and update the AppointmentOutcomeRecord; **(c)** SubmitReplenishmentRequestController, which is used by a Pharmacist to send a request to an Administrator to replenish a Medicine's stock.

1.2.3 Boundary/View Classes

Our Boundary/View classes are the highest layer of the HMS, forming the interface between the Controller and the user. These classes are called by Controllers to format and display options in menus, as well any information from Entity/Model classes. Boundary/View classes inherit from a base View class, which has an abstract `displayHeader()` method, to require them to display a header in its respective sub-menu. All Boundary/View classes make use of the `InputHandler` class and its static methods to receive input from the user. Some examples of our View classes include **(a)** `PatientMenuView` and `DoctorMenuView`, which display and gather inputs for Patient and Doctor menus; **(b)** `MedicalRecordView` and `ScheduleView`, which fetch relevant attributes from the `MedicalRecord` or `Schedule` to present to the user.

1.2.4 Exception Classes

These classes represent the runtime exceptions that can occur when the user keys in erroneous or unexpected input. These classes inherit from a base `Exception` class and are `Throwable`. They are chiefly used by the `InputHandler` as a form of handling input validation for user choices to ensure the robustness of the application. Some examples of our Exception classes include **(a)** `InvalidChoiceFormatException`, which is thrown when a user inputs an invalid choice format; **(b)** `InvalidChoiceValueException`, which is thrown when a user inputs an invalid choice value.

1.2.5 Serializer and Repository Classes

These classes are used to parse data from CSV files to retrieve information. `Serializer` is mainly used to handle file operations that relate to the serialization or encoding of Entity objects, while `Repository` holds a collection of Entity objects when they have been deserialized from the CSV files. Serializers all inherit from a base `Serializer` class; User Serializers inherit from a `UserSerializer` class which in turn is a generic subclass of `Serializer`. Repositories implement a base `Repository` interface, which provides a clean, uniform interface for performing CRUD (Create, Read, Update, Delete) operations on Entity objects. These Repositories act as databases; on startup, they call their corresponding `Serializer`. Some examples include **(a)** `DoctorRepository`, which contains a map of all Doctor IDs to Doctors; **(b)** `PatientRepository`, which contains a map of all Patient IDs to Patients.

1.2.6 Others

Other classes include Validation and Attribute enums. The Validation class is used to validate credentials, such as User ID and Password when the user logs in. It makes use of BCrypt to verify salted passwords. We represented some attributes such as BloodType and Gender as enums classes to represent a group of constant values.

1.3 Design Principles

1.3.1 Single Responsibility Principle (SRP)

SRP states that each class should have a clear singular purpose. Each class in HMS has a well-defined, singular purpose:

- Entity Classes: Encapsulates the business logic and data of our application of our domain objects.
- Business Classes: Handles user interaction, including input/output operations and formatting.
- Control Classes: Act as intermediaries between Entity and Boundary classes, processing user inputs and coordinating actions.
- Serializer Class: Handles file operations, such as reading and writing to CSV files.
- Repository Class: Provides a uniform interface for clean CRUD operations on collections of Entity objects.

This separation prevents any unintended ripple effect when changes are made.

1.3.2 Open-Closed Principle (OCP)

OCP states that each class should not be modified but can still be open for extension.

This is directly supported by our application architecture, where adding new roles such as Receptionist can be extended with the User class in the **entity** package. Existing User classes are not affected, while our application has been extended with a new User role. This goes for Repositories and Control classes as well, where each inherits from a base Controller class with some functionality and attributes. Replacing Boundary classes, such as integrating a new menu or UI, can be achieved by ensuring that the View conforms to existing Control APIs.

We also made use of generics to achieve this—our IUserRepository and UserSerializer classes enable easy code-reuse that eliminates explicit type-casting. They allow for inheritance of useful methods or attributes specific to Users.

1.3.3 Liskov Substitution Principle (LSP)

LSP states that all subclasses should be substitutable for their parent classes. You can see this through our base abstract classes, such as View, Controller and User. View subclasses are required to override **displayHeader()**, Controller subclasses are required to override **navigate()**, and User classes are required to contain some common User attributes and methods. In all cases, subclasses of these base classes are substitutable with their parent classes. You can further see this in 1.3.5., where we may declare a subclass as its base class in some methods.

