GitHub Repository: https://github.com/firepiratex/SC2002

1. Design Considerations and Design Approach Taken

a. Design Considerations

We will be designing this application by using "Handler" classes as a “Manager” to separate method definitions from entity classes, which contains the data information attributes. The entity classes rely on these Handler classes to manipulate the data effectively. Additionally, we use Interface classes to manage the output terminal, ensuring that the application’s output is displayed clearly to the user. As we are not allowed to display on a graphical user interface (GUI), we will implement a pseudo Model-View-ViewModel (MVVM) architecture, where the program's output is presented in the terminal rather than through a GUI.

We chose to use CSV files as our database format because it allows us to store and extract data using a simple, comma-separated structure, which aligns with our current knowledge and ensures ease of data handling and processing. Our program reads the data from the user information from the csv files to create the necessary objects needed for the program and updates the csv files to store user created data in real-time. We also made our code robust by implementing the SOLID design principles to ensure that our code has full error checking and adopts a user-friendly approach by having a simple, easy-to-read user interface flow and immediate prompts for user inputs whenever erroneous input is received.

b. Design Approach Taken

Login Process

Upon starting the HMS application, you will be greeted with the ASCII code of the word “HMS”. You will then be prompted to enter a UserID and the plain text password. The UserID is first checked against all the existing UserIDs in the Patient_List and Staff_List txt files. It will then check whether the user is logging in for the first time. The default password is “password”. Once the user changes their password, it is hashed using SHA-256 and stored in a text file. The user has to log in using his UserID and plain text password. The user must also use a strong password. The password must be at least 8 characters long and contain special characters, uppercase letters and also numbers.

Data Storage via CSV

To load and store data for the various roles, we will be utilising the “data” folder which contains all the csv files for the application which includes user and appointment details. The CSVHandler class acts as a boundary class as it can provide connections to external sources by reading and modifying CSV files from the data folder. When reading, it returns string values that correspond to the lines of the CSV file. It also allows you to add, remove or modify individual lines in a selected CSV file. Once modified, the CSV file is updated in real-time to the existing CSV file. In each CSV file, each line corresponds to the data of an individual entity, such as user or appointment. Each line is able to store all required attributes about a particular entity, such as strings, integers, and even arrays are stored by using square brackets as separators since it is crucial that the CSV file remains comma-separated about their columns.

To convert the data into entity classes, control classes also known as database managers, inherit from the CSVHandler class to transform each line returned by its functions into corresponding entity classes, and vice versa. For instance, the AppointmentManagement class reads the Appointment_Log.csv file, allowing it to return an array of Appointment classes, each containing information about individual appointments. The AppointmentHandler class, distinct from AppointmentManagement, aids in processing data from the AppointmentManagement and offers specialized methods for appointment operations. This design helps prevent class bloat and adheres to the Single Responsibility Principle.

The use of these control classes facilitates dependency injection for classes reliant on entity classes. For example, a User class might need to retrieve a list of available appointments for selection. Instead of directly creating appointment classes within the user-related functions, the list of appointments provided by AppointmentManagement can be inserted into the user methods. This approach streamlines the code, minimizes future modifications, and simplifies the workflow for developers working on the User class. This demonstrates the Open/Closed Principle (OCP) by allowing for future enhancements without modifying existing code.

User Roles: Patient, Doctor, Pharmacist and Administrator

The Patient, Doctor, Pharmacist and Administrator classes inherit from the User abstract class and they contain some similar methods and we further split into different roles to get specific methods to their respective roles. This exemplifies the Single Responsibility Principle. Since the User class has the basic attributes of each role, each role is able to inherit from User thus they can access User methods.

We used method overriding as a form of polymorphism by overriding the displayMenu method across various subclasses that inherit from the main User class. Each subclass, such as Patient, Doctor, and Administrator, is able to provide its own specific implementation of the displayMenu method which allows for tailored menu options relevant to each user role. This polymorphism enables the application to call the displayMenu method on a User reference and dynamically invoke the appropriate version of the method based on the actual object type at runtime. For example, when a Doctor instance calls displayMenu, it will show options pertinent to doctors, while a Patient instance will display options suitable for patients.

