

SC2002 OBJECT ORIENTED DESIGN & PROGRAMMING HOSPITAL MANAGEMENT SYSTEM

Report of Project Structure Design & Functionality AY24/25

Sem 1 | SCS5 Group 3

Additional Links:

Project Main Page : https://kuroinit.github.io/SC2002_Assignment/#

Github Main Page : https://github.com/KuroInit/SC2002_Assignment

Youtube Video: <https://www.youtube.com/watch?v=mmMCV20Z7zc&feature=youtu.be>





Project Document : https://kuroinit.github.io/SC2002_Assignment/javadoc.html

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 the 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
Ashwin Suresh U2321541F	SC2002	SCS5	
Chawla Preet Singh U2323574A	SC2002	SCS5	
Venkatesh Arun Moorthy U2323092H	SC2002	SCS5	
Vishesh Vinod Jain U2323363G	SC2002	SCS5	

1 DESIGN CONSIDERATIONS

HMS (Hospital Management System) is a robust and scalable Java console application crafted with a strong emphasis on modularity, extensibility, and long-term maintainability. Designed to automate and enhance hospital operations, HMS centralizes critical functions like patient management, appointment scheduling, staff coordination, and inventory tracking. The system is tailored to accommodate diverse user roles—patients, doctors, pharmacists, and administrators—each empowered with role-specific capabilities to ensure smooth, secure, and efficient workflows. With its adaptable architecture, HMS seamlessly supports future upgrades and integrations, making it an ideal solution for dynamic healthcare environments.

1.1 DESIGN APPROACH

The design approach is guided by the **MVC architecture**, which separates responsibilities across three core layers:

1. **Model:** Represents the core data and business logic (e.g., users, appointments, feedback, and records).
2. **View:** Handles the user interface for interactions, such as displaying menus and collecting user inputs.
3. **Controller:** Acts as the intermediary, processing user inputs, invoking model methods, and updating views.

The design ensures **separation of concerns**, allowing each layer to operate independently while interacting seamlessly.

1.2 HIGHLIGHTS OF SOME DESIGNS

1. Role-Specific Capabilities

- **Purpose:** Tailors the system for patients, doctors, pharmacists, and administrators.
- **Implementation:** Each user role has access to customized menus and features, ensuring targeted workflows.
- **Advantage:** Enhances usability and efficiency by reducing clutter and providing role-relevant functionalities.
- **Trade-Off:** Development complexity increases with role-specific logic and views.

2. Password Encryption and Hiding

- **Purpose:** Strengthens authentication security.
- **Implementation:** Utilizes SHA-256 hashing for password encryption and hides input during entry with `Console.readPassword()`.
- **Advantage:** Prevents password leaks, ensures secure storage, and improves user confidence in the system.
- **Trade-Off:** Increased computation overhead during authentication.

3. Appointment Management with No Sunday Scheduling

- **Purpose:** Ensures proper scheduling while respecting staff availability.
- **Implementation:** Excludes Sundays from appointment slots and validates input for compliance.
- **Advantage:** Balances operational efficiency with work-life balance for hospital staff.
- **Trade-Off:** Potential inconvenience for urgent cases.

4. Feedback System

- **Purpose:** Collects user input for service improvements.
- **Implementation:** Users can submit feedback and ratings, stored in a CSV file for analysis.
- **Advantage:** Provides actionable insights to refine user experience and service quality.
- **Trade-Off:** Scalability issues may arise with increased feedback volume.

5. Dynamic Data Management with CSV

- **Purpose:** Enables efficient storage and retrieval of hospital data.
- **Implementation:** All critical data, such as patient details, appointments, and inventory, are stored in CSV files for easy batch processing.
- **Advantage:** Simplifies data management, ensuring scalability and ease of updates.

6. Factory Design Pattern for Object Creation

- **Purpose:** Streamlines object creation processes for system entities.
- **Implementation:** Factory methods generate user-specific objects dynamically, adhering to their roles and attributes.
- **Advantage:** Enhances scalability and adaptability for future feature additions.
- **Trade-Off:** Requires more memory and design complexity for multiple factories.

