NANYANG TECHNOLOGICAL UNIVERSITY

Declaration of Original Work for CE/CZ2002 Assignment

We hereby declare that the attached group assignment has been researched, undertaken, completed, and submitted as a collective effort by the group members listed below.

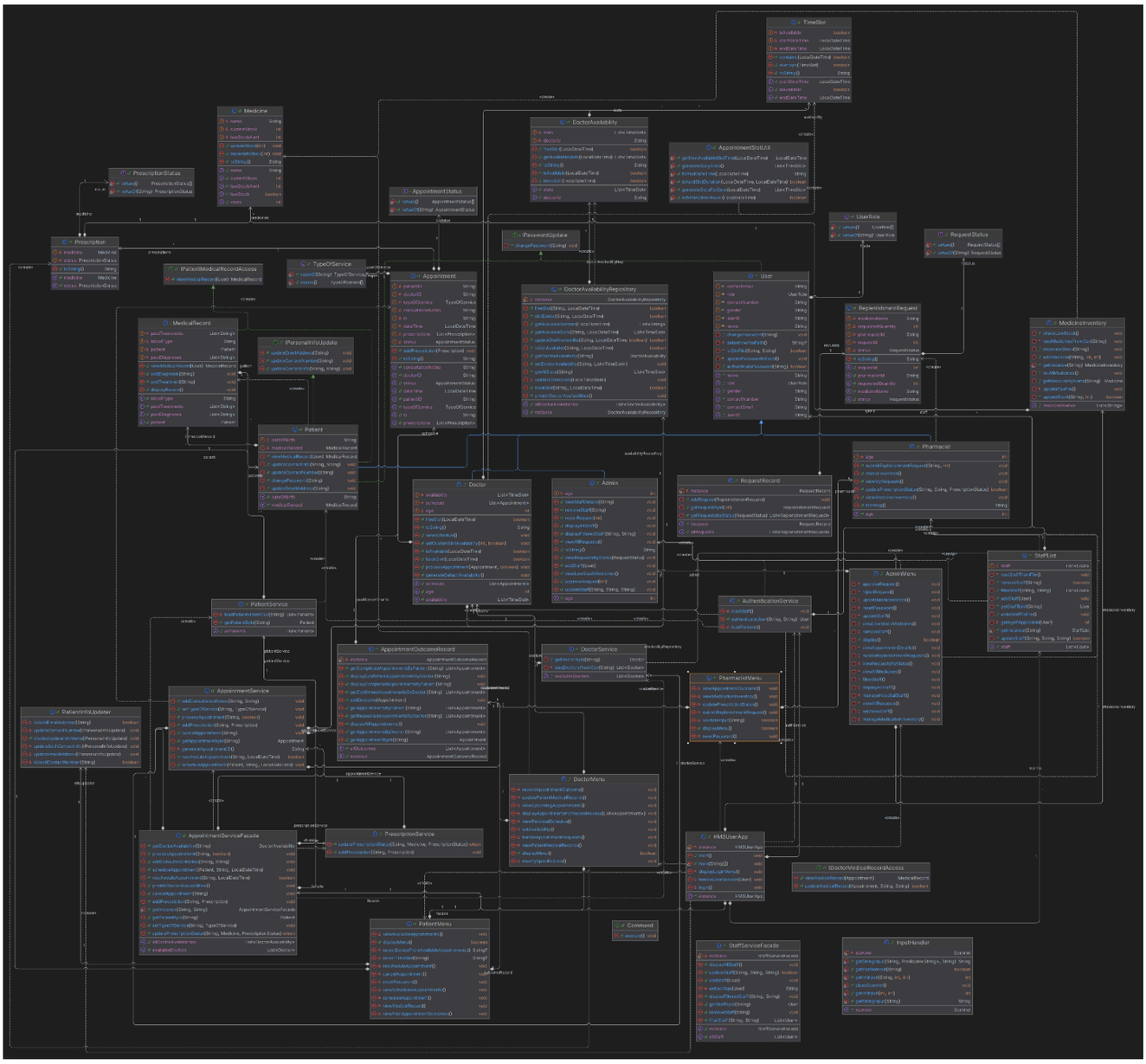
We have honored the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work.

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

Name	Course (CE2002 or CZ2002)	Lab Group	Signature /Date
Kamarudeen Hana Fathima	SC2002	SCEC	21/11/2024
Revathi Selvasevaran	SC2002	SCEC	21/11/2024
Dosi Veer	SC2002	SCEC	21/11/2024

GitHub Link: https://github.com/veerdosi/HMS

UML Class Diagram



Entity Classes (Domain Objects):

1. User
2. Patient
3. Doctor
4. Admin
5. Pharmacist
6. MedicalRecord
7. Medicine
8. Appointment
9. Prescription
10. TimeSlot
11. ReplenishmentRequest
12. DoctorAvailability

Boundary Classes (UI/Interface):

1. AdminMenu
2. DoctorMenu
3. PatientMenu
4. PharmacistMenu
5. InputHandler

Control Classes (Business Logic/Service):

1. AuthenticationService
2. AppointmentService
3. AppointmentServiceFacade
4. DoctorService
5. PatientService
6. PrescriptionService
7. StaffServiceFacade
8. MedicineInventory
9. DoctorAvailabilityRepository
10. AppointmentOutcomeRecord
11. RequestRecord
12. StaffList
13. PatientInfoUpdater

Additional Types:

- Enums:
    1. UserRole
    2. AppointmentStatus
    3. PrescriptionStatus
    4. RequestStatus
    5. TypeOfService
- Interfaces:
    1. IPasswordUpdate
    2. IPersonalInfoUpdate
    3. IPatientMedicalRecordAccess
    4. IDoctorMedicalRecordAccess
    5. Command

Test Cases

Module	#	Test Case	Description and Screenshots	

General Login	25	First Time Login & Password Change	User logs in with the default password and changes it.
			`Please select an option:`
			`1. Log in`
			`2. Exit the program`
			`Enter your choice: 1`
			`Enter User ID: P1001`
			`Enter Password: password`
			
			`First Login: Please reset your password!`
			
			`New Password: pass123`
			`Password updated successfully in the file.`
			`Invalid credentials. Please try again.`
				Verify that the password change is successful, and the user can log in with the new password.
				`New Password: pass123`
				`Password updated successfully in the file.`
				`Invalid credentials. Please try again.`
				
				`Please select an option:`
				`1. Log in`
				`2. Exit the program`
				`Enter your choice: 1`
				`Enter User ID: P1001`
				`Enter Password: password`
				`Invalid credentials. Please try again.`
				
				`Please select an option:`
				`1. Log in`
				`2. Exit the program`
				`Enter your choice: 1`
				`Enter User ID: P1001`
				`Enter Password: pass123`
				
				`Login successful! Welcome, Alice Brown`
				
				`---- Patient Menu ----`

	26	Login with Incorrect Credentials	User attempts to log in with an incorrect password
			`Please select an option:`
			`1. Log in`
			`2. Exit the program`
			`Enter your choice: 1`
			`Enter User ID: P1001`
			`Enter Password: password`
			`Invalid credentials. Please try again.`
			
			`Please select an option:`
			`1. Log in`
			`2. Exit the program`
				Verify that the system displays an error message indicating invalid credentials, and login is denied.
				[See previous]

Patient	1	View Medical Record	Patients view their own medical record.
			Verify that the system displays the patient's medical record, including Patient ID, Name, Date of Birth, Gender, Contact Information, Blood Type, and Past Diagnoses and Treatments.
				`Enter your choice: 1`
				`Fetching Medical Record...`
				`------ Medical Record ------`
				`Name: Alice Brown`
				`Patient ID: P1001`
				`Date of Birth: 1980-05-14`
				`Gender: Female`
				`Contact Info: 1111 1111 | alice.brown@example.com`
				`Blood Type: A+`
				`Past Diagnoses: []`
				`Past Treatments: []`

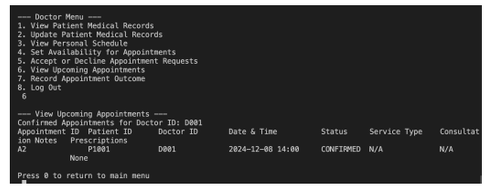
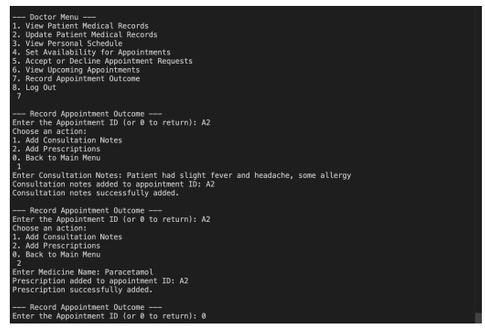
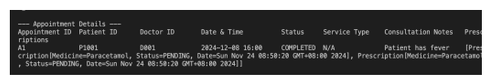
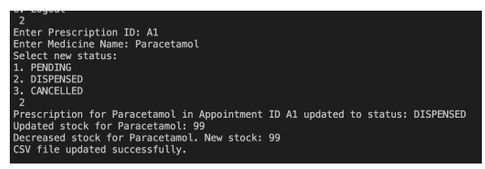
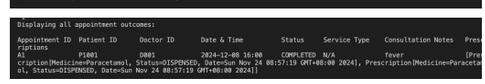
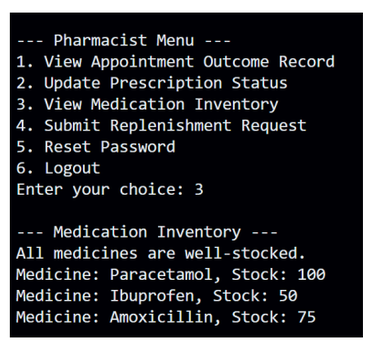
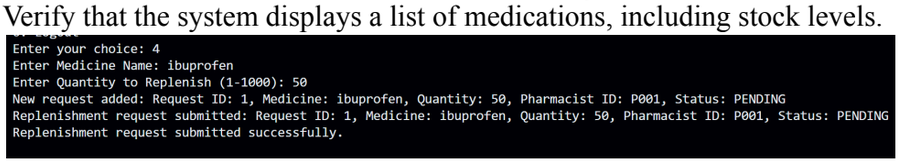
	2	Update Personal Information	Patient updates their email address and contact number.
			Verify that patient's contact information is updated successfully, and the changes are reflected in the medical record.
				`Enter your choice: 2`
				
				`---- Update Personal Information ----`
				`1. Update Contact Number`
				`2. Update Email Address`
				`3. Update Both`
				`4. Back to Main Menu`
				`Enter your choice: 3`
				`Enter new contact number (8 digits): 12345678`
				`Enter new email address: fw@egwg.com`
				`Contact information updated successfully.`
				
