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 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 actions may be taken.

Name	Course (CE2002 or CZ2002)	Lab Group	Signature/Date
Darrell Ma Wei Ze	CZ2002	SCS7/Group 4	17/11/2024
Joyce Chen	CZ2002	SCS7/Group 4	17/11/2024
Kim Dobin	CZ2002	SCS7/Group 4	17/11/2024
Sam Chern Kai, Chester	CZ2002	SCS7/Group 4	17/11/2024
Shaun Cheong Jia Yong	CZ2002	SCS7/Group 4	17/11/2024
Yang Jerry	CZ2002	SCS7/Group 4	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.

1. Design Considerations and Approach Taken

For our design approach, we decided to tackle the project application through the lens of Object-Oriented Programming (OOP) principles: Encapsulation, Inheritance, Polymorphism and Abstraction. These principles provide a robust and flexible foundation that will be the basis of a maintainable, extensible and modular software.

We began by creating the Unified Modeling Language (UML) class diagram, adhering as strictly as possible to the SOLID design principles taught. We ensured persistent data storage by having the program read and write to the Comma-Separated Values (CSV) files provided. The program uses the necessary objects such as Users from the data from the CSV files. Any new data is written back to the CSV files, ensuring that the dataset is retained and the program can resume smoothly on the next start-up.

In order to properly address the requirements of the assignment, we added new data columns to the CSV files when necessary. We also implemented comprehensive error checking and adopted a user-friendly interface with a simple and intuitive design that prompts for correct user input whenever erroneous entries are detected.

We considered the use of the Model-View-Controller(MVC) architecture but chose to stay with the simple and effective SOLID design principles. Reasons for this include potentially an increase in difficulty of managing dependencies, limited suitability for real-time applications, as well as the architecture being too advanced and complex.

The separation of the application into three components in MVC could potentially create tightly coupled dependencies, particularly when scaling the system. This makes it challenging to manage interactions between Model, View, and Controller.

Furthermore, in this case for Hospital Management System (HMS), timely data access and updates are essential for effective patient care and efficient workflow. For example, hospital staff often require the latest information on patient status, appointments and medical history to ensure that proper treatments are administered. Patients booking appointments also require real-time feedback on appointment availability to avoid double bookings.

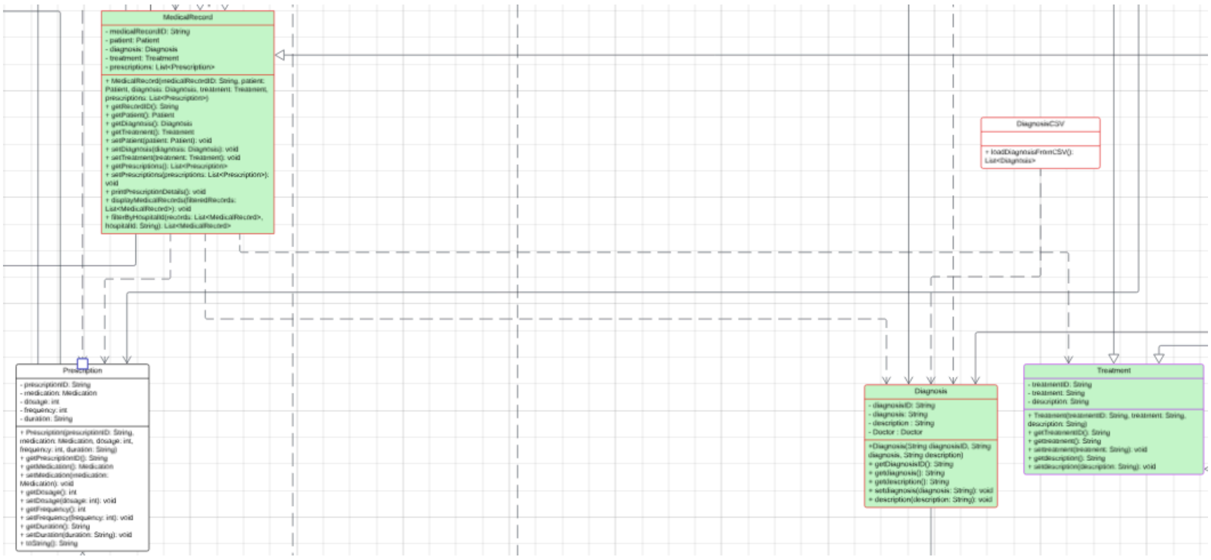
Real-time synchronisation is especially crucial when multiple staff members need to access and update patient records simultaneously. Thus, we chose to steer away from MVC architecture. Ultimately, when designing our application, we mainly adhered to OOP and SOLID design principles which will be further elaborated.

2. Principles Used

a. Single Responsibility Principle (SRP)

During the design of the application, we ensured that every class has a clear, single responsibility, resulting in loose coupling and easy troubleshooting. For example, we split the medical records of a patient into three specialised classes (Figure 1), Diagnosis, Treatment and Prescription.