1.3.4 Interface Segregation Principle (ISP)

ISP states that all classes that implement an interface should fully utilise each and every method defined in the interface. In our code, all Repository classes implement the IUserRepository interface and effectively implement the 4 methods defined.

1.3.5 Dependency Inversion Principle (DIP)

DIP states that higher level classes should not depend on concrete implementations of an object but rather on abstractions of it. This allows for decoupling.

High level classes do not need to know the implementation of lower level classes, but simply the function of the interface or superclass it calls, allowing higher flexibility and reduced coupling in our code. This is widely used in our application, where we (whenever possible) refer to objects by their base class or interface, rather than the object itself.

One example is the typical declaration of objects such as ArrayList or HashMap as their interface implementations, such as List or Map instead. This allows for open extension: a modification to the object type would not affect existing code as long as it extended the same superclass or implemented the same interface. For example, if I now wanted a LinkedList instead of an ArrayList, this change would not affect existing code that called List.

Another example may be found in our LoginController—here we declare the user as their base User class in some methods.

1.4 Object-Oriented Programming (OOP) Principles

1.4.1 Abstraction

In OOP, abstraction is the practice of hiding the precise implementation of an object from a user. One example of this is covered in 1.3.5, where we describe the examples of DIP in our

code, which is also a form of abstraction. We do not need to understand the details of how a method is implemented, but only the function of the method we are calling.

1.4.2 Encapsulation

Encapsulation is the practice of protecting all attributes of a class. Direct access to these attributes are restricted and are retrieved by getters and setters. In our code, this would look like the Doctor class which has a private **Schedule** that can only be fetched with the public **getSchedule()** method or added on to using the public **scheduleAppointment()** method.

1.4.3 Polymorphism

Polymorphism allows an object to have different implementations of a method in a subclass that it inherited from its parent. This can be seen in our CancelAppointmentView class, which inherits from the AppointmentView class, but overrides the **displayAppointments()** and **displayNoAppointments()** methods to display them slightly differently.

1.4.4 Inheritance

Our application makes heavy use of inheritance; many of our classes inherit from a superclass. Notable root classes are View, Controller and User, all which define useful common methods and attributes that subclasses may need.

1.5 New Features

1.5.1 Data Persistence (Deserialization)

When a user quits HMS, our App deserializes relevant objects into encoding data that is written to CSV files. This saves the state of the application and prevents the loss of data during downtime.

1.5.2 New Patient Registration

Our newly created Receptionist user type is able to register new Patients by entering relevant Patient data into the HMS to create a new Patient object. A newly registered Patient has their password set to the default password “password” and will be prompted to reset this password when they first log into the HMS.

1.5.3. Password Encryption

All passwords are salted with BCrypt when saved in the CSV files, which ensures security. Even if attackers gained access to the CSV files, decrypting the ciphertext would be extremely challenging.

1.5.4 Password Reset

In the event any user forgets their password, all Administrators have the ability to reset the user's password back to the default password "password". Much like when a new user is created, the affected user will be prompted to reset this password when they next log in.

1.5.5 Input Validation

Our application accounts for general input validation, where it does not crash when unexpected or erroneous input is provided by the user. This ensures a nice user experience.

It also accounts for email and password validation, where users are prompted to save a strong password and correct email format. For example, when a user changes his password, it is checked for the following conditions: (a) At least 8 characters. (b) At least 1 uppercase character. (c) At least 1 lowercase character. (d) At least 1 digit. (e) At least 1 special character. (e) Cannot contain any spaces or tabs. If any one of the conditions are unmet, the user is prompted to retype his new password.