We also used method overloading through the use of the writeCSV and readCSV methods. It allows flexible handling of various CSV files. By defining multiple versions of these methods, we can tailor the input parameters to accommodate different data types and structures based on the specific CSV files being processed. For instance, the writeCSV method might be overloaded to accept parameters for writing patient data to the Patient_List.csv file as well as appointment data to the Appointment_Log.csv file, each with their unique data formats. Similarly, the readCSV method can be overloaded to read from different files such as Staff_List.csv or Medicine_List.csv, enabling the application to efficiently handle the diverse requirements of the system.

We will utilize the Singleton pattern which restricts the instantiation of a class to a single instance and provides a global point of access to that instance. This ensures that a class is instantiated only once throughout the application. By adding an instance attribute to the

class intended to be a singleton, we can create a getInstance() method. This method allows us to retrieve the single instance of the class. If it is the first time accessing it, the method will initialize the class; otherwise, it will return the already created instance. Consequently, in the future, when we need to access the Patient and Staff, we do not need to reinitialize them, thus maintaining consistency and preventing errors. This is directly related to the Single Responsibility Principle.

Creating Entities

To be able to add entities into the CSV database, such as appointments, medicine and inventory, a method is invoked within their respective management packages to create a new entity object. This method converts the entity into a CSV line format and appends that line to the corresponding CSV file, enabling the storage and retrieval of entire entities. Once an entity is created in the database, it contains a unique identifier, such as AppointmentID or MedicineID, so that we are able to reference them in the future.

They also contain the UserID of the person who made the entity, such as the staff member who is managing appointments. It is saved in the Appointment_Log CSV for appointments. This helps in identifying and tracking the entities created by different users. It is worth noting that all data managers inherit from the CSVHandler class, since they require the reading and writing of CSV files. This design adheres to the Open-Closed Principle, because if we wanted to create a new database in the future, we can just extend from the CSVHandler again to access database editing capabilities, without the need to modify the inner workings of the CSVHandler.

Additional Features

1) We have implemented a Medical Certificate (MC) system. Patients would be able to request for a MC only when they have consulted a doctor and the doctor must've given a diagnosis. Patients would also be able to view the MC that has been approved or rejected by any doctor, the MC would also reflect which doctor has approved/rejected their request. The Doctor is able to view all Pending/Approved/Rejected MCs. They would have an option to

approve/reject a patient's requests for MC, which includes reason for MC and the number of days requested to name a few.