Figure 1: Application of Single Responsibility Principle


MedicalRecord
- medicalRecordID: String
- patient: Patient
- diagnosis: Diagnosis
- treatment: Treatment
- prescriptions: List<Prescription>
+ MedicalRecord(medicalRecordID: String, patient: Patient, diagnosis: Diagnosis, treatment: Treatment, prescriptions: List<Prescription>)
+ getRecordID(): String
+ getPatient(): Patient
+ getDiagnosis(): Diagnosis
+ getTreatment(): Treatment
+ setPatient(patient: Patient): void
+ setDiagnosis(diagnosis: Diagnosis): void
+ setTreatment(treatment: Treatment): void
+ getPrescriptions(): List<Prescription>
+ setPrescriptions(prescriptions: List<Prescription>): void
+ printPrescriptionDetails(): void
+ displayMedicalRecords(filteredRecords: List<MedicalRecord>): void
+ filterByHospitalId(records: List<MedicalRecord>, hospitalId: String): List<MedicalRecord>
DiagnosisCSV
+ loadDiagnosisFromCSV(): List<Diagnosis>
Prescription
- prescriptionID: String
- medication: Medication
- dosage: int
- frequency: int
- duration: String
+ Prescription(prescriptionID: String, medication: Medication, dosage: int, frequency: int, duration: String)
+ getPrescriptionID(): String
+ getMedication(): Medication
+ setMedication(medication: Medication): void
+ getDosage(): int
+ setDosage(dosage: int): void
+ getFrequency(): int
+ setFrequency(frequency: int): void
+ getDuration(): String
+ setDuration(duration: String): void
+ toString(): String
Diagnosis
- diagnosisID: String
- diagnosis: String
- description : String
- Doctor : Doctor
+Diagnosis(String diagnosisID, String diagnosis, String description)
+ getDiagnosisID(): String
+ getdiagnosis(): String
+ getdescription(): String
+ setdiagnosis(diagnosis: String): void
+ description(description: String): void
Treatment
- treatmentID: String
- treatment: String
- description: String
+ Treatment(treatmentID: String, treatment: String, description: String)
+ getTreatmentID(): String
+ gettreatment(): String
+ settreatment(treatment: String): void
+ getdescription(): String
+ setdescription(description: String): void

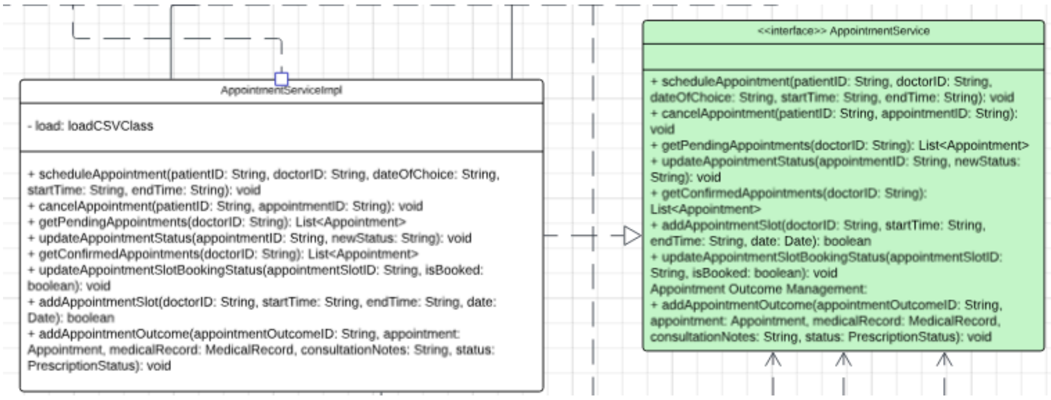
Diagnosis, Treatment and Prescription classes have their individual attributes and focus specifically on distinct aspects of a patient's medical records. This separation enables changes to be made to one part of the system without impacting unrelated components, and facilitates easier maintenance and troubleshooting of any bugs due to the system's modularity. These individual classes are then aggregated under the MedicalRecord class, which manages the patient's entire medical record by referencing Diagnosis, Treatment and Prescription. The MedicalRecord class is responsible for managing the patient's medical history without being concerned with each individual component, thus adhering to SRP.

b. Open-Closed Principle (OCP)

OCP was implemented during the design of the application. We designed the program to be modular, ensuring it is closed to modification but open for extension by enabling the addition of new functionality without affecting the existing code. For instance, the AppointmentServiceImpl implements the AppointmentService interface, which contains methods for viewing, scheduling, rescheduling and cancelling their appointments (Figure 2).

The interface embodies OCP as it can be extended to support new classes. For example, a new class could be created to implement and provide its own version of the same methods, while the AppointmentService interface remains closed and unaltered.

Figure 2: Application of Open-Closed Principle


AppointmentServiceImpl
- load: loadCSVClass
+ scheduleAppointment(patientID: String, doctorID: String, dateOfChoice: String, startTime: String, endTime: String): void
+ cancelAppointment(patientID: String, appointmentID: String): void
+ getPendingAppointments(doctorID: String): List<Appointment>
+ updateAppointmentStatus(appointmentID: String, newStatus: String): void
+ getConfirmedAppointments(doctorID: String): List<Appointment>
+ updateAppointmentSlotBookingStatus(appointmentSlotID: String, isBooked: boolean): void
+ addAppointmentSlot(doctorID: String, startTime: String, endTime: String, date: Date): boolean
+ addAppointmentOutcome(appointmentOutcomeID: String, appointment: Appointment, medicalRecord: MedicalRecord, consultationNotes: String, status: PrescriptionStatus): void
<<interface>> AppointmentService
+ scheduleAppointment(patientID: String, doctorID: String, dateOfChoice: String, startTime: String, endTime: String): void
+ cancelAppointment(patientID: String, appointmentID: String): void
+ getPendingAppointments(doctorID: String): List<Appointment>
+ updateAppointmentStatus(appointmentID: String, newStatus: String): void
+ getConfirmedAppointments(doctorID: String): List<Appointment>
+ addAppointmentSlot(doctorID: String, startTime: String, endTime: String, date: Date): boolean
+ updateAppointmentSlotBookingStatus(appointmentSlotID: String, isBooked: boolean): void
Appointment Outcome Management:
+ addAppointmentOutcome(appointmentOutcomeID: String, appointment: Appointment, medicalRecord: MedicalRecord, consultationNotes: String, status: PrescriptionStatus): void

c. Liskov Substitution Principle (LSP)

LSP states that any subclasses of a base class can be used interchangeably with that base class without altering the class's expected behaviour. We ensured that subclasses have the same pre and post-conditions. For example, the steps that a Doctor takes to login should not differ significantly from those of a Patient. This means that the system is able to handle different user types polymorphically, providing customised access levels without the need to rewrite code. This ensures that new user types or roles can be added seamlessly, allowing for extension without modifying core user methods repeatedly.