				`Enter your choice: 1`
				`Fetching Medical Record...`
				`------ Medical Record ------`
				`Name: Alice Brown`
				`Patient ID: P1001`
				`Date of Birth: 1980-05-14`
				`Gender: Female`
				`Contact Info: 12345678 | fw@egwg.com`
				`Blood Type: A+`
				`Past Diagnoses: []`
				`Past Treatments: []`

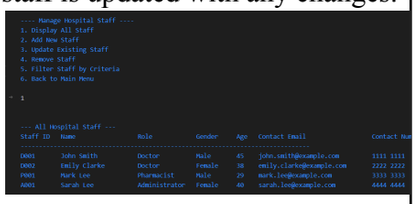
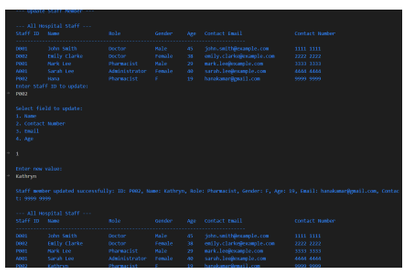
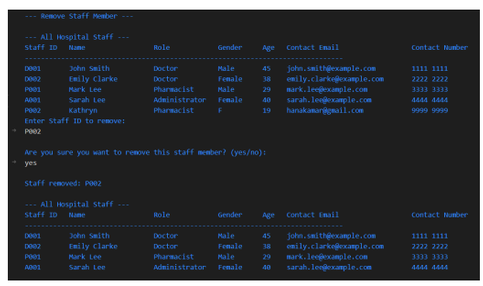
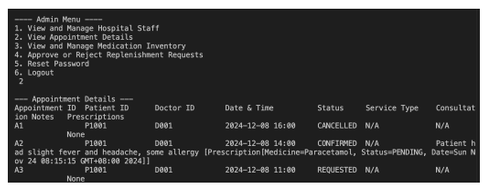
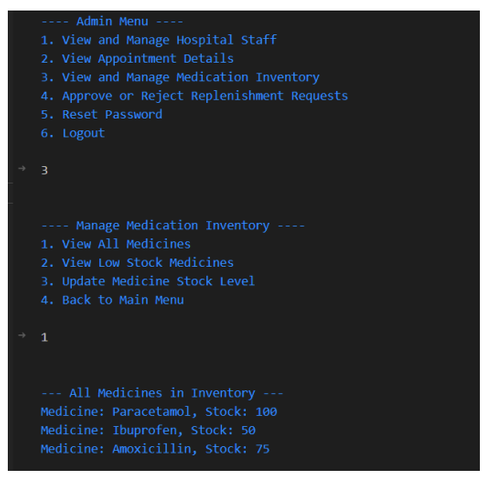
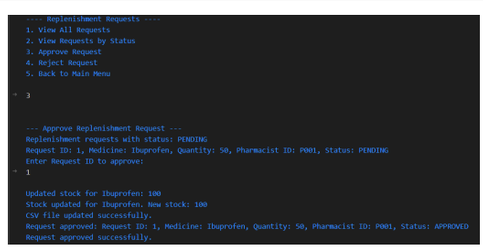
	3	View Available Appointment Slots	Patient views available appointment slots with doctors.
			Verify that the system displays a list of available appointment slots, showing doctors' ID, dates, and times.
			(System design to show DocID rather than name can be modified. Currently can only book for next day.)
				`--- Schedule New Appointment ---`
				
				`--- Available Appointments ---`
				`+---+---+---+`
				`| Doctor ID| Start Time | End Time |`
				`+---+---+---+`
				`| D001 | 25-11-2024 09:00 | 25-11-2024 10:00 |`
				`| D001 | 25-11-2024 10:00 | 25-11-2024 11:00 |`
				`| D001 | 25-11-2024 11:00 | 25-11-2024 12:00 |`
				`| D001 | 25-11-2024 14:00 | 25-11-2024 15:00 |`
				`| D001 | 25-11-2024 15:00 | 25-11-2024 16:00 |`
				`| D001 | 25-11-2024 16:00 | 25-11-2024 17:00 |`
				`| D001 | 26-11-2024 09:00 | 26-11-2024 10:00 |`
				`| D001 | 26-11-2024 10:00 | 26-11-2024 11:00 |`
				`| D001 | 26-11-2024 11:00 | 26-11-2024 12:00 |`