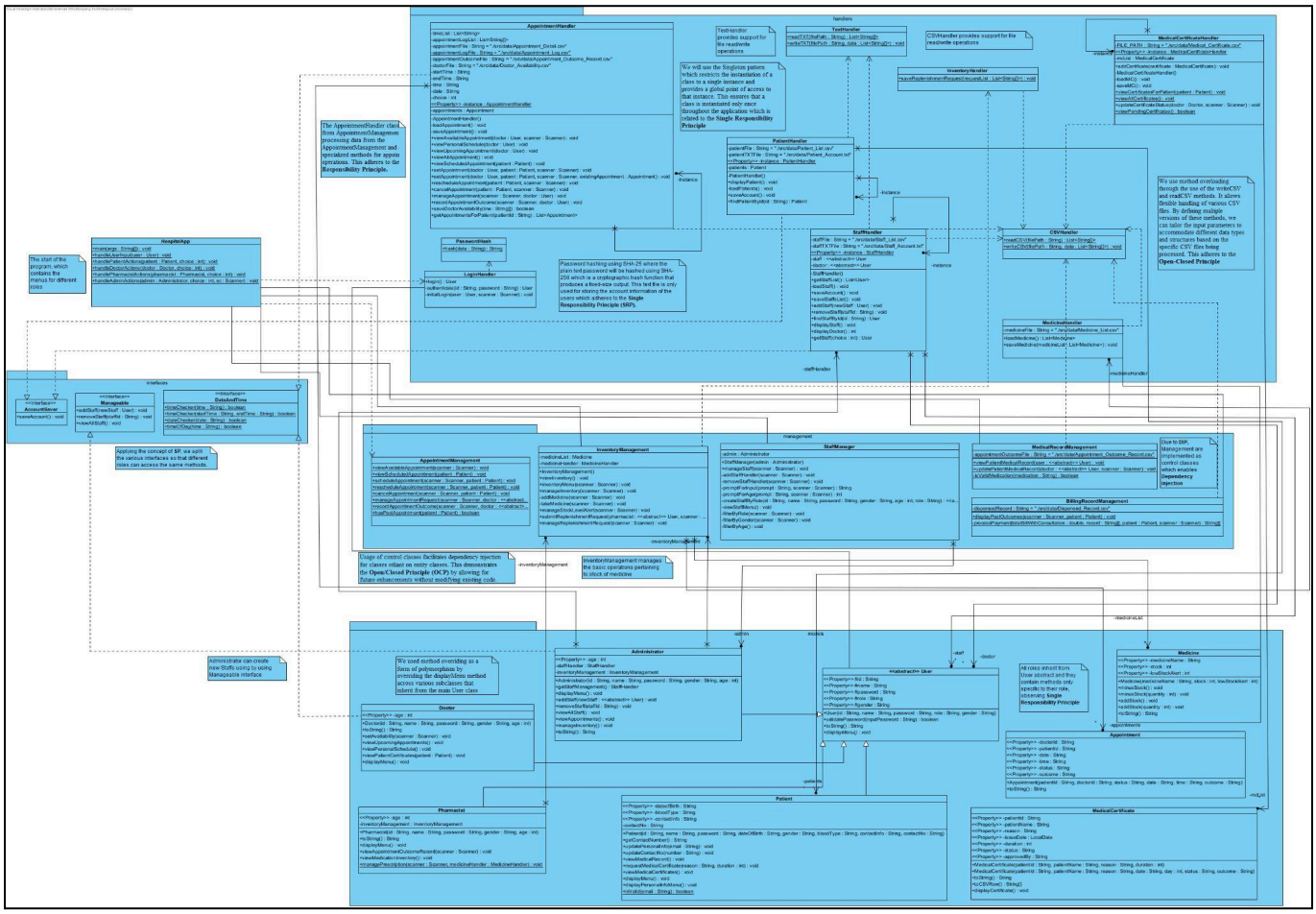
2) We implemented a password hashing using SHA-256. The plain text password will be hashed using SHA-256 which is a cryptographic hash function that produces a fixed-size output. This hashed password is compared with the stored passwords in the Patient_Account or Staff_List text file which ensures that passwords are safely secured in the text file. This text file is only used for storing the account information of the users which adheres to the Single Responsibility Principle (SRP). We also implemented password complexity where users have to have a "Strong" password which is of at least 8 characters, containing special characters, uppercase letters and also numbers.

3) We implemented a billing report system which also includes payment. Patients can view their billing records and have an overview of the breakdown of each record billed to them (medicine cost, consultation cost). They can also choose to pay the bill if the bill states that it is unpaid. The bills will only be issued to the patient once the pharmacist has updated the prescription and disbursed medicine to the patient.

Assumptions Made

- There will not be multiple users logged in at the same time using the application as it may lead to data inconsistency from the delay in changes within the database.
- The data should always be available and is never corrupted to ensure readable data.
- Database is securely protected in a safe location as it contains sensitive information about users in plaintext for the CSV files.
- The doctors will work 9am to 6pm every day unless they change their schedule and the clinic is open 24/7.