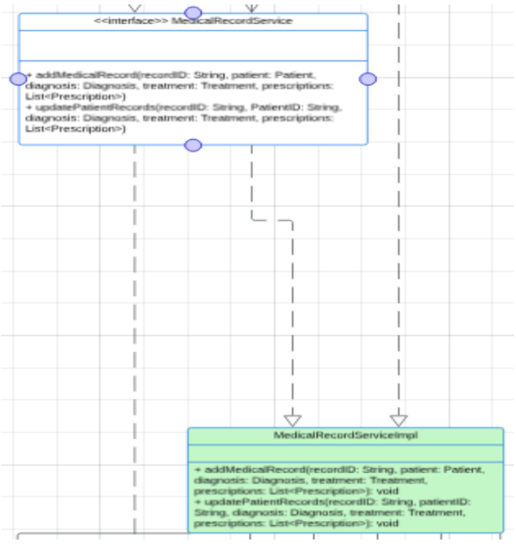
d. Interface Segregation Principle (ISP)

ISP states that a class should not be forced to implement interfaces that it does not use. The UserAuthentication interface is focused solely on login and password actions, and only the User class implements it. This aligns with OOP principles such as encapsulation and abstraction, ensuring that classes are limited to viewing and using methods essential to them. The user experience is also streamlined by providing the necessary methods relevant to each role.

e. Dependency Inversion Principle (DIP)

DIP emphasises that high-level modules should depend on abstractions and interfaces rather than concrete implementations. For our HMS, DIP is shown through the use of interfaces such as AppointmentService and MedicalRecordService, which define abstract contracts for behaviour, and their concrete implementations, such as AppointmentServiceImpl and Medical Record ServiceImpl (Figure 3). This decoupling allows high-level modules to depend on abstractions rather than the concrete classes.

Figure 3: Application of Dependency Inversion Principle



<<interface>> MedicalRecordService
+ addMedicalRecord(recordID: String, patient: Patient, diagnosis: Diagnosis, treatment: Treatment, prescriptions: List<Prescription>)
+ updatePatientRecords(recordID: String, PatientID: String, diagnosis: Diagnosis, treatment: Treatment, prescriptions: List<Prescription>)
MedicalRecordServiceImpl
+ addMedicalRecord(recordID: String, patient: Patient, diagnosis: Diagnosis, treatment: Treatment, prescriptions: List<Prescription>): void
+ updatePatientRecords(recordID: String, patientID: String, diagnosis: Diagnosis, treatment: Treatment, prescriptions: List<Prescription>): void

This approach not only aligns with polymorphism, but also promotes encapsulation by hiding implementation details behind interfaces. Abstraction is also achieved by defining clear interfaces for the various components.

3. Assumptions Made

When creating the UML class diagram, the following key assumptions were made:

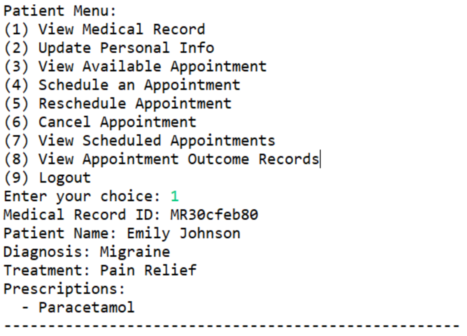
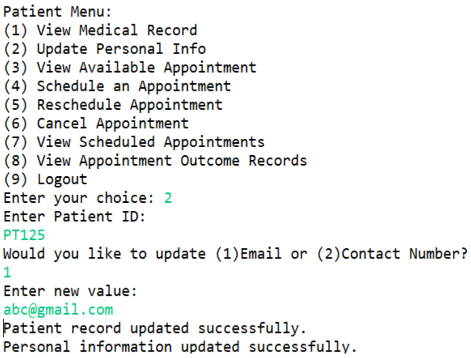
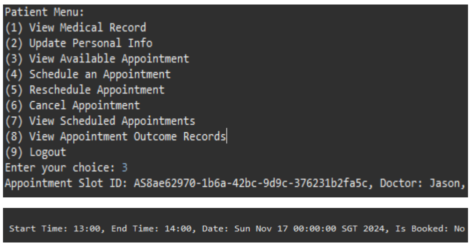
- Passwords and user credentials are stored securely.
- Any updates to the system, such as appointment status, are updated in real-time and synchronised across all users. This prevents issues such as double-booking the same time slot.
- A time buffer exists to prevent last-minute cancellation and rescheduling.
- All initial data imports are made in the standard CSV format.
- Low stock alerts trigger internal notifications for Pharmacists and Administrators.
- The system is designed for a single hospital and currently does not support integration with other hospitals.
- Data backups and recovery are handled externally and periodically and thus not part of the system's core functionality.