	4	Schedule	Patient schedules a new	

		Appointment	appointment with a doctor. Verify that the appointment is scheduled successfully with status "confirmed". The selected time slot becomes unavailable to other patients. The system should prevent the patient from booking a time slot that is unavailable/already booked.	Select a doctor by entering the corresponding number: 1. Doctor ID: D001 1 Available Time Slots: +----+----------+----------+ | No. | Start Time | End Time | +----+----------+----------+ | 1 | 25-11-2024 09:00 | 25-11-2024 10:00 | | 2 | 25-11-2024 10:00 | 25-11-2024 11:00 | | 3 | 25-11-2024 11:00 | 25-11-2024 12:00 | | 78 | 07-12-2024 16:00 | 07-12-2024 17:00 | | 79 | 08-12-2024 09:00 | 08-12-2024 10:00 | | 80 | 08-12-2024 10:00 | 08-12-2024 11:00 | | 81 | 08-12-2024 11:00 | 08-12-2024 12:00 | | 82 | 08-12-2024 14:00 | 08-12-2024 15:00 | | 83 | 08-12-2024 15:00 | 08-12-2024 16:00 | | 84 | 08-12-2024 16:00 | 08-12-2024 17:00 | +----+----------+----------+ 84 Appointment scheduled successfully! ---------- Appointment ID: A1 Doctor: John Smith Date/Time: 08-12-2024 16:00
	5	Reschedule Appointment	Patient reschedules an existing appointment to a new slot. Verify that the appointment is rescheduled successfully. The previous time slot becomes available, and the new slot is reserved.	| 76 | 07-12-2024 14:00 | 07-12-2024 15:00 | | 77 | 07-12-2024 15:00 | 07-12-2024 16:00 | | 78 | 07-12-2024 16:00 | 07-12-2024 17:00 | | 79 | 08-12-2024 09:00 | 08-12-2024 10:00 | | 80 | 08-12-2024 10:00 | 08-12-2024 11:00 | | 81 | 08-12-2024 11:00 | 08-12-2024 12:00 | | 82 | 08-12-2024 15:00 | 08-12-2024 16:00 | | 83 | 08-12-2024 16:00 | 08-12-2024 17:00 | +----+----------+----------+ 81 Appointment rescheduled successfully to 08-12-2024 11:00 Appointment successfully rescheduled. New appointment details: Doctor ID: D001 Date/Time: 08-12-2024 11:00
	6	Cancel Appointment	Patient cancels an existing appointment. Verify that the appointment is canceled successfully, and the time slot becomes available for others.	6 --- Cancel Appointment --- Select an appointment to cancel: 1. Appointment ID: A1, Doctor ID: D001, Date: 08-12-2024 16:00 1 Appointment with ID A1 has been canceled. Appointment canceled successfully. | D001 | 08-12-2024 10:00 | 08-12-2024 11:00 | | D001 | 08-12-2024 11:00 | 08-12-2024 12:00 | | D001 | 08-12-2024 14:00 | 08-12-2024 15:00 | | D001 | 08-12-2024 15:00 | 08-12-2024 16:00 | | D001 | 08-12-2024 16:00 | 08-12-2024 17:00 | +----------+----------+----------+
	7	View Scheduled Appointments	Patient views their list of scheduled appointments. Verify that the system displays all upcoming appointments with details like doctor name, date, time, and status.	7 --- Scheduled Appointments --- +----------+----------+----------+----------+ | Appointment ID| Doctor ID| Status | Date & Time | +----------+----------+----------+----------+ | A1 | D001 | CANCELLED | 08-12-2024 16:00 | | A2 | D001 | REQUESTED | 08-12-2024 14:00 | +----------+----------+----------+----------+
	8	View Past Appointment Outcome Records	Patient views outcome records of past appointments. Verify that the system displays past appointment details, including services provided, prescribed medications, and consultation notes.	8 --- Past Appointment Outcomes --- +----------+----------+----------+----------+----------+ | Appointment ID| Doctor ID| Status | Date & Time | Notes | +----------+----------+----------+----------+----------+ | A1 | D001 | COMPLETED | 08-12-2024 16:00 | Patient has fever | +----------+----------+----------+----------+----------+
Doctor	9	View Patient Medical Records	Doctor views medical records of patients under their care. Verify that the patient's medical record is displayed, including all relevant medical history.	1 --- View Patient Medical Records --- Select an appointment: +----+----------+----------+----------+ | No. | Appointment ID| Patient ID | Date & Time | +----+----------+----------+----------+ | 1 | A2 | P1001 | 08-12-2024 14:00 | +----+----------+----------+----------+ 1 ------- Medical Record ------- Name: Alice Brown Patient ID: P1001 Date of Birth: 1980-05-14 Gender: Female Contact Info: 1111 1111 | alice.brown@example.com Blood Type: A+ Past Diagnoses: [] Past Treatments: [] ------- Medical Record ------- Name: Alice Brown Patient ID: P1001 Date of Birth: 1980-05-14 Gender: Female Contact Info: 1111 1111 | alice.brown@example.com Blood Type: A+ Past Diagnoses: [] Past Treatments: []