2. Detailed UML Class Diagram (image is also on our github)



3. Testing:

1) Login

Invalid user:

```
Enter User ID: P1001
Enter Password: 1234
Login failed. Invalid credentials.
Enter User ID:
```

Invalid password:

```
Enter User ID: D004
Enter Password: www
Login failed. Invalid credentials.
Enter User ID:
```

Users are asked to change their password when they login for the first time:

```
Enter User ID: P1001
Enter Password: password
This is your first time logging in so you need to change your password
----Change Password----
Enter your new password: 1
Enter your new password again: 1
You have changed your password successfully!
Enter User ID: P1001
Enter Password: 1
Login successful! Welcome, Alice Brown (Patient)
```

EXTRA FEATURE: In the text file, users password is hashed with SHA-256.

```
1 P1001,6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b
2 P1002,6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b
3 P1003,5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8
```

EXTRA FEATURE: Password Complexity, minimum of 8 characters including special characters, numbers and uppercase.

```
----Change Password----
Enter your new password: 123
Enter your new password again: 123
The password does not meet the requirement.
[x] More than 8 characters
[x] At least one upper character
[x] At least one special character (!, ", #, $, %, &, ', (, ), *, @)
[/] At least one number
----Change Password----
Enter your new password: Password1@
Enter your new password again: Password1@
You have changed your password successfully!
```

User cannot change the password to the default password.

```
Enter User ID: A001
Enter Password: password
This is your first time logging in so you need to change your password
----Change Password----
Enter your new password: password
Enter your new password again: password
Cannot be the default password.
----Change Password----
Enter your new password:
```

Patient:

```
Patient Menu:
1. View Medical Record
2. Update Personal Information
3. View Available Appointment Slots
4. Schedule Appointment
5. Reschedule Appointment
6. Cancel Appointment
7. View Scheduled Appointments
8. View Past Appointment Outcome Records
9. Request Medical Certificate
10. View Medical Certificates
11. View Billing Records
12. Logout
```

Doctor:

```
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. View Patient Medical Certificates
9. Approve/Reject Medical Certificates
10. Logout
```

Administrator:

```
Administrator Menu:
1. Manage Hospital Staff
2. View Appointments
3. View and Manage Medication Inventory
4. Approve Replenishment Requests
5. Logout
```

Pharmacist:

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

2) Patient

(1)View Medical Record

```
Medical Record for Patient ID: P1001
Name: Alice Brown
Date of Birth: 1980-05-14
Gender: Female
Blood Type: A+
Contact Info: alice.brown@example.com
Contact Number: -
```

(2)Update Personal Information:

```
Medical Record for Patient ID: P1001
Name: Alice Brown
Date of Birth: 1980-05-14
Gender: Female
Blood Type: A+
Contact Info: aliceinwonderland@gmail.com
Contact Number: 99998888
```

(3)View available appointment

```
Enter option: 3
----Available Doctors----
1. John Smith
2. Emily Clarke
Enter the doctor (0 to exit): 1
Enter the date (DD/MM/YYYY): 07/11/2024
1. 09:00        2. 10:00
3. 11:00        4. 12:00
5. 13:00        6. 14:00
7. 15:00        8. 16:00
9. 17:00
```

(4)Schedule an Appointment

```
Enter option: 4
----Available Doctors----
1. John Smith
2. Emily Clarke
Enter the doctor (0 to exit): 1
Enter the date (DD/MM/YYYY): 12/12/2024
1. 09:00        2. 10:00
3. 11:00        4. 12:00
5. 13:00        6. 14:00
7. 15:00        8. 16:00
9. 17:00

Enter the slot (0 to exit): 1
Scheduled appointment successful.
```

(5) View Reschedule & Scheduled Appointments

```
Enter the appointment you want to reschedule (0 to exit): 1
----Available Doctors----
1. John Smith
2. Emily Clarke
Enter the doctor (0 to exit): 1
Enter the date (DD/MM/YYYY): 07/11/2024
1. 11:00        2. 12:00
3. 13:00        4. 14:00
5. 15:00        6. 16:00
7. 17:00

Enter the slot (0 to exit): 4
```