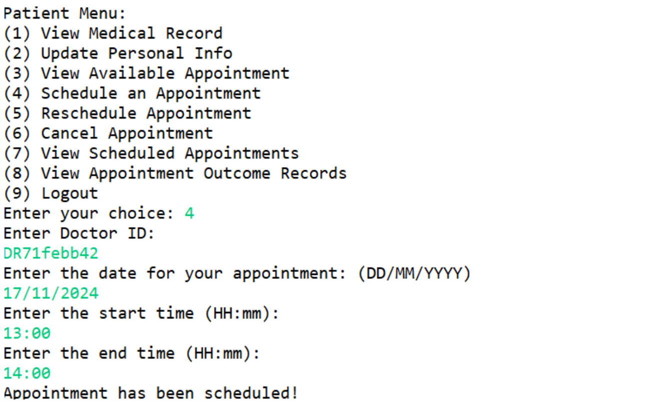
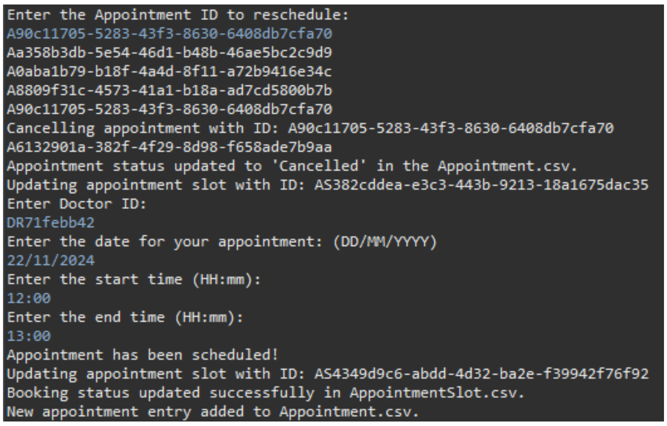
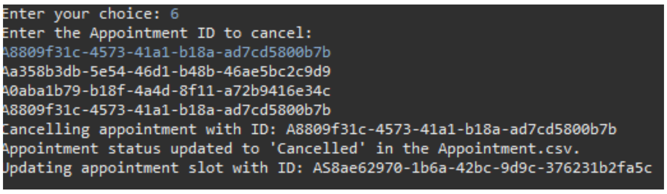
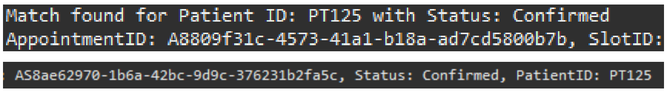
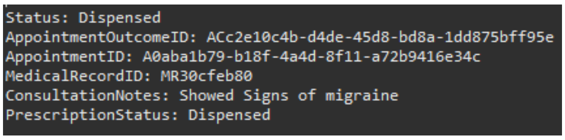
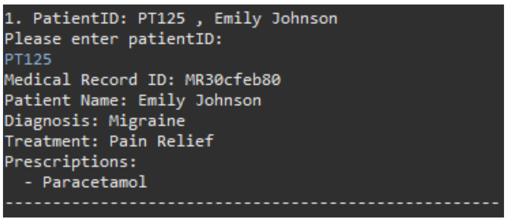
- Consistency checks are performed to ensure that updates to medical records, prescriptions, and inventory levels do not create conflicts.

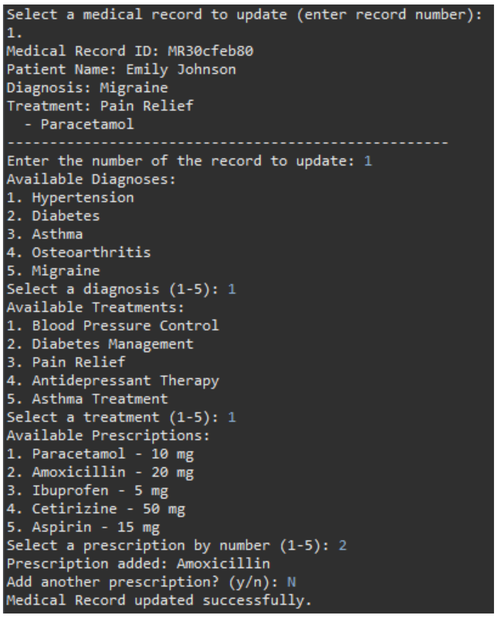
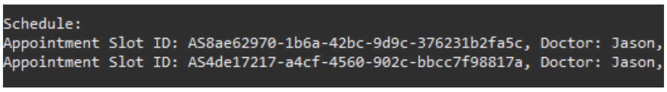
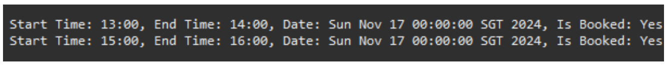
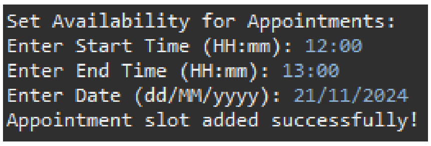
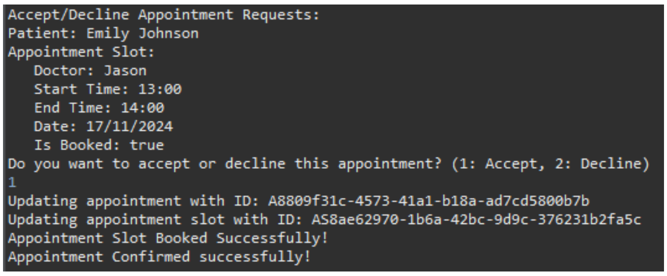
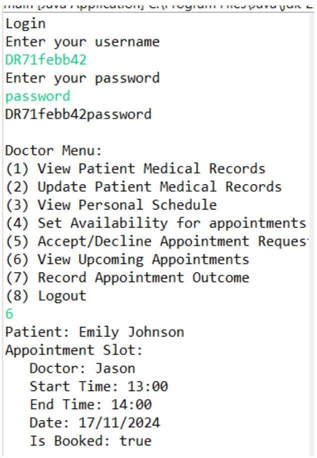
4. Detailed UML Class Diagram