	10	Update Patient Medical Records	Doctors add a new diagnosis and treatment plan to a patient's medical record. Verify that the medical record is updated successfully, reflecting the new information.	2 --- Update Patient Medical Records --- Select an appointment: +-----+----------+----------+----------+ | No. | Appointment ID| Patient ID | Date & Time | +-----+----------+----------+----------+ | 1 | A2 | P1001 | 08-12-2024 14:00 | +-----+----------+----------+----------+ 1 ------ Medical Record ------ Name: Alice Brown Patient ID: P1001 Date of Birth: 1980-05-14 Gender: Female Contact Info: 1111 1111 | alice.brown@example.com Blood Type: A+ Past Diagnoses: [] Past Treatments: [] 1. Add Diagnosis 2. Add Treatment 1 Enter Diagnosis: Allergic to seafood Diagnosis added successfully. 2 --- Update Patient Medical Records --- Select an appointment: +-----+----------+----------+----------+ | No. | Appointment ID| Patient ID | Date & Time | +-----+----------+----------+----------+ | 1 | A2 | P1001 | 08-12-2024 14:00 | +-----+----------+----------+----------+ 1 ------ Medical Record ------ Name: Alice Brown Patient ID: P1001 Date of Birth: 1980-05-14 Gender: Female Contact Info: 1111 1111 | alice.brown@example.com Blood Type: A+ Past Diagnoses: [Allergic to seafood] Past Treatments: [] 1. Add Diagnosis 2. Add Treatment 2 Enter Treatment: Allegra would be good Treatment added successfully. --- View Patient Medical Records --- Select an appointment: +-----+----------+----------+----------+ | No. | Appointment ID| Patient ID | Date & Time | +-----+----------+----------+----------+ | 1 | A2 | P1001 | 08-12-2024 14:00 | +-----+----------+----------+----------+ 1 ------ Medical Record ------ Name: Alice Brown Patient ID: P1001 Date of Birth: 1980-05-14 Gender: Female Contact Info: 1111 1111 | alice.brown@example.com Blood Type: A+ Past Diagnoses: [Allergic to seafood, Sleep deprived] Past Treatments: [Allegra would be good]
	11	View Personal Schedule	Doctor views their personal appointment schedule. Verify that the system displays the doctor's upcoming appointments and availability slots.	Enter your choice: 1 --- Schedule for Doctor: John Smith --- No scheduled appointments.
	12	Set Availability for Appointments	Doctors set their availability for patient appointments. Verify that the doctor's availability is updated, and patients can see the new slots when scheduling appointments.	--- 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 Requests 6. View Upcoming Appointments 7. Record Appointment Outcome 8. Log Out 4 --- Set Availability --- 1. Generate Default Weekly Schedule 2. Modify Specific Slots 0. Back 1 Generated 84 slots for the next 14 days. Default availability generated for Doctor: John Smith Default weekly schedule generated. --- Available Slots --- +----------+----------+----------+ | Start Date/Time | End Date/Time | Status | +----------+----------+----------+ | 25-11-2024 09:00 | 25-11-2024 10:00 | Available | | 25-11-2024 10:00 | 25-11-2024 11:00 | Available | | 25-11-2024 11:00 | 25-11-2024 12:00 | Available | | 25-11-2024 14:00 | 25-11-2024 15:00 | Available | | 25-11-2024 15:00 | 25-11-2024 16:00 | Available | | 25-11-2024 16:00 | 25-11-2024 17:00 | Available |
	13	Accept or Decline Appointment Requests	Doctor accepts or declines an appointment request from a patient. Verify that the appointment status changes to "confirmed" when accepted or “cancelled” when declined , and the patient is able to see the updated status of the appointment.	Enter Appointment ID to process: A1 Invalid Appointment ID or appointment not found. --- Accept or Decline Appointment Requests --- Pending Appointments: +----------+----------+----------+ | Appointment ID| Patient ID | Date & Time | +----------+----------+----------+ | A2 | P1001 | 08-12-2024 14:00 | +----------+----------+----------+ Choose an action: 1. Process Appointment Request 0. Back to Main Menu 1 Enter Appointment ID to process: A2 1. Accept 2. Decline 1 Appointment accepted successfully. Appointment confirmed successfully. Appointment accepted successfully. --- Accept or Decline Appointment Requests --- No pending appointment requests.

				
	14	View Upcoming Appointments	Doctor views all upcoming confirmed appointments. Verify that the system displays a list of all upcoming appointments with patient details and appointment times.	
	15	Record Appointment Outcome	Doctor records the outcome of a completed appointment. Verify that the appointment outcome is recorded, and relevant updates are visible to the patient under “View Past Appointment Outcome Records”.	
Pharmacist	16	View Appointment Outcome Record	Pharmacists view appointment outcome records to process prescriptions. Verify that the system displays the appointment outcome details, including prescribed medications.	
	17	Update Prescription Status	Pharmacist updates the status of a prescription to "dispensed." Verify that the prescription status is updated, and the change is reflected in the patient's records.	 
	18	View Medication Inventory	Pharmacists view the current medication inventory. Verify that the system displays a list of medications, including stock levels.	
	19	Submit Replenishment Request	Pharmacists view the current medication inventory. Verify that the system displays a list of medications, including stock levels. 	