1.5.6 Unit Testing

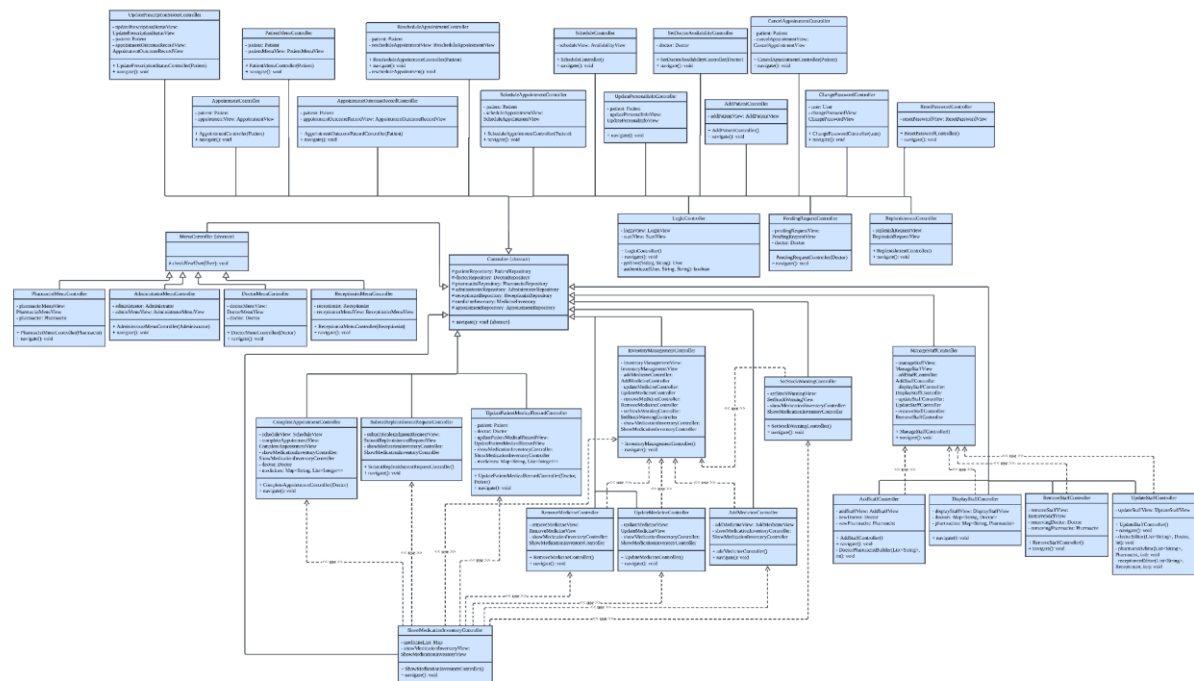
Testing of our code was performed for critical modules to ensure its correctness.

1.6 Reflection

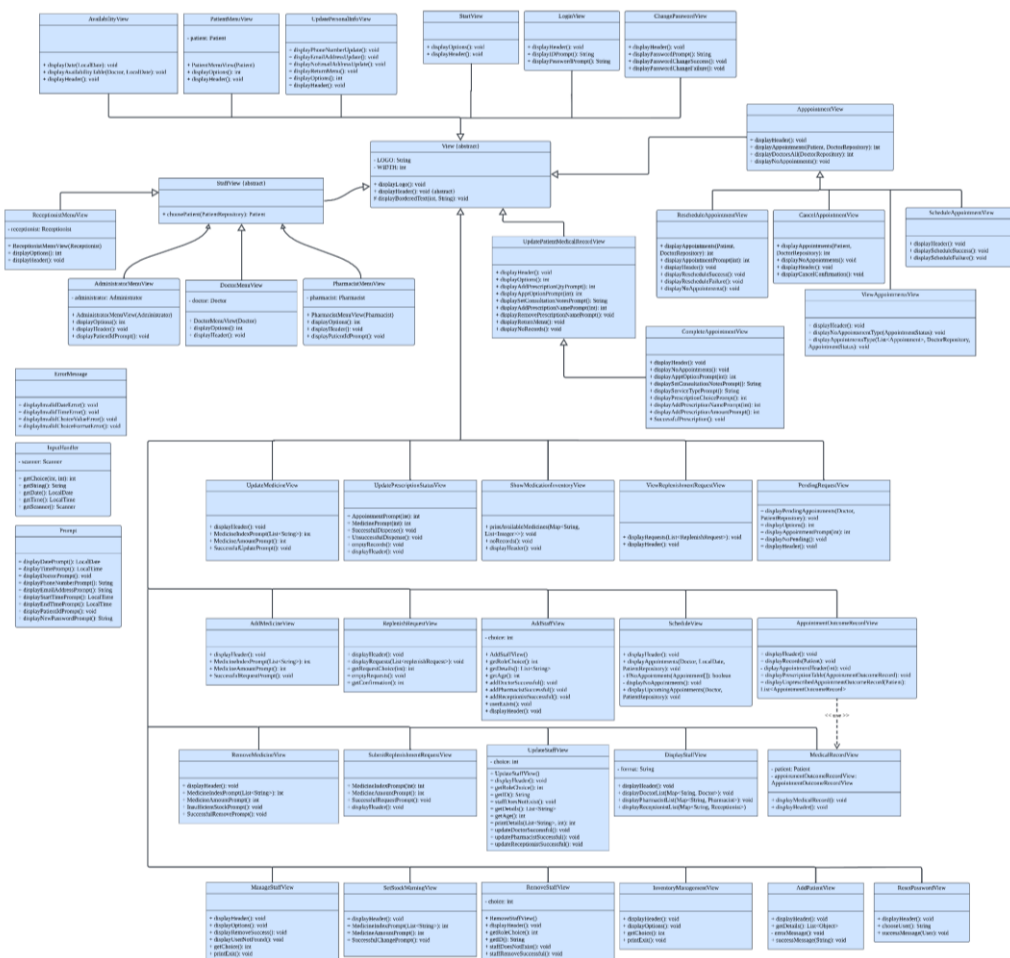
In this assignment, we gained a deeper appreciation of the OOP and design principles we learnt in this course. At the beginning of the project, we initially intended to only have a single class specific to each user type. However, we soon realised that was poor design that would lead to tight coupling and spaghetti code. Any small change would have a huge ripple effect which cascaded and caused our entire project to fail. This highlighted the importance of having good design and loose coupling in our project, leading us to adopt the ECB architecture. This improved both extensibility and maintainability.