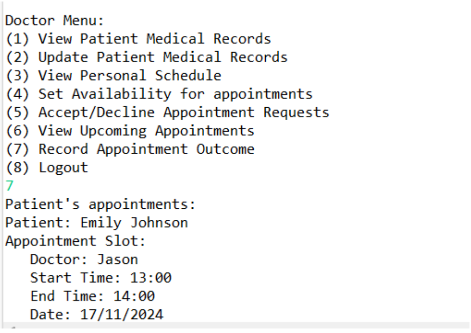
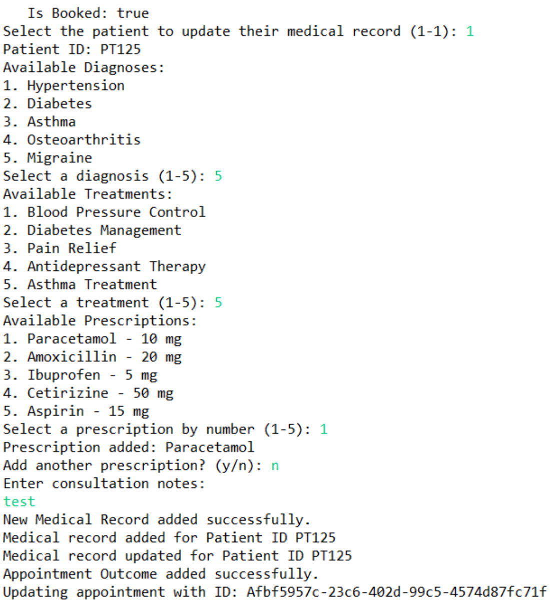
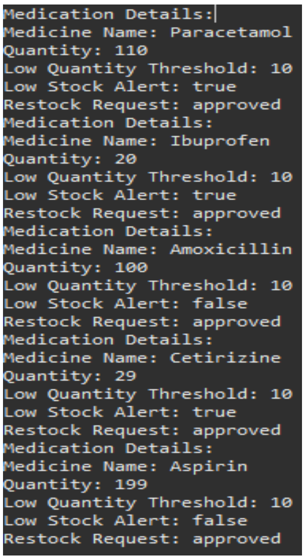
Refer to the PDF of the UML Class Diagram in the folder.

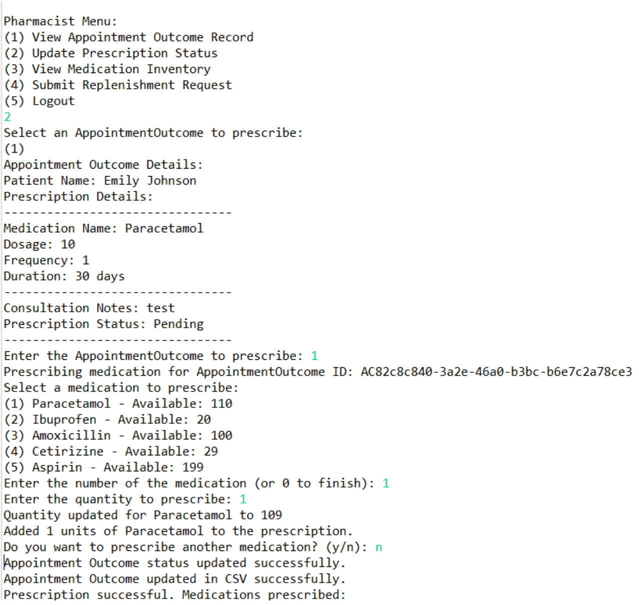
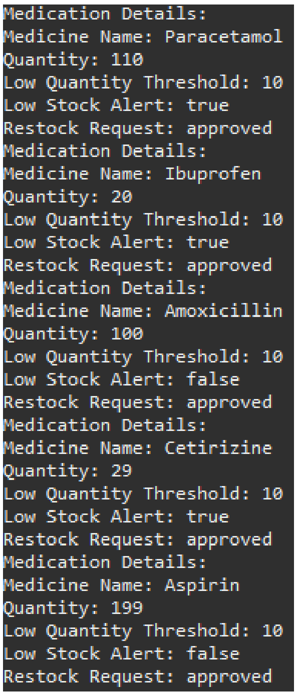
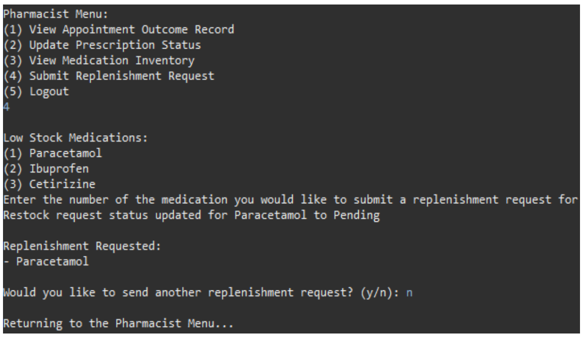
5. Testing

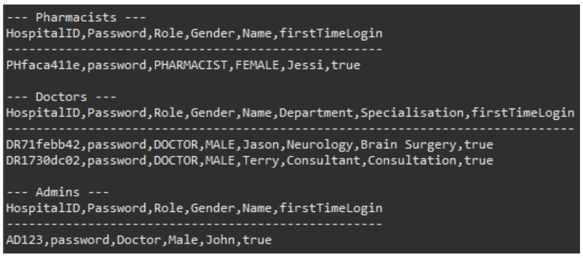
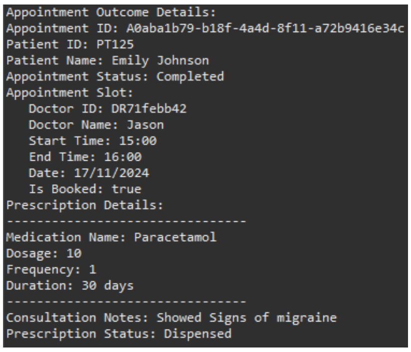
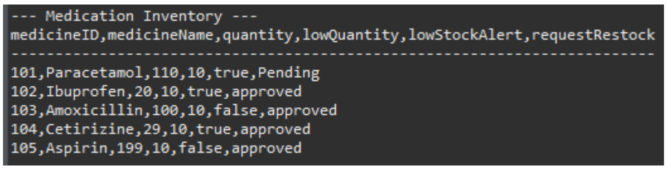
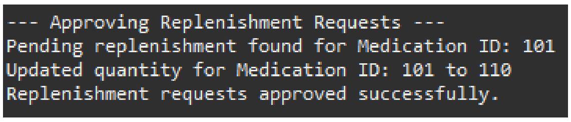
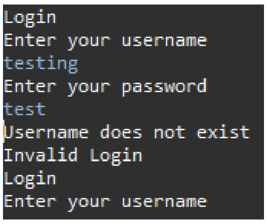
Test Cases	Input	Output
Patient: View Medical Record	1	 Patient Menu: (1) View Medical Record (2) Update Personal Info (3) View Available Appointment (4) Schedule an Appointment (5) Reschedule Appointment (6) Cancel Appointment (7) View Scheduled Appointments (8) View Appointment Outcome Records (9) Logout Enter your choice: 1 Medical Record ID: MR30cfeb80 Patient Name: Emily Johnson Diagnosis: Migraine Treatment: Pain Relief Prescriptions: - Paracetamol --
Patient: Update Personal Information	2 PT125 1 abc@gmail.com	 Patient Menu: (1) View Medical Record (2) Update Personal Info (3) View Available Appointment (4) Schedule an Appointment (5) Reschedule Appointment (6) Cancel Appointment (7) View Scheduled Appointments (8) View Appointment Outcome Records (9) Logout Enter your choice: 2 Enter Patient ID: PT125 Would you like to update (1)Email or (2)Contact Number? 1 Enter new value: abc@gmail.com Patient record updated successfully. Personal information updated successfully.
Patient: View Available Appointment Slots	PT123 Password3 3	 Patient Menu: (1) View Medical Record (2) Update Personal Info (3) View Available Appointment (4) Schedule an Appointment (5) Reschedule Appointment (6) Cancel Appointment (7) View Scheduled Appointments (8) View Appointment Outcome Records (9) Logout Enter your choice: 3 Appointment Slot ID: AS8ae62970-1b6a-42bc-9d9c-376231b2fa5c, Doctor: Jason, Start Time: 13:00, End Time: 14:00, Date: Sun Nov 17 00:00:00 SGT 2024, Is Booked: No