7. Improved User Registration Flow

- **Purpose:** Facilitates a seamless and secure registration process for new users.
- **Implementation:** Automatically generates unique IDs, stores hashed passwords, and adds user details to the appropriate CSV files during registration.
- **Advantage:** Provides a streamlined and secure onboarding experience, ensuring data consistency and security.

1.3 APPLIED DESIGN PRINCIPLE

1.3.1 Single Responsibility Principle (SRP)

- **Purpose:** ○ To ensure that a class has only one reason to change, i.e., it focuses on a single responsibility, which simplifies code maintenance and improves readability.

Implementation:

- Controllers:
 - DoctorController, PatientController, AdministratorController, and PharmacistController each focus on managing their respective role-specific logic.
 - For example, DoctorController manages the doctor's schedule, availability, and appointment updates, while PatientController handles patient feedback and record viewing.
- Views:
 - DoctorView, PatientView, AdministratorView, and PharmacistView are solely responsible for displaying role-specific information.
 - No business logic is embedded in the views.
- Utilities
 - Obfuscation is dedicated to password encryption and validation, encapsulating cryptographic logic in one place
- **Advantage:**
 - Simplifies debugging and testing as each class has a focused responsibility.
 - Enhances maintainability, as changes in one functionality (e.g., updating the appointment system) do not affect unrelated parts of the application.

1.3.2 Open-Closed Principle (OCP)

- **Purpose:**
- To ensure that the system is open for extension but closed for modification, allowing the addition of new features without altering existing code.
- **Implementation:**
- Role-based Architecture:
 - Adding a new role (e.g., Nurse) requires creating a new controller (NurseController) and view (NurseView) without modifying existing roles.
- Dynamic Menu Handling:
 - The Main class dynamically interacts with user roles based on their login credentials. Adding new roles extends the switch-case structure without altering existing conditions.
- Utilities and Abstract Methods:
 - Utility functions like readDataFromFile and writeDataToFile are designed to handle generic file operations, making them reusable and extendable without modification.
- **Advantage:**
- Encourages scalability by supporting new functionalities (e.g., inventory tracking for a new department) without disrupting existing code.
- Minimizes risk during upgrades or extensions.

1.3.3 Liskov Substitution Principle (LSP)

- **Purpose:** ○ To ensure that subclasses can substitute for their base classes without altering the correctness of the program, supporting polymorphism and consistent behavior.
- **Implementation:**
- Role-based Controllers and Views:
 - Subclasses like DoctorController and PatientController are used interchangeably where controllers are required.
 - Views like DoctorView and PatientView follow the same principle, allowing the system to handle different user roles dynamically.
- Polymorphism:
 - UserController dynamically identifies the role (Doctor, Patient, Administrator) and routes it to the respective controller.
- Dynamic Method Overriding:
 - Each controller implements role-specific methods, such as DoctorController for viewSchedule and PatientController for submitFeedback, while adhering to the base class structure.
- **Advantage:**
- Simplifies user-role management, allowing seamless interactions with various roles without hardcoding behaviors.
- Reduces redundancy by leveraging shared base functionality while supporting role-specific customizations.

1.3.4 Interface Segregation Principle (ISP)

- **Purpose:**
- To ensure that classes implement only the methods they need, avoiding the inclusion of unnecessary methods that may not be relevant to certain classes.
- **Implementation:**
- Role-Specific Methodology:
 - Doctors can view and manage their schedules, while patients cannot.
 - Patients can provide feedback, while administrators focus on staff and inventory management.
- Specialized Controllers:
 - DoctorController handles appointment and schedule-related tasks, while PharmacistController focuses on inventory and prescription management.
 - Role-specific models (DoctorModel, PatientModel) include only the attributes and methods required for their functionality.
- **Advantage:**
- Reduces complexity by ensuring each class or interface is focused on its specific functionality.
- Simplifies testing and debugging as there is no risk of unused methods causing issues.