Coding a system that had real-world applications also forced us to adopt a more user-centric approach to our code. By considering the point of view of potential users of HMS, we modified our design to be more user-friendly, as well as minimise and prevent any conflicts between users.

2.3 Controller Sub-Diagram



2.4 Boundary Sub-Diagram



3. Test Cases (All test cases can be found in test-cases.mp4 in the report folder.)

1

Medical Record

Patient Id: P1001
Name: Alice Brown
Date of Birth: 1980-05-14
Gender: Female
Phone Number: 1234567
Email Address: alice.brown@example.com
Blood Type: A+

Appointment Records

Appointment 1

UUID: 3a42a909-2b13-4d5d-9a3b-2fd543a668a4
Date: 2024-11-11
Service Type: X-ray
Diagnosis: Broken arm.

Medicine Prescribed

No medicine prescribed.

2

Medical Record

Patient Id: P1001
Name: Alice Brown
Date of Birth: 1980-05-14
Gender: Female
Phone Number: 81234567
Email Address: example@example.com
Blood Type: A+

3

Doctor: Dr. John Smith

Date

Time

Availability

2024-11-20

08:00

Available

2024-11-20

08:30

Available

2024-11-20

09:00

Available

2024-11-20

09:30

Available

2024-11-20

10:00

Available

2024-11-20

10:30

Available

2024-11-20

11:00

Available

2024-11-20

11:30

Available

2024-11-20

12:00

Available

2024-11-20

13:00

Available

2024-11-20

13:30

Available

2024-11-20

14:00

Available

2024-11-20

14:30

Available

2024-11-20

15:00

Available

2024-11-20

15:30

Available

2024-11-20

16:00

Available

2024-11-20

16:30

Available

4

Schedule Appointment

Please enter a date (YYYY-MM-DD): 2024-11-20
Note that the time must be in a half hour interval (e.g. 09:00, 09:30, etc).
Please enter a time (HH:MM): 16:00
Please select from the following doctors:
1. Dr. Emily Clarke (D002)
2. Dr. John Smith (D001)
1
Appointment scheduled.

5

Reschedule Appointment

Index

Date

Time

Status

Doctor Name

1

2024-11-24

16:00

[?] Pending

John Smith

2

2024-11-22

13:30

[?] Pending

Emily Clarke

3

2024-11-20

16:00

[?] Pending

Emily Clarke

Please select an appointment index:
5
Please enter a date (YYYY-MM-DD): 2024-11-20
Note that the time must be in a half hour interval (e.g. 09:00, 09:30, etc).
Please enter a time (HH:MM): 10:00
Please select from the following doctors:
1. Dr. Emily Clarke (D002)
2. Dr. John Smith (D001)
2
Appointment rescheduled.

6

Cancel Appointment

Index

Date

Time

Status

Doctor Name

1

2024-11-24

16:00

[?] Pending

John Smith

2

2024-11-22

13:30

[?] Pending

Emily Clarke

3

2024-11-20

10:00

[?] Pending

John Smith

Please choose an appointment to cancel:
3
Appointment has been cancelled.