Admin	20	View and Manage Hospital Staff	Administrators can view the list of hospital staff and add, update or remove staff members. Verify that the displayed list of staff is updated with any changes.  ---- Manage Hospital Staff ---- 1. Display All Staff 2. Add New Staff 3. Update Existing Staff 4. Remove Staff 5. Filter Staff by Criteria 6. Back to Main Menu → 1 --- All Hospital Staff --- Staff ID Name Role Gender Age Contact Email Contact Num D001 John Smith Doctor Male 45 john.smith@example.com 1111 1111 D002 Emily Clarke Doctor Female 38 emily.clarke@example.com 2222 2222 P001 Mark Lee Pharmacist Male 29 mark.lee@example.com 3333 3333 A001 Sarah Lee Administrator Female 40 sarah.lee@example.com 4444 4444	 --- Add New Staff Member --- Enter Staff ID: → P002 Enter Name: → Hana Select Role: 1. Doctor 2. Pharmacist 3. Administrator → 2 Enter Gender (M/F): → F Enter age: → 19 Enter Contact Email: → hanakamar@gmail.com Enter Contact Number: → 9999 9999  Staff added: ID: P002, Name: Hana, Role: Pharmacist, Gender: F, Age: 19, Email: hanakamar@gmail.com, Contact: 9999 9999 --- All Hospital Staff --- Staff ID Name Role Gender Age Contact Email Contact Number D001 John Smith Doctor Male 45 john.smith@example.com 1111 1111 D002 Emily Clarke Doctor Female 38 emily.clarke@example.com 2222 2222 P001 Mark Lee Pharmacist Male 29 mark.lee@example.com 3333 3333 A001 Sarah Lee Administrator Female 40 sarah.lee@example.com 4444 4444 P002 Hana Pharmacist F 19 hanakamar@gmail.com 9999 9999
			 --- Update Staff Member --- --- All Hospital Staff --- Staff ID Name Role Gender Age Contact Email Contact Number D001 John Smith Doctor Male 45 john.smith@example.com 1111 1111 D002 Emily Clarke Doctor Female 38 emily.clarke@example.com 2222 2222 P001 Mark Lee Pharmacist Male 29 mark.lee@example.com 3333 3333 A001 Sarah Lee Administrator Female 40 sarah.lee@example.com 4444 4444 P002 Hana Pharmacist F 19 hanakamar@gmail.com 9999 9999 Enter Staff ID to update: → P002 Select field to update: 1. Name 2. Contact Number 3. Email 4. Age → 1 Enter new value: → Kathryn Staff member updated successfully: ID: P002, Name: Kathryn, Role: Pharmacist, Gender: F, Age: 19, Email: hanakamar@gmail.com, Contact: 9999 9999 --- All Hospital Staff --- Staff ID Name Role Gender Age Contact Email Contact Number D001 John Smith Doctor Male 45 john.smith@example.com 1111 1111 D002 Emily Clarke Doctor Female 38 emily.clarke@example.com 2222 2222 P001 Mark Lee Pharmacist Male 29 mark.lee@example.com 3333 3333 A001 Sarah Lee Administrator Female 40 sarah.lee@example.com 4444 4444 P002 Kathryn Pharmacist F 19 hanakamar@gmail.com 9999 9999	 --- Remove Staff Member --- --- All Hospital Staff --- Staff ID Name Role Gender Age Contact Email Contact Number D001 John Smith Doctor Male 45 john.smith@example.com 1111 1111 D002 Emily Clarke Doctor Female 38 emily.clarke@example.com 2222 2222 P001 Mark Lee Pharmacist Male 29 mark.lee@example.com 3333 3333 A001 Sarah Lee Administrator Female 40 sarah.lee@example.com 4444 4444 P002 Kathryn Pharmacist F 19 hanakamar@gmail.com 9999 9999 Enter Staff ID to remove: → P002 Are you sure you want to remove this staff member? (yes/no): → yes Staff removed: P002 --- All Hospital Staff --- Staff ID Name Role Gender Age Contact Email Contact Number D001 John Smith Doctor Male 45 john.smith@example.com 1111 1111 D002 Emily Clarke Doctor Female 38 emily.clarke@example.com 2222 2222 P001 Mark Lee Pharmacist Male 29 mark.lee@example.com 3333 3333 A001 Sarah Lee Administrator Female 40 sarah.lee@example.com 4444 4444
	21	View Appointments Details	Administrator views all appointments. Verify that the system displays a list of appointments including details like Patient ID, Doctor ID, status, and date/time.	 === Admin Menu === 1. View and Manage Hospital Staff 2. View Appointment Details 3. View and Manage Medication Inventory 4. Approve or Reject Replenishment Requests 5. Reset Password 6. Logout → 2 === Appointment Details === Appointment ID Patient ID Doctor ID Date & Time Status Service Type Consultation Notes Prescriptions A1 P1001 D001 2024-12-08 16:00 CANCELLED N/A N/A None A2 P1001 D001 2024-12-08 14:00 CONFIRMED N/A Patient had slight fever and headache, some allergy [Prescription(Medicine=Paracetamol, Status=PENDING, Date=Sun Nov 24 00:15:15 GMT+08:00 2024)] A3 P1001 D001 2024-12-08 11:00 REQUESTED N/A N/A None
	22	View and Manage Medication Inventory	Administrator updates the stock level of a medication. Verify that the medication's stock level is updated in the inventory.	 ---- Admin Menu ---- 1. View and Manage Hospital Staff 2. View Appointment Details 3. View and Manage Medication Inventory 4. Approve or Reject Replenishment Requests 5. Reset Password 6. Logout → 3 ---- Manage Medication Inventory ---- 1. View All Medicines 2. View Low Stock Medicines 3. Update Medicine Stock Level 4. Back to Main Menu → 1 --- All Medicines in Inventory --- Medicine: Paracetamol, Stock: 100 Medicine: Ibuprofen, Stock: 50 Medicine: Amoxicillin, Stock: 75
	23	Approve Replenishment Requests	Administrator approves a replenishment request from a pharmacist. Verify that the request status changes to "approved," and the medication inventory is updated accordingly.	 --- Replenishment Requests --- 1. View All Requests 2. View Requests by Status 3. Approve Request 4. Reject Request 5. Back to Main Menu → 3 --- Approve Replenishment Request --- Replenishment requests with status: PENDING Request ID: 1, Medicine: Ibuprofen, Quantity: 50, Pharmacist ID: P001, Status: PENDING Enter Request ID to approve: → 1 Updated stock for Ibuprofen: 100 Stock updated for Ibuprofen. New stock: 100 CSV file updated successfully. Request approved: Request ID: 1, Medicine: Ibuprofen, Quantity: 50, Pharmacist ID: P001, Status: APPROVED Request approved successfully.

Design Considerations

When designing our hospital management software, we aimed for modular design that was open to modifications and can be maintained and is reusable. Ideally we tried to optimize the implementation to have loose coupling by managing the dependencies and to have high cohesion by grouping classes and methods by the singular purpose they are meant to serve. Reducing the cross-dependencies and grouping the classes well makes it easier for us to maintain the program since it is easier to implement any changes in a few places rather than having to hunt around for all the pieces that need to change together. It also makes the program more flexible and extensible since we would not have to change too much around pre-existing program files to implement a new feature or remove any obsolete legacy features.

Unfortunately in practice, we had to put strange code repetitions in places due to page structure in CLI. You have Login menu, which opens straight to the different PatientMenu, DoctorMenu, PharmacistMenu, AdminMenu depending on the user but it makes no sense resetting password on the main menu before logging in, so since we missed out on a middle layer of abstraction, 4 different implementations of reset password happened.