1.3.5 Dependency Inversion Principle (DIP)

- **Purpose:**

- To decouple high-level modules from low-level modules by relying on abstractions, promoting flexibility and reducing tight coupling.
- **Implementation:**
 - Controllers and Models:
 - Controllers like DoctorController depend on abstractions (DoctorModel) instead of tightly coupling with concrete classes.
 - Utility Abstraction:
 - Utilities such as Obfuscation and file-handling methods (readDataFromFile, writeDataToFile) are used independently by multiple components without direct dependency on specific controllers or views.
- Main Class:
 - The Main class interacts with abstracted controllers (UserController, DoctorController) without being tied to specific implementation details.
- **Advantage:**
 - Enhances flexibility by allowing changes in low-level modules (e.g., updating CSV handling to a database) without impacting high-level modules.
 - Promotes modularity and supports better testing by mocking dependencies.

1.4 FURTHER ENHANCEMENT

To further enhance our Hospital Management System (HMS), we envision implementing an advanced scheduling feature that dynamically manages appointment limits for each doctor. This feature would monitor doctors' schedules and automatically restrict appointment requests once their slots reach a predefined capacity. By providing prompt feedback to patients on the availability of doctors, this enhancement would help maintain balanced schedules, prevent overbooking, and improve patient access to alternate scheduling options.

Another proposed improvement focuses on refining the user interface by preemptively displaying pertinent data before requesting user inputs. For instance, when scheduling or updating patient information, HMS could display available appointment slots, current records, or medication statuses. This context-based guidance would reduce input errors and make the system more intuitive, enhancing the overall user experience and ensuring that users interact efficiently with the application.

Finally, an integrated notification system would further optimize communication within the hospital. Doctors, patients, and pharmacists could receive notifications regarding appointment status changes, prescription updates, and stock replenishment statuses. This feature would promote seamless communication and keep all users informed, thus fostering a collaborative environment in the hospital and ensuring timely, informed decision-making for patient care.

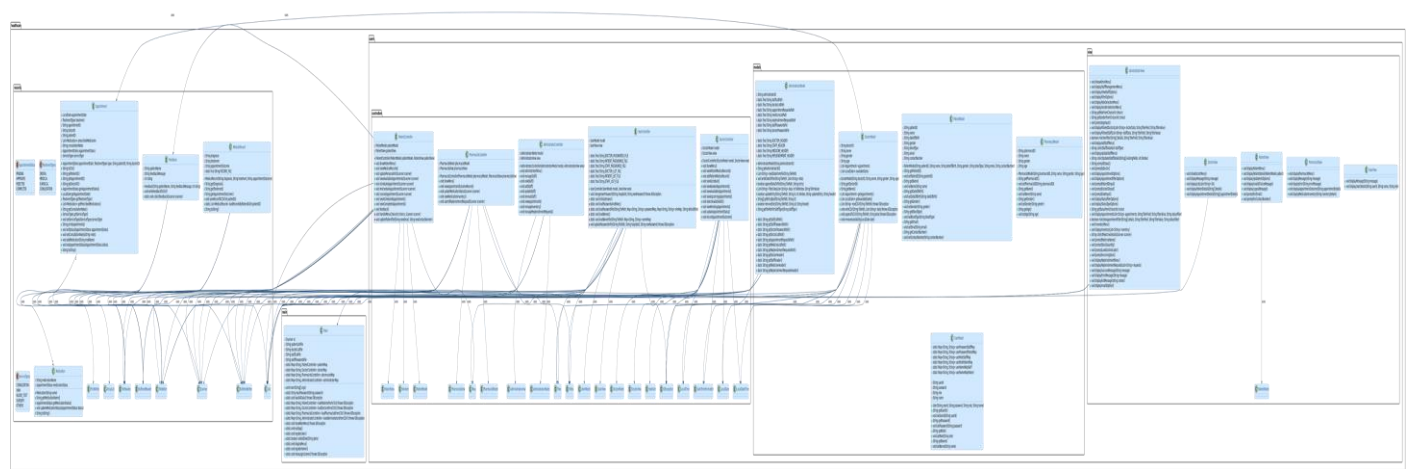
1.5 REFLECTION

From this assignment, we have seen the importance of design principles through real application. At the beginning phase of this assignment, we found that a slight change in our code would trigger a ripple effect,