(6) Cancel Appointment

```
Enter option: 6
----Existing Appointment----
1. Emily Clarke 07/11/2024 10:00 (Confirmed)
2. John Smith 07/11/2024 14:00 (Confirmed)

Enter the appointment you want to cancel (0 to exit): 2
You have cancelled the appointment.
```

(7)View Scheduled Appointments

```
Enter option: 7

----Scheduled Appointments----
P1001,D002,Confirmed,07/11/2024,10:00,-
```

(8)View Past Appointment Outcome Records

```
11. View Billing Records
12. Logout
Enter option: 8
----Patient Medical Records----
Doctor  Patient         Date            Type of Service
tion Status     Consultation Notes
D001    P1001           08/11/2024      Consultation
ending                  Fever
```

EXTRA FEATURE (Medical Certificate)

(9)Request Medical Certificate

```
Enter option: 8
Enter reason for medical certificate: fever
Enter duration (in days): 3
Medical certificate requested successfully.
```

(10)View Medical Certificates

```
---- Medical Certificate ----
Patient ID: P1001
Patient Name: Alice Brown
Reason: fever
Issue Date: 2024-11-02
Duration: 3 days
Status: Pending
-----------------------------
```

```
Enter option: 9
---- Medical Certificate -----
Patient ID: P1001
Patient Name: Alice Brown
Reason: fever
Issue Date: 2024-11-02
Duration: 3 days
Status: Approve
-----------------------------
```

EXTRA FEATURE (Billing)

(11) View for Billing Records if all paid

```
Enter option: 10
Displaying past appointment outcomes for Patient ID: P1001
============================================================
Appointment on: 23/10/24 with the type of service: Mild Flu
Payment Status: Paid
Medicine and Quantity List:
Medicine: Paracetamol, Quantity: 2
Total Medicine Bill Cost: $4.00
Total Bill Cost with Consultation Fee: $24.00
============================================================
```

(11) View for bills unpaid

```
============================================================
2. Appointment on: 08/11/2024 with the type of service: mild flu
Payment Status: Unpaid
Medicine and Quantity List:
Medicine: Paracetamol, Quantity: 10
Total Medicine Bill Cost: $20.00
Total Bill Cost with Consultation Fee: $40.00
------------------------------------------------------------
Choose an appointment to manage (0 to exit):
```

(11) Successful payment of bill and change

```
------------------------------------------------------------
Choose an appointment to manage (0 to exit): 1
============================================================
Appointment on: 08/11/2024 with the type of service: mild flu
Payment Status: Unpaid
Medicine and Quantity List:
Medicine: Paracetamol, Quantity: 10
Total Medicine Bill Cost: $20.00
Total Bill Cost with Consultation Fee: $40.00
Your total bill is $40.00.
Dear Alice Brown, You have an outstanding bill of $40.0
Please enter the amount you wish to pay or type '0' if you do not wish to pay at this moment: 42
Processing payment...
Payment of $42.00 has been successfully completed. Your change is $2.00.
```

(11) Unsuccessful payment

```
Your total bill is $40.00.
Dear Alice Brown, You have an outstanding bill of $40.0
Please enter the amount you wish to pay or type '0' if you do not wish to pay at this moment: 20
Invalid payment amount. Please enter a valid amount.
```

3) Doctor

(1) View Patient Medical Records

```
Enter option: 1
----Patient Medical Records----
Doctor  Patient         Date            Type of Service         Prescription Status     Consultation Notes
D001    P1001           8/11/2024       Consultation                    Dispensed                       Fever
```

(2) Update Patient Medical Records

```
Enter option: 2
----Patient Medical Records----
1. [D001, P1001, 08/11/2024, Consultation, Dispensed, Fever]

Choose a record to update (0 to exit): 1
New Blood Type (blank if no changes): X
Type of service provided: Consultation
Consultation Notes: X blood type found
Update successfully.
Doctor Menu:
1. View Patient Medical Records
```