Assumptions

- HMS system scale is that of a General Practitioner Clinic setting rather than a large hospital. This helped us scope the project without having to group staff into various departments as there would be in a hospital.
- The operating hours of the clinic are limited to be from 0900h to 1700h but the clinic is open 7 days a week so we do not have to also check if the date of Appointment is a weekday.
- Each appointment slot is 1 hr long, booking unavailable from 12-2PM as the clinic is closed for lunch.
- Patients can only view available appointments for 1 day ahead (like polyclinics), and cannot book for today if the time is past the current system time.
- Patients can only book an appointment in the upcoming 2 weeks

Design Patterns

Relying on design patterns allowed us to utilize well-tested templates to address common software development problems, ensuring loose coupling and high cohesion. These patterns played a significant role in creating a robust and maintainable codebase.

Facade Design Pattern

The **Facade** pattern was implemented using classes such as AppointmentServiceFacade and StaffServiceFacade. This approach established a clean and organized method for accessing implementations of various operations, adhering to the Dependency Inversion Principle. By centralizing functionality under facade classes, developers could access related methods through a single interface rather than tracking implementations across multiple classes. The suffix “-Facade” signified their purpose and streamlined system interactions.

Advantages:

- Simplified interactions with complex subsystems
- Unified interface for related operations
- Reduced coupling between components

Singleton Design Pattern

The **Singleton** pattern was implemented in critical classes like AppointmentOutcomeRecord, MedicineInventory, RequestRecord, StaffList, and HMSUserApp. Given the design choice to store data in lists of objects (e.g., appointments, medicines, prescriptions), the Singleton pattern ensured a single source of truth, preventing data duplication and unnecessary errors caused by multiple instantiations.

Advantages:

- Single point of truth for critical data

- Prevention of multiple instances of resource-heavy objects
- Centralized access to shared resources

Command Design Pattern
A basic implementation of the **Command** pattern was achieved through a Command interface. This allowed for extensible operation handling and set a foundation for more sophisticated command processing in the future.
Advantages:

- Simplifies operation handling
- Enables easy extensibility for additional commands

Alternative Patterns Considered

1. **Observer Pattern:**
    - Could have been used for real-time updates, such as:
        - Notifying staff of appointment changes
        - Updating inventory when prescriptions are dispensed
        - Alerting administrators of low stock
    - **Trade-off:** While it provided real-time notifications, the added complexity and time constraints led to its exclusion.

SOLID Design Principles

- **Single Responsibility Principle (SRP)**
    - Each class has a single, clearly defined responsibility according to the Single Responsibility Principle (SRP). This design facilitates system expansion and maintenance. For instance, the PatientMenu manages appointments and other patient-specific chores, whereas the DoctorMenu concentrates on doctor-specific duties like medical record management. Only logic pertaining to appointments, such as scheduling, rescheduling, or canceling appointments, is handled by the AppointmentService. Comparably, the MedicalRecord contains the logic for handling patient medical histories, whereas the AppointmentOutcomeRecord handles and stores previous appointment information. By keeping these duties distinct, each class fulfills a distinct role, which lowers the possibility of defects being introduced during functionality updates and streamlines the maintenance procedure.
- **Open/Closed Principle (OCP)**
    - The Open/Closed Principle (OCP) states that classes should be open for extension but closed for modification. This idea makes it possible to introduce new features without changing the code that already exists. For example, by permitting new implementations, the IPatientMedicalRecordAccess interface enables new methods of accessing medical information while maintaining the integrity of the current codebase. By offering a consistent interface for appointment management, the AppointmentServiceFacade enables the integration of new services or functionalities without altering its structure. Similarly, rather than changing pre-existing menu classes like PatientMenu or DoctorMenu, new roles like AdminMenu can be created by developing new ones. By facilitating the smooth incorporation of new features, this modular design keeps the system adaptable and expandable.
- **Liskov Substitution Principle (LSP)**
    - The Liskov Substitution Principle (LSP) ensures that derived classes can be substituted for their base types without affecting the correctness of the program. For instance, Patient and Doctor are extensions of the User class that can be used interchangeably in situations where a User type is required. Any implementing class, including MedicalRecord, is guaranteed to adhere to the expected behavior by the IPatientMedicalRecordAccess interface. In a similar vein, entities such as Patient guarantee consistent handling of updates to personal information through the implementation of the IPersonalInfoUpdate interface. Following LSP promotes both flexibility and accuracy by guaranteeing that the system stays stable and that component substitutions do not impair functioning.
- **Interface Segregation Principle (ISP)**

- The Interface Segregation Principle (ISP) focuses on creating small, specific interfaces, avoiding the need for classes to implement unnecessary methods. To ensure that classes like DoctorMenu or MedicalRecord are not overloaded with unnecessary features, IPatientMedicalRecordAccess, for instance, is made only for accessing medical records. While the IPasswordUpdate interface only handles password updates and targets entities that need this capability, the IPersonalInfoUpdate interface offers a simple contract for changing personal information, such as contact data. By avoiding bloated and excessively complicated interfaces, this design promotes cohesiveness and guarantees that classes only implement the functionality that is pertinent to their duties.