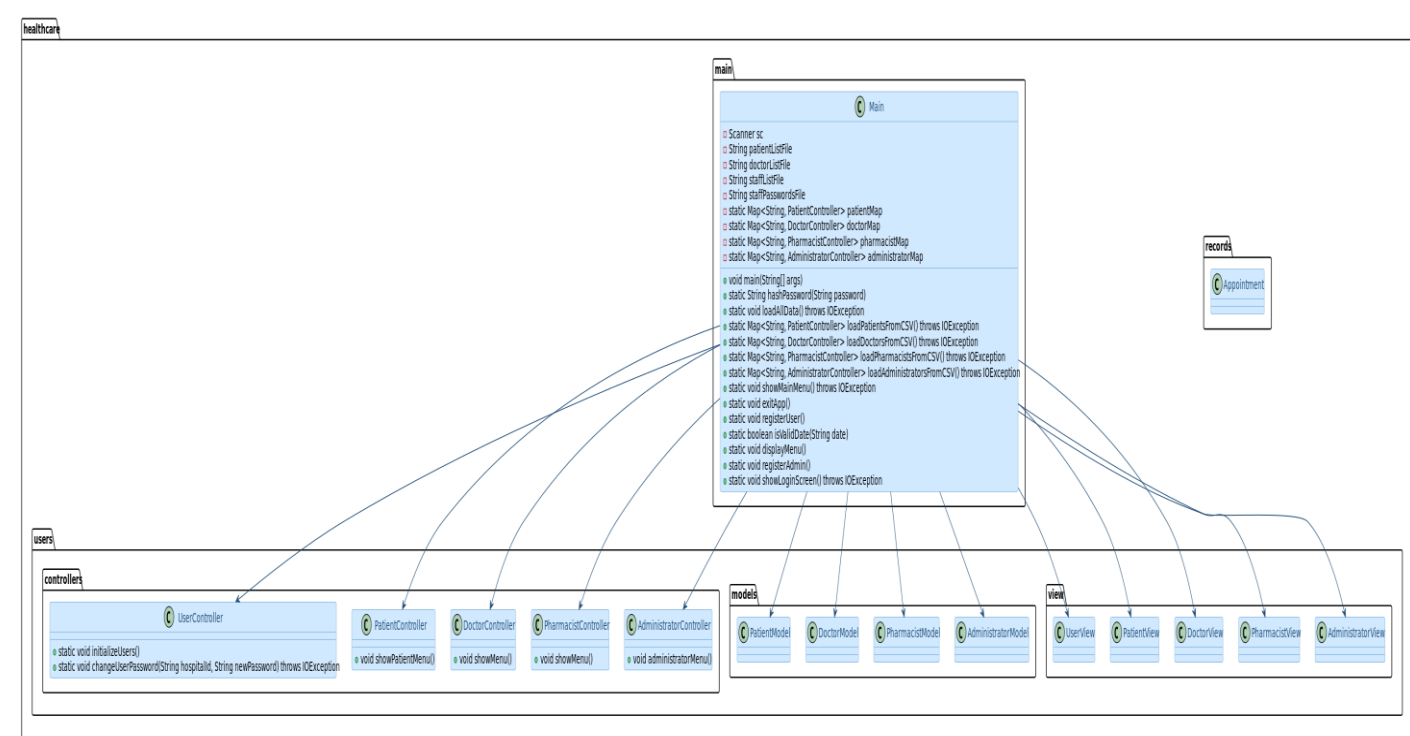
causing almost all other parts of our code to require adjustments. We then referred to design principles and applied them widely in our assignment. By implementing principles such as single responsibility, openclosed, and interface segregation, we minimized dependencies and ensured modularity.

Therefore, software with high cohesion and loose coupling is ultimately important, as it makes the system highly flexible, easily maintainable, and extendable. At the same time, we have learned to design software that fits its functions and real-world applications. Considering all users of the software, we continuously modified our design to accommodate different possible scenarios and ensure there are no conflicts between the users. These considerations reinforced the necessity of user-centric design and robust architecture for practical software solutions.