Patient: Schedule an Appointment	4 DR71febb42 17/11/2024 13:00 14:00	 Patient Menu: (1) View Medical Record (2) Update Personal Info (3) View Available Appointment (4) Schedule an Appointment (5) Reschedule Appointment (6) Cancel Appointment (7) View Scheduled Appointments (8) View Appointment Outcome Records (9) Logout Enter your choice: 4 Enter Doctor ID: DR71febb42 Enter the date for your appointment: (DD/MM/YYYY) 17/11/2024 Enter the start time (HH:mm): 13:00 Enter the end time (HH:mm): 14:00 Appointment has been scheduled!
Patient: Reschedule an Appointment	5 A90c11705-5283-43f3-8630-6408db7cfa70 DR71febb42 22/11/2024 12:00 13:00	 Enter the Appointment ID to reschedule: A90c11705-5283-43f3-8630-6408db7cfa70 Aa358b3db-5e54-46d1-b48b-46ae5bc2c9d9 A0aba1b79-b18f-4a4d-8f11-a72b9416e34c A8809f31c-4573-41a1-b18a-ad7cd5800b7b A90c11705-5283-43f3-8630-6408db7cfa70 Cancelling appointment with ID: A90c11705-5283-43f3-8630-6408db7cfa70 A6132901a-382f-4f29-8d98-f658ade7b9aa Appointment status updated to 'Cancelled' in the Appointment.csv. Updating appointment slot with ID: AS382cddea-e3c3-443b-9213-18a1675dac35 Enter Doctor ID: DR71febb42 Enter the date for your appointment: (DD/MM/YYYY) 22/11/2024 Enter the start time (HH:mm): 12:00 Enter the end time (HH:mm): 13:00 Appointment has been scheduled! Updating appointment slot with ID: AS4349d9c6-abdd-4d32-ba2e-f39942f76f92 Booking status updated successfully in AppointmentSlot.csv. New appointment entry added to Appointment.csv.
Patient: Cancel an Appointment	6 A8809f31c-4573-41a1-b18a-ad7cd5800b7b	 Enter your choice: 6 Enter the Appointment ID to cancel: A8809f31c-4573-41a1-b18a-ad7cd5800b7b Aa358b3db-5e54-46d1-b48b-46ae5bc2c9d9 A0aba1b79-b18f-4a4d-8f11-a72b9416e34c A8809f31c-4573-41a1-b18a-ad7cd5800b7b Cancelling appointment with ID: A8809f31c-4573-41a1-b18a-ad7cd5800b7b Appointment status updated to 'Cancelled' in the Appointment.csv. Updating appointment slot with ID: AS8ae62970-1b6a-42bc-9d9c-376231b2fa5c
Patient: View Scheduled Appointments	7	 Match found for Patient ID: PT125 with Status: Confirmed AppointmentID: A8809f31c-4573-41a1-b18a-ad7cd5800b7b, SlotID: AS8ae62970-1b6a-42bc-9d9c-376231b2fa5c, Status: Confirmed, PatientID: PT125
Patient: View Past Appointment Outcome Records	8	 Status: Dispensed AppointmentOutcomeID: ACc2e10c4b-d4de-45d8-bd8a-1dd875bff95e AppointmentID: A0aba1b79-b18f-4a4d-8f11-a72b9416e34c MedicalRecordID: MR30cfeb80 ConsultationNotes: Showed Signs of migraine PrescriptionStatus: Dispensed
Doctor: View Patient Medical Records	DR71febb42 password 1 PT125	 1. PatientID: PT125 , Emily Johnson Please enter patientID: PT125 Medical Record ID: MR30cfeb80 Patient Name: Emily Johnson Diagnosis: Migraine Treatment: Pain Relief Prescriptions: - Paracetamol --