7

Scheduled Appointment

Index

Date

Time

Status

Doctor Name

1

2024-11-24

16:00

[?] Pending

John Smith

2

2024-11-22

13:30

[?] Pending

Emily Clarke

8

Appointment Records

Appointment 1

UUID: 3a42a909-2b13-4d5d-9a3b-2fd543a668a4
Date: 2024-11-11
Service Type: X-ray
Diagnosis: Broken arm.

Medicine Prescribed

No medicine prescribed.

9

Enter patient ID: P1001

Medical Record

Patient Id: P1001
Name: Alice Brown
Date of Birth: 1980-05-14
Gender: Female
Phone Number: 81234567
Email Address: example@example.com
Blood Type: A+

Appointment Records

Appointment 1

UUID: 3a42a909-2b13-4d5d-9a3b-2fd543a668a4
Date: 2024-11-11
Service Type: X-ray
Diagnosis: Broken arm.

Medicine Prescribed

No medicine prescribed.

10

Appointment 2

UUID: 94aa0595-d22c-48d0-a898-0bfdf2e5cd0
Date: 2024-11-17
Service Type: Consultation
Diagnosis: Fever, sore throat. Drink more water.

Medicine Prescribed

Index

Medicine Name

Amount

Status

1

Ibuprofen

5

[?] Pending

2

Paracetamol

12

[?] Pending

11

Please enter a date (YYYY-MM-DD): 2024-11-20

Schedule

Index

Date

Time

Status

Patient Name

No appointments scheduled on this date.

12

Please enter a date (YYYY-MM-DD): 2024-11-22
Choose a start time. Note that the time must be in a half hour interval (e.g. 09:00, 09:30, etc).
Please enter a time (HH:MM): 17:00
Choose a end time. Note that the time must be in a half hour interval (e.g. 09:00, 09:30, etc).
Please enter a time (HH:MM): 17:30

13

Index

Date

Time

Status

Patient Name

1

2024-11-24

16:00

[?] Pending

Alice Brown

Please select an appointment index:
1
Please select an option:
1. Confirm
2. Cancel
1