(3) View Personal Schedule

```
Enter option: 3
----John Smith's Schedule----
1. P1001,D001,Confirmed,08/11/2024,09:00,Refer to Record
2. P1001,D001,Confirmed,08/11/2024,10:00,Refer to Record
3. P1001,D001,Confirmed,08/11/2024,11:00,Refer to Record
```

(4) Set Availability for Appointments

```
Enter option: 4
Enter the date (DD/MM/YYYY): 08/11/2024
Enter the start time (00:00 - 23:59): 09:00
Enter the end time (09:01 - 23:59): 12:00
Doctor's availability set for 08/11/2024 from 09:00 to 12:00
```

(5) Accept/Decline Appointment

```
Enter option: 5
----John Smith's Schedule----
1. P1001,D001,Pending,03/11/2024,09:00,-

Choose the appointment you want to manage (0 to exit): 1
Accept or Decline (0 to exit): Accept
```

(6)View Upcoming Appointment

```
----John Smith's Schedule----
1. P1001,D001,Confirmed,03/11/2024,09:00,-
```

(7) Record Appointment Outcome

```
Enter option: 7
----John Smith's Confirmed Schedule----
1. P1001,D001,Confirmed,08/11/2024,10:00,-

Choose the appointment you want to record (0 to exit): 1
Type of service provided: Mild Flu
Consultation Notes: Mild flu and fever, to prescribe paracetamol
Record successfully.
```

EXTRA FEATURES

(8) View Patient Medical Certificates

```
Enter option: 8
---- All Medical Certificate Requests ----
Patient ID: P1001, Name: Alice Brown, Reason: Fever, Duration: 3 days, Status: Pending, Approved/Rejected By: N/A
```

```
Enter option: 8
---- All Medical Certificate Requests ----
Patient ID: P1001, Name: Alice Brown, Reason: Fever, Duration: 3 days, Status: Approved, Approved/Rejected By: D001
```

```
Enter option: 8
---- All Medical Certificate Requests ----
No pending medical certificates found.
```

(9)Approve/Reject Medical Certificate

```
Enter option: 9
---- Pending Medical Certificate Requests ----
1. P1001 Alice Brown Fever 3
Enter your choice (0 to exit): 1
Enter new status (Approved/Rejected): Approved
```

4) Pharmacists

(1) View All Appointment Outcome Records

```
Pharmacist Menu:
1. View Appointment Outcome Record
2. Update Prescription Status
3. View Medication Inventory
4. Submit Replenishment Request
5. Logout
Enter option: 1
----Appointment Outcome Record(s)----
[D001, P1001, 08/11/2024, mild flu, Pending, runny nose and fever,  to administer paracetamol]
```

(2) Update Prescription Status

```
Choose a record to update (0 to exit): 1
----Medications----
1. Paracetamol
2. Ibuprofen
3. Amoxicillin
```

(3) View Medication Inventory

```
Pharmacist Menu:
1. View Appointment Outcome Record
2. Update Prescription Status
3. View Medication Inventory
4. Submit Replenishment Request
5. Logout
Enter option: 3
---- Medicine Inventory ----
Amoxicillin: 109
Paracetamol: 80
Ibuprofen: 0 (Low Stock Level)
```

(4) Submit Medication Request

```
Pharmacist Menu:
1. View Appointment Outcome Record
2. Update Prescription Status
3. View Medication Inventory
4. Submit Replenishment Request
5. Logout
Enter option: 4
----Low Stock Level Medicine----
1. Ibuprofen
0. Exit

Enter the medicine no. you want to submit a request for: 1
Amount to request: 90
Request submitted successfully.
```