Doctor: Update Patient Medical Records	2 PT125 1 1 2 N	 Select a medical record to update (enter record number): 1. Medical Record ID: MR30cfeb80 Patient Name: Emily Johnson Diagnosis: Migraine Treatment: Pain Relief - Paracetamol --- Enter the number of the record to update: 1 Available Diagnoses: 1. Hypertension 2. Diabetes 3. Asthma 4. Osteoarthritis 5. Migraine Select a diagnosis (1-5): 1 Available Treatments: 1. Blood Pressure Control 2. Diabetes Management 3. Pain Relief 4. Antidepressant Therapy 5. Asthma Treatment Select a treatment (1-5): 1 Available Prescriptions: 1. Paracetamol - 10 mg 2. Amoxicillin - 20 mg 3. Ibuprofen - 5 mg 4. Cetirizine - 50 mg 5. Aspirin - 15 mg Select a prescription by number (1-5): 2 Prescription added: Amoxicillin Add another prescription? (y/n): N Medical Record updated successfully.
Doctor: View Personal Schedule	DR71febb42 Password 3	 Schedule: Appointment Slot ID: AS8ae62970-1b6a-42bc-9d9c-376231b2fa5c, Doctor: Jason, Appointment Slot ID: AS4de17217-a4cf-4560-902c-bbcc7f98817a, Doctor: Jason,  Start Time: 13:00, End Time: 14:00, Date: Sun Nov 17 00:00:00 SGT 2024, Is Booked: Yes Start Time: 15:00, End Time: 16:00, Date: Sun Nov 17 00:00:00 SGT 2024, Is Booked: Yes
Doctor: Set Availability for Appointments	4 12:00 13:00 21/11/2024	 Set Availability for Appointments: Enter Start Time (HH:mm): 12:00 Enter End Time (HH:mm): 13:00 Enter Date (dd/MM/yyyy): 21/11/2024 Appointment slot added successfully!
Doctor: Accept or Decline Appointment Requests	5 1	 Accept/Decline Appointment Requests: Patient: Emily Johnson Appointment Slot: Doctor: Jason Start Time: 13:00 End Time: 14:00 Date: 17/11/2024 Is Booked: true Do you want to accept or decline this appointment? (1: Accept, 2: Decline) 1 Updating appointment with ID: A8809f31c-4573-41a1-b18a-ad7cd5800b7b Updating appointment slot with ID: AS8ae62970-1b6a-42bc-9d9c-376231b2fa5c Appointment Slot Booked Successfully! Appointment Confirmed successfully!
Doctor: View Upcoming Appointments	DR71febb42 Password 6	 Login Enter your username DR71febb42 Enter your password password DR71febb42password Doctor Menu: (1) View Patient Medical Records (2) Update Patient Medical Records (3) View Personal Schedule (4) Set Availability for appointments (5) Accept/Decline Appointment Reques (6) View Upcoming Appointments (7) Record Appointment Outcome (8) Logout 6 Patient: Emily Johnson Appointment Slot: Doctor: Jason Start Time: 13:00 End Time: 14:00 Date: 17/11/2024 Is Booked: true

Doctor: Record Appointment Outcome	7 1 5 5 n test	 Doctor Menu: (1) View Patient Medical Records (2) Update Patient Medical Records (3) View Personal Schedule (4) Set Availability for appointments (5) Accept/Decline Appointment Requests (6) View Upcoming Appointments (7) Record Appointment Outcome (8) Logout 7 Patient's appointments: Patient: Emily Johnson Appointment Slot: Doctor: Jason Start Time: 13:00 End Time: 14:00 Date: 17/11/2024  Is Booked: true Select the patient to update their medical record (1-1): 1 Patient ID: PT125 Available Diagnoses: 1. Hypertension 2. Diabetes 3. Asthma 4. Osteoarthritis 5. Migraine Select a diagnosis (1-5): 5 Available Treatments: 1. Blood Pressure Control 2. Diabetes Management 3. Pain Relief 4. Antidepressant Therapy 5. Asthma Treatment Select a treatment (1-5): 5 Available Prescriptions: 1. Paracetamol - 10 mg 2. Amoxicillin - 20 mg 3. Ibuprofen - 5 mg 4. Cetirizine - 50 mg 5. Aspirin - 15 mg Select a prescription by number (1-5): 1 Prescription added: Paracetamol Add another prescription? (y/n): n Enter consultation notes: test New Medical Record added successfully. Medical record added for Patient ID PT125 Medical record updated for Patient ID PT125 Appointment Outcome added successfully. Updating appointment with ID: Afbf5957c-23c6-402d-99c5-4574d87fc71f
Pharmacist: View Appointment Outcome Record	PHfaca411e password 3	 Medication Details: Medicine Name: Paracetamol Quantity: 110 Low Quantity Threshold: 10 Low Stock Alert: true Restock Request: approved Medication Details: Medicine Name: Ibuprofen Quantity: 20 Low Quantity Threshold: 10 Low Stock Alert: true Restock Request: approved Medication Details: Medicine Name: Amoxicillin Quantity: 100 Low Quantity Threshold: 10 Low Stock Alert: false Restock Request: approved Medication Details: Medicine Name: Cetirizine Quantity: 29 Low Quantity Threshold: 10 Low Stock Alert: true Restock Request: approved Medication Details: Medicine Name: Aspirin Quantity: 199 Low Quantity Threshold: 10 Low Stock Alert: false Restock Request: approved