- **Dependency Inversion Principle (DIP)**
  - The Dependency Inversion Principle (DIP) emphasizes that high-level modules should depend on abstractions rather than low-level implementations. For example, the AppointmentServiceFacade enables high-level modules, such as menu classes, to communicate via a single interface by abstracting interactions with several underlying services, such as DoctorService, PatientService, and AppointmentService. By separating high-level functionality from particular implementations, interfaces such as IPatientMedicalRecordAccess and IPersonalInfoUpdate facilitate the smooth integration of new components. Furthermore, by centralizing user input functionality, the InputHandler lessens the need for menu classes to handle user input in-depth. The system's reliance on abstractions guarantees loose coupling, which improves its testability, maintainability, and flexibility in response to future developments.

By adhering to the SOLID principles, the project achieves maintainability, extensibility, reusability, and testability. Each principle contributes to a system that is scalable, robust, and easy to evolve, ensuring that new requirements can be integrated smoothly without disrupting existing functionality.

Extensibility & Maintainability

Our design adheres to modular principles, promoting extensibility and maintainability through logical separation of concerns, leveraging design patterns, and implementing the SOLID principles. The following points demonstrate how these goals were achieved:

1. **User Type Functionality Extensions**:
   The updatePersonalInfo functionality for Patients can be seamlessly extended to other user types, such as Doctors, Pharmacists, or Admins, by creating or modifying shared interfaces like IPersonalInfoUpdate. Similarly, new features for any user type can be added without affecting the existing ones due to the loosely coupled design.
2. **Role-Specific Menus and Operations**:
   By designing role-specific menus (e.g., DoctorMenu, AdminMenu), the program is highly maintainable and can adapt to new user roles by creating additional menu classes. These menus interact with the facades, ensuring a consistent interface and preventing menu logic from becoming tangled with service-level details.
3. **Integration of Additional Features**:
   Future requirements, such as adding new appointment management rules (e.g., dynamic slot durations or priority appointments), can be integrated by extending existing services without impacting other components. The Appointment-related classes (e.g., AppointmentService, AppointmentOutcomeRecord) are designed to accommodate additional fields or methods without disrupting the structure.
4. **Reduction of Redundant Implementations**:
   The centralization of functionality into facades and utility classes reduces code duplication and promotes maintainability. For example, the resetPassword functionality could be refactored into a centralized handler to eliminate the current repetition across multiple menus.
5. **Input Validation and Error Handling**:
   Input validation via the InputHandler and regex ensures robustness and reduces fragility in user interactions. Future enhancements to input validation (e.g., support for dynamic validation rules) can be implemented centrally without altering individual menu classes.
6. **Separation of Logic**:
   Each class is focused on a singular responsibility. For instance, StaffServiceFacade handles staff-related operations, while MedicineInventory is solely responsible for managing medicines. This separation ensures that changes in one part of the system (e.g., modifying staff attributes) do not cascade into unrelated areas.

By leveraging these principles, the system is designed to accommodate future enhancements with minimal disruption to existing functionality. This maintainable and extensible architecture ensures that the software remains robust and scalable over time

Additional Features & Functionalities

- Password Reset Functionality
- Regex Pattern Matching for input validation to promote robustness of program and reduce fragility
- Try to switch inputs to integer MCQ as much as possible to reduce likelihood of error, spelling out dates or medications more error prone than entering a number that tallies with the data, we can process in the program logic.
- Incorporation of several design patterns, Facade, Command and Singleton, which ensured we could implement SOLID principles in an organized, efficient and modular manner

Future opportunities for extension

- Billing functionalities which would have dependencies to AppointmentOutcomeRecord and PatientMedicalRecord and produces an invoice when an appointment is marked as COMPLETED. A singleton class to hold invoices or adding functionalities to process billing in AppointmentService will be helpful if we implement an InsuranceManagement class
- Nurse classes which would connect to Doctor and AppointmentOutcomeRecord classes. Nurses would be able to gather data by a pre-consultation check-up by recording weight, height, labs etc. These results can be collated in a PreConsultationData class that can be incorporated into an Appointment class.

Reflections

The process of planning and designing this project presented significant challenges. One of the most complex aspects was ensuring the integrity of the system. For example, we needed to prevent duplicate appointment slots for a given doctor, which could result in double-bookings. Addressing these issues required careful thought, independent research, and the application of appropriate design patterns, which we had to identify and execute on our own.

Another challenge lay in adhering to the SOLID design principles to ensure extensibility and maintainability. It was particularly tempting to consolidate all functionalities within the respective class menu pages, which would have violated the Single Responsibility Principle (SRP) and led to poor design practices. Given the constraints of time, we strived to balance implementation and refactoring, making iterative improvements to maintain design quality wherever possible.

In addition to technical difficulties, we encountered interpersonal challenges. Coordinating with team members who were unresponsive or lacked the competence to deliver their assigned tasks was particularly taxing. Moreover, integrating the contributions and logic of multiple developers proved difficult. Despite frequent meetings, misunderstandings often surfaced during the implementation phase, highlighting the importance of clear and consistent communication within a team. This project underscored the critical role of effective teamwork.

Time management was another significant hurdle, especially given our academic commitments during the final examination period. Some team members had exams scheduled during the week—and even on the day—of the project presentation. As a result, we had to reallocate tasks among a smaller group, which increased the workload for the remaining contributors. In project management, resources such as time, manpower, and budget are typically balanced; however, as this was a school project, financial resources were not an option for addressing the manpower shortage. The reduced team size inevitably led to increased time pressure and compromises in project execution.

Overall working in Object-Oriented Programming has been a whole lot more fun than procedural. Although C is more versatile, that structure in Java just provides a lot less of a headache in planning and debugging is much simpler. We are truly thankful to our professors and our TA for being very kind, understanding and promptly making alternate arrangements for us for the presentation and submission.