5) Administrator

(1)View and Manage Hospital Staff

```
Administrator Menu:
1. Manage Hospital Staff
2. View Appointments
3. View and Manage Medication Inventory
4. Approve Replenishment Requests
5. Logout
Enter option: 1
Manage Hospital Staff:
1. Add Staff
2. Remove Staff
3. View Staff
4. Back to Main Menu
Enter your choice: 1
Enter staff ID: D009
Enter staff name: Weiyu
Enter staff gender: Male
Enter staff age: 24
Enter staff role (Doctor, Pharmacist, Administrator): Doctor
New staff added: Weiyu (Doctor)
```

(1)View with filter & Remove Staff

```
3. View Staff
4. Back to Main Menu
Enter your choice: 3
----Filter Staff----
1. By Role
2. By Gender
3. By Age

Enter your choice (0 to exit): 1
----Role(s)----
1. Doctor
2. Pharmacist
3. Administrator
0. Exit

Choose an option: 1
D001 John Smith Doctor Male 45
D002 Emily Clarke Doctor Female 38
D009 Weiyu Doctor Male 24
```

```
Manage Hospital Staff:
1. Add Staff
2. Remove Staff
3. View Staff
4. Back to Main Menu
Enter your choice: 2
Enter staff ID to remove: D009
```

(2)View Appointment details

```
Enter option: 2

----All Appointments----
No appointments.
```

(2)View Appointment details(With appointments:)

```
1. Manage Hospital Staff
2. View Appointments
3. View and Manage Medication Inventory
4. Approve Replenishment Requests
5. Logout
Enter option: 2

----All Appointments----
P1001,D001,Confirmed,08/11/2024,09:00,Refer to Record
P1001,D001,Confirmed,08/11/2024,10:00,Refer to Record
```

(3)View and manage medicine inventory

```
1. View Inventory of Medicine
2. Manage Inventory of Medicine
3. Update Medicine Stock Level Alert
0. Exit
Enter your choice: 1
---- Medicine Inventory ----
Amoxicillin: 109
Paracetamol: 80
Ibuprofen: 0 (Low Stock Level)
```

(3)(Adding and removing 5 units of Amoxicillin):

```
1. View Inventory of Medicine
2. Manage Inventory of Medicine
3. Update Medicine Stock Level Alert
0. Exit
Enter your choice: 2
---- Medicine Inventory ----
Amoxicillin: 109
Paracetamol: 80
Ibuprofen: 0 (Low Stock Level)
1. Add Medicine
2. Take Medicine
0. Exit
Enter your choice: 1
---- Add Medicine ----
Enter medicine name: Amoxicillin
Enter quantity to add: 5
5 units of Amoxicillin added.
---- Medicine Inventory ----
```

```
---- Medicine Inventory ----
Amoxicillin: 114
Paracetamol: 80
Ibuprofen: 0 (Low Stock Level)
1. Add Medicine
2. Take Medicine
0. Exit
Enter your choice: 2
---- Take Medicine ----
Enter medicine name: Amoxicillin
Enter quantity to take: 5
5 units of Amoxicillin taken.
---- Medicine Inventory ----
Amoxicillin: 109
Paracetamol: 80
Ibuprofen: 0 (Low Stock Level)
1. Add Medicine
2. Take Medicine
0. Exit
```

(4)Approve Replenishment Requests

```
Enter option: 4
----Request List----
1. [P001, Ibuprofen, 90]
2. [P001, Ibuprofen, 50]
0. Exit

Choose a request to manage: 1
1. Approve
2. Reject
0. Exit

Enter a choice: 1
[Paracetamol: 60, Ibuprofen: 90, Amoxicillin: 75]
Request approved and stock updated.
```

(4) Approve Replenishment Requests (Rejected)

```
Enter option: 4
----Request List----
1. [P001, Ibuprofen, 50]
0. Exit

Choose a request to manage: 1
1. Approve
2. Reject
0. Exit

Enter a choice: 2
Request rejected.
```