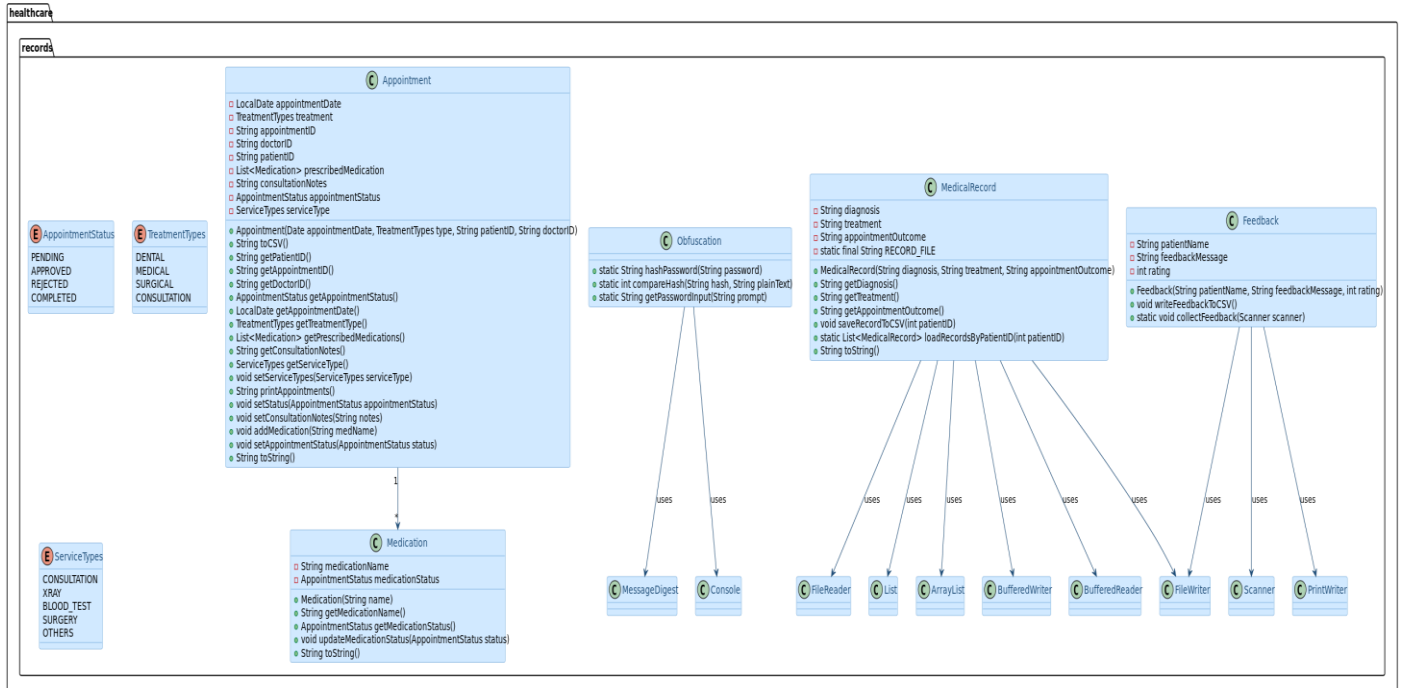
2. DETAILED UML CLASS DIAGRAM



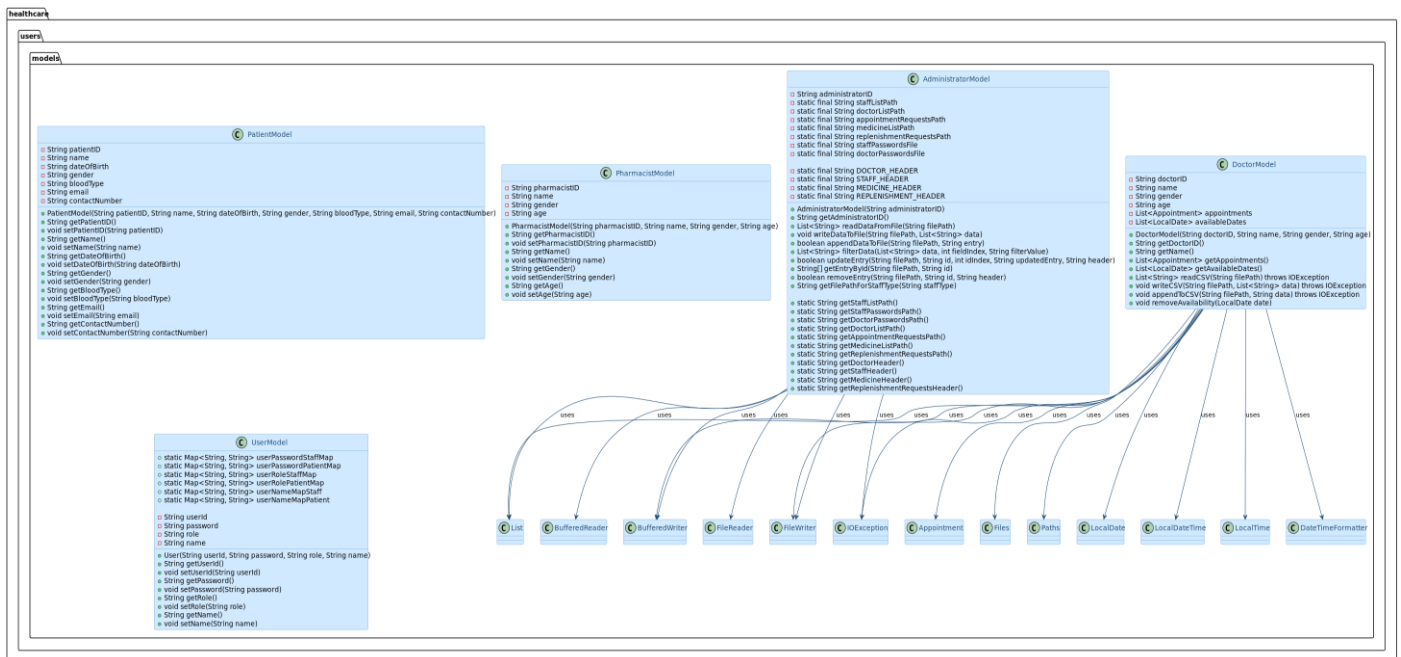
2.1 MAIN DIAGRAM



2.2 RECORDS SUB-DIAGRAM



2.3 MODEL SUB-DIAGRAM



3.2. Pharmacist Test Cases : View Medication Inventory

```
=====
                        Login Screen
=====
Enter Hospital ID: P002
Enter Password: █
```

```
=====
                        Pharmacist Menu
=====
| 1. View Appointment Outcome Record |
| 2. Update Prescription Status      |
| 3. View Medication Inventory       |
| 4. Submit Replenishment Request    |
| 5. Logout                          |
=====
Enter your choice: 3█
```

```
=====
                        Medication Inventory
=====
Medication Inventory:
-----
Medicine Name      Quantity      Stock Level
-----
Paracetamol        20           Low Stock
Amoxicillin        60           In Stock
Crocin             17           Low Stock
Cetirizine         50           In Stock
Panadol Cold       38           In Stock
Gabapantin         42           In Stock
Cyclopam           30           In Stock
Tiger Balm         15           Low Stock
Band-Aid           200          In Stock
-----

Press Enter to continue...
█
```

3.3 Doctor Test Cases : Set Appointment Availability

```