14

Please enter a date (YYYY-MM-DD): 2024-11-24

Schedule

Index

Date

Time

Status

Patient Name

1

2024-11-24

16:00

[+] Confirmed

Alice Brown

Index

Date

Time

Status

Patient Name

1

2024-11-24

16:00

[+] Confirmed

Alice Brown

15	<pre> ===== Complete Appointment ===== Index Date Time Status Patient Name ----- ----- ----- ----- ----- 1 2024-11-24 16:00 [+] Confirmed Alice Brown Please enter the appointment index: 1 Enter service type: Consultation Enter consultation notes: Coughing and runny nose. Enter number of prescriptions to give (0-999): 1 ===== Medication Inventory ===== Available Medicines: Index Medicine Name Amount Warning ----- ----- ----- ----- 1 Paracetamol 100 OK 2 Ibuprofen 50 OK 3 Amoxicillin 75 OK Enter index of Medicine to be added: 3 Enter amount of Medicine to be prescribed (1-999): 16 Medicine prescribed successfully </pre>	16	<pre> ===== Appointment 3 ===== UUID: c3dcb872-e970-4974-a31d-2ca9d8fbaffc Date: 2024-11-24 Service Type: Consultation Diagnosis: Coughing and runny nose. ===== Medicine Prescribed ===== Index Medicine Name Amount Status ----- ----- ----- ----- 1 Amoxicillin 16 [?] Pending </pre>
17	<pre> ===== Appointment 3 ===== UUID: c3dcb872-e970-4974-a31d-2ca9d8fbaffc Date: 2024-11-24 Service Type: Consultation Diagnosis: Coughing and runny nose. ===== Medicine Prescribed ===== Index Medicine Name Amount Status ----- ----- ----- ----- 1 Amoxicillin 16 [X] Dispensed </pre>	18	<pre> ===== Medication Inventory ===== Available Medicines: Index Medicine Name Amount Warning ----- ----- ----- ----- 1 Paracetamol 100 OK 2 Ibuprofen 50 OK 3 Amoxicillin 59 OK </pre>
19	<pre> ===== Submit Replenishment Request ===== ===== Medication Inventory ===== Available Medicines: Index Medicine Name Amount Warning ----- ----- ----- ----- 1 Paracetamol 100 OK 2 Ibuprofen 50 OK 3 Amoxicillin 59 OK Enter Index of Medicine to be replenished: 3 Enter Amount of Medicine to be replenished (1-999): 60 Request successfully made ===== View Replenishment Requests ===== Active Pending Requests: Index Medicine Name Amount ----- ----- ----- 1 Amoxicillin 60 </pre>	20	<pre> ===== Add Staff ===== Choose role (Doctor/Pharmacist/Receptionist): Hospital: 1: Doctor 2: Pharmacist 3: Receptionist 4: Cancel 5: Doctors: Index ID Doctor Name Enter ID: ----- ----- ----- ----- 1 D002 Emily Clarke P001 2 D001 John Smith Enter Name: 3 Enter Gender: 4 1: Male 5 2: Female 6 Enter Age: 7 72 Pharmacists: Index ID Pharmacist Name ----- ----- ----- 1 P001 Mark Lee Receptionists: No receptionist in records. Receptionist added successfully. </pre>
21	<pre> Enter patient ID: P1001 ===== All Appointments ===== [?] Pending appointments: Index Date Time Doctor ID Doctor Name ----- ----- ----- ----- ----- 1 2024-11-22 13:30 D002 Emily Clarke [+] Confirmed appointments: Index Date Time Doctor ID Doctor Name ----- ----- ----- ----- ----- 1 2024-11-20 16:00 D002 Emily Clarke 2 2024-11-20 10:00 D001 John Smith [!] Cancelled appointments: Index Date Time Doctor ID Doctor Name ----- ----- ----- ----- ----- 1 2024-11-20 16:00 D002 Emily Clarke 2 2024-11-20 10:00 D001 John Smith [X] Completed appointments: Index Date Time Doctor ID Doctor Name ----- ----- ----- ----- ----- 1 2024-11-11 09:00 D002 Emily Clarke Service Type: X-ray Diagnosis: Broken arm. ===== Medicine Prescribed ===== No medicine prescribed. 2 2024-11-17 16:00 D001 John Smith Service Type: Consultation Diagnosis: Fever, sore throat. Drink more water. ===== Medicine Prescribed ===== Index Medicine Name Amount Status ----- ----- ----- ----- 1 Ibuprofen 5 [?] Pending 2 Paracetamol 12 [?] Pending 3 2024-11-24 16:00 D001 John Smith </pre>	22	<pre> ===== Medication Inventory ===== Available Medicines: Index Medicine Name Amount Warning ----- ----- ----- ----- 1 Paracetamol 100 OK 2 Ibuprofen 50 OK 3 Amoxicillin 59 OK ===== Medication Inventory ===== Available Medicines: Index Medicine Name Amount Warning ----- ----- ----- ----- 1 Paracetamol 100 OK 2 Ibuprofen 50 OK 3 Amoxicillin 59 OK Enter Index of Medicine to be set: 1 Enter amount of currently available Medicine to be set at: 50 Medicine successfully updated ===== Medication Inventory ===== Available Medicines: Index Medicine Name Amount Warning ----- ----- ----- ----- 1 Paracetamol 50 OK 2 Ibuprofen 50 OK 3 Amoxicillin 59 OK </pre>
23	<pre> ===== Replenish Request ===== Index Medicine Name Amount ----- ----- ----- 1 Amoxicillin 60 Would you like to approve or deny this request? 1. Approve 2. Deny 3. Cancel 1 ===== Medication Inventory ===== Available Medicines: Index Medicine Name Amount Warning ----- ----- ----- ----- 1 Paracetamol 50 OK 2 Ibuprofen 50 OK 3 Amoxicillin 119 OK </pre>	25 26	<pre> ===== HMS ===== ===== Login ===== Enter id: D001 Enter password: Password@13 Authentication failed. Please try again. ===== Login ===== Enter id: P001 Enter password: password Welcome to HMS. Please enter your new password: Password@123 [X] Password contains at least 8 characters long. [X] Password contains at least one uppercase letter. [X] Password contains at least one lowercase letter. [X] Password contains at least one digit. [X] Password contains at least one special character (!, @, #, \$, %, ^, &, *). [X] Password does not contain any spaces or tabs. [X] Password is valid. </pre>