Pharmacist: Update Prescription Status	2 1 1 1 n	 Pharmacist Menu: (1) View Appointment Outcome Record (2) Update Prescription Status (3) View Medication Inventory (4) Submit Replenishment Request (5) Logout 2 Select an AppointmentOutcome to prescribe: (1) Appointment Outcome Details: Patient Name: Emily Johnson Prescription Details: -------------------------------- Medication Name: Paracetamol Dosage: 10 Frequency: 1 Duration: 30 days -------------------------------- Consultation Notes: test Prescription Status: Pending -------------------------------- Enter the AppointmentOutcome to prescribe: 1 Prescribing medication for AppointmentOutcome ID: AC82c8c840-3a2e-46a0-b3bc-b6e7c2a78ce3 Select a medication to prescribe: (1) Paracetamol - Available: 110 (2) Ibuprofen - Available: 20 (3) Amoxicillin - Available: 100 (4) Cetirizine - Available: 29 (5) Aspirin - Available: 199 Enter the number of the medication (or 0 to finish): 1 Enter the quantity to prescribe: 1 Quantity updated for Paracetamol to 109 Added 1 units of Paracetamol to the prescription. Do you want to prescribe another medication? (y/n): n Appointment Outcome status updated successfully. Appointment Outcome updated in CSV successfully. Prescription successful. Medications prescribed:
Pharmacist: View Medication Inventory	3	 Medication Details: Medicine Name: Paracetamol Quantity: 110 Low Quantity Threshold: 10 Low Stock Alert: true Restock Request: approved Medication Details: Medicine Name: Ibuprofen Quantity: 20 Low Quantity Threshold: 10 Low Stock Alert: true Restock Request: approved Medication Details: Medicine Name: Amoxicillin Quantity: 100 Low Quantity Threshold: 10 Low Stock Alert: false Restock Request: approved Medication Details: Medicine Name: Cetirizine Quantity: 29 Low Quantity Threshold: 10 Low Stock Alert: true Restock Request: approved Medication Details: Medicine Name: Aspirin Quantity: 199 Low Quantity Threshold: 10 Low Stock Alert: false Restock Request: approved
Pharmacist: Submit Replenishment Request	PHfaca411e Password 4 1 n	 Pharmacist Menu: (1) View Appointment Outcome Record (2) Update Prescription Status (3) View Medication Inventory (4) Submit Replenishment Request (5) Logout 4 Low Stock Medications: (1) Paracetamol (2) Ibuprofen (3) Cetirizine Enter the number of the medication you would like to submit a replenishment request for Restock request status updated for Paracetamol to Pending Replenishment Requested: - Paracetamol Would you like to send another replenishment request? (y/n): n Returning to the Pharmacist Menu...

Administrator: View and Manage Hospital Staff	AD123 Password 1 4	 --- Pharmacists --- HospitalID,Password,Role,Gender,Name,firstTimeLogin --- PHfaca411e,password,PHARMACIST,FEMALE,Jessi,true --- Doctors --- HospitalID,Password,Role,Gender,Name,Department,Specialisation,firstTimeLogin --- DR71febb42,password,DOCTOR,MALE,Jason,Neurology,Brain Surgery,true DR1730dc02,password,DOCTOR,MALE,Terry,Consultant,Consultation,true --- Admins --- HospitalID,Password,Role,Gender,Name,firstTimeLogin --- AD123,password,Doctor,Male,John,true
Administrator: View Appointment Details	AD123 Password 5	 Appointment Outcome Details: Appointment ID: A0aba1b79-b18f-4a4d-8f11-a72b9416e34c Patient ID: PT125 Patient Name: Emily Johnson Appointment Status: Completed Appointment Slot: Doctor ID: DR71febb42 Doctor Name: Jason Start Time: 15:00 End Time: 16:00 Date: 17/11/2024 Is Booked: true Prescription Details: ------------------------------ Medication Name: Paracetamol Dosage: 10 Frequency: 1 Duration: 30 days ------------------------------ Consultation Notes: Showed Signs of migraine Prescription Status: Dispensed
Administrator: View and Manage Medication Inventory	AD123 Password 6 1	 --- Medication Inventory --- medicineID,medicineName,quantity,lowQuantity,lowStockAlert,requestRestock --- 101,Paracetamol,110,10,true,Pending 102,Ibuprofen,20,10,true,approved 103,Amoxicillin,100,10,false,approved 104,Cetirizine,29,10,true,approved 105,Aspirin,199,10,false,approved
Administrator: Approve Replenishment Requests	AD123 Password 7	 --- Approving Replenishment Requests --- Pending replenishment found for Medication ID: 101 Updated quantity for Medication ID: 101 to 110 Replenishment requests approved successfully.
Login System: Login with Incorrect Credentials	testing test	 Login Enter your username testing Enter your password test Username does not exist Invalid Login Login Enter your username

6. Reflection

a. Difficulties Encountered and Workarounds

During the initial planning of the system, we decided to first create the UML diagram to have a coordinated and general overview of the classes to be implemented. Thereafter, we divided the work up among each group member and collated the individual parts once done. However, we ran into multiple issues, including the classes from different individuals being interdependent. This made it challenging for us to complete the work independently.

To tackle this issue, we closely collaborated and regularly communicated, frequently updating each other on our progress to ensure alignment. We also practised loose coupling wherever possible, minimising interdependencies between classes.

Another issue we ran into was that the preliminary UML class diagram lacked many required features and did not fully align with the intended SOLID design principles. Hence, many revisions were required to refine the implementation, apply the OOP principles effectively and avoid violations in the UML structure. The necessitated frequent discussions and feedback allowed us to work together and address issues.

b. Knowledge Learnt from Course

During the creation of this HMS, we gained a deeper appreciation for OOP principles and the SOLID design principle, as they offered a basis for collaboration and simplification of the design. By adhering these principles, we managed to create a system that was extensible, scaleable, maintainable and robust. This ensured that any new feature could be easily implemented with minimal disruption.

Through the collaborative effort of this group project, we learnt to resolve differences and rely on each other's strengths, effectively dividing and conquering the challenges of building this system in a timely and efficient manner. We also had to engage in self-learning to import and use libraries such as "LocalDate" and "LocalTime", and to implement read/write methods for CSV files, ensuring data retention between startups.

c. Further improvement suggestions for the course

To make the course more engaging and practical, the inclusion of real-world data could be beneficial. We could better evaluate how practical our system is in realistic scenarios. For instance, anonymized patient files could be provided, and datasets could include common real-world issues like typos, duplicates, and missing values. Integrating hands-on experience with complementary technical skills alongside OOP principles could potentially give more valuable insights, especially when these skills are likely required during internships.

7. Link to Github repository

https://github.com/Exhilion/SC2002-Project