=====
                        Doctor Menu
=====
| 1. View Patient Medical Records    |
| 2. Update Patient Medical Records  |
| 3. View Personal Schedule          |
| 4. Set Availability for Appointments|
| 5. Accept or Decline Appointment Req. |
| 6. View Upcoming Appointments      |
| 7. Record Appointment Outcome      |
| 8. Logout                          |
=====
Enter your choice: 4█
```

```
Mark slots as unavailable by entering the slot number. Enter 'done' when finished.
1. 09:00
2. 10:00
3. 11:00
4. 12:00
5. 14:00
6. 15:00
7. 16:00
Enter slot number to mark as unavailable, or 'done' to finish: █
```

```
Mark slots as unavailable by entering the slot number. Enter 'done' when finished.
1. 09:00
2. 10:00
3. 11:00
4. 12:00
5. 14:00
6. 15:00
7. 16:00
Enter slot number to mark as unavailable, or 'done' to finish: 1
Marked slot as unavailable: 09:00
Enter slot number to mark as unavailable, or 'done' to finish: 2
Marked slot as unavailable: 10:00
Enter slot number to mark as unavailable, or 'done' to finish: 3
Marked slot as unavailable: 11:00
Enter slot number to mark as unavailable, or 'done' to finish: 4
Marked slot as unavailable: 12:00
Enter slot number to mark as unavailable, or 'done' to finish: done█
```


3.4. Admin Test Cases: View Staff & Approve Replenishment Requests

```
=====
                        Staff List
=====
|
| 1. View All
| 2. Filter by Role, Gender, or Age
|
=====
Choose an option: 1
```

```
=====
                        Doctor List
=====
| Doctor ID | Name       | Gender | Age | Specialisation |
|-----|-----|-----|-----|-----|
| D001 | Ross      | Male   | 45  | Pediatrician   |
| D002 | Rachel    | Female | 38  | Gynaecologist  |
| D003 | Chandler  | Male   | 25  | Cardiology     |
| D004 | Joey      | Male   | 45  | Pediatrician   |
| D005 | Monica    | Female | 38  | Gynaecologist  |
| D006 | Phoebe    | Female | 25  | neurology      |
| D007 | Will      | Female | 37  | orthopaedic    |
| D008 | Jacky     | Male   | 30  | Orthopaedic    |
| D009 | Minnie    | Female | 40  | Vet            |
|-----|-----|-----|-----|-----|

Press Enter to continue...
```

```
=====
                        Login Screen
=====
Enter Hospital ID: A002
Enter Password:
```

```
=====
                        Administrator Menu
=====
|
| 1. View and Manage Hospital Staff
| 2. View Appointment Details
| 3. View and Manage Medication Inventory
| 4. Approve Replenishment Requests
| 5. View Patient Feedback
| 6. Logout
|
=====
Choose an option: 1
```

```
Good Day Michael!
Do you want to change your password? (yes/no): no
```

```
=====
                        Staff List
=====
| Staff ID | Name       | Role       | Gender | Age |
|-----|-----|-----|-----|-----|
| A001 | Harvey    | Administrator | Male   | 40  |
| A002 | Michael   | Administrator | Male   | 19  |
| A003 | James     | Administrator | Male   | 29  |
| A004 | Charles   | Administrator | Male   | 39  |
| A005 | Becky     | Administrator | Female | 45  |
| A006 | Jack      | Administrator | Male   | 23  |
| A007 | Vishesh   | Administrator | Male   | 54  |
| P001 | Arun      | Pharmacist   | Male   | 21  |
| P002 | Roland    | Pharmacist   | Male   | 33  |
| P004 | Alexis    | Pharmacist   | Female | 23  |
| P005 | Sheldon   | Pharmacist   | Male   | 43  |
| P006 | Howard    | Pharmacist   | Male   | 56  |
| P007 | Raj       | Pharmacist   | Male   | 23  |
| N001 | Alex      | Nurse        | Male   | 43  |
| N002 | Claire    | Nurse        | Male   | 56  |
| N003 | Cam       | Nurse        | Male   | 23  |
| N004 | Mitch     | Nurse        | Male   | 25  |
| N005 | Mitch     | Nurse        | male   | 40  |
| N006 | David     | Nurse        | Male   | 27  |
|-----|-----|-----|-----|-----|

Press Enter to continue...
```

```
=====
                        Administrator Menu
=====
|
| 1. View and Manage Hospital Staff
| 2. View Appointment Details
| 3. View and Manage Medication Inventory
| 4. Approve Replenishment Requests
| 5. View Patient Feedback
| 6. Logout
|
=====
Choose an option: 4
```

```
=====
                        Replenishment Requests Management
=====
|
| 1. View Replenishment Requests
| 2. Restock Medicine
| 3. Exit
|
=====
Choose an option: 1
```

```
=====
                        Replenishment Requests
=====
| Medicine Name | Stock | Date       | Replenishment Request |
|-----|-----|-----|-----|
| Ibuprofen     | 8     | 2024-11-09 | APPROVED               |
| Amoxicillin   | 60    | 2024-11-10 | APPROVED               |
| Paracetamol   | 24    | 2024-11-01 | REQUESTED              |
| Crocin        | 17    | 2024-11-16 | REQUESTED              |
| Band-Aid      | 200   | 2024-11-16 | APPROVED               |
|-----|-----|-----|-----|

Press Enter to continue...
```