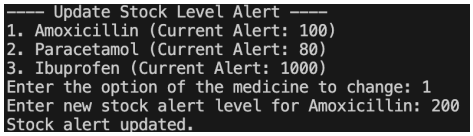
(4)Current Alert Levels:

```
Enter your choice: 3
---- Update Stock Level Alert ----
1. Amoxicillin (Current Alert: 100)
2. Paracetamol (Current Alert: 80)
3. Ibuprofen (Current Alert: 1000)
```

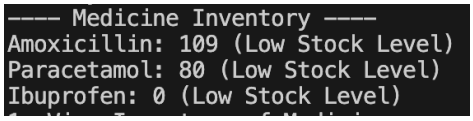
(4)Current Inventory Display:

```
---- Medicine Inventory ----
Amoxicillin: 109
Paracetamol: 80 (Low Stock Level)
Ibuprofen: 0 (Low Stock Level)
```

(4)Adjusted alert level for Amoxicillin to 200:



(4)Adjusted Inventory Display:



4. Reflection

a. Difficulties Faced

Upon reading through the requirements of the project, we realised that we were unable to use any database application to store our data and information. After reviewing the provided files, we realized we could leverage Excel/CSV files as a makeshift database to store our information.

As we worked to implement and optimize our code, we struggled to apply concepts we learnt in lectures, particularly around object-oriented design and the SOLID principles. The UML diagram, which was derived from our code, required multiple iterations as we sought to align our implementation with these principles.

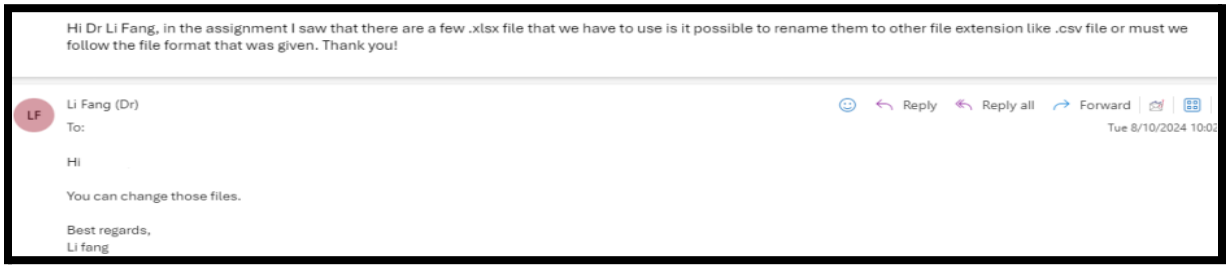
Through this learning process, we recognised that our initial code structure was difficult to read and prone to breaking whenever we attempted to add new features. This experience has highlighted the importance of the concepts that were taught to us in the lectures which ultimately helps us to become better programmers. We learned to appreciate the value of clean, scalable code and the need to adhere to design principles to create robust software.

b. Knowledge learnt from this course

From the lectures, we had learnt about the object-oriented (OO) concepts alongside the SOLID design principles. We tried to ensure that any necessary changes to our project could be implemented smoothly with minimal impact. We also learnt to manage the implementation of a relatively large project given a short timeframe by learning the importance of divide and conquer techniques so that we are able to do the tasks efficiently. Additionally, we took the initiative to self-learn and utilize libraries and built-in functions, such as java.time and CSV read/write methods using java.io which enhances our application's functionality and robustness.

c. Area of improvements for the course

There were project specifications and descriptions which were vague and unclear. This made us confused. We had to come up with our own assumptions and interpretations as information given was vague. Initially, the given excel files were in xlsm format which were incompatible with the java.io library. We had to email the professor to ensure that we did not go against any restrictions that were pre-placed for the project. The suggested sample test cases also did not align to the features required of the application. The report requirements were also misaligned with the appendix. The first figure shows our email to the professor. The second figure does not mention the "Testing" section to be included while the figure below mentions adding test cases.



Hi Dr Li Fang, in the assignment I saw that there are a few .xlsx file that we have to use is it possible to rename them to other file extension like .csv file or must we follow the file format that was given. Thank you!

LF Li Fang (Dr)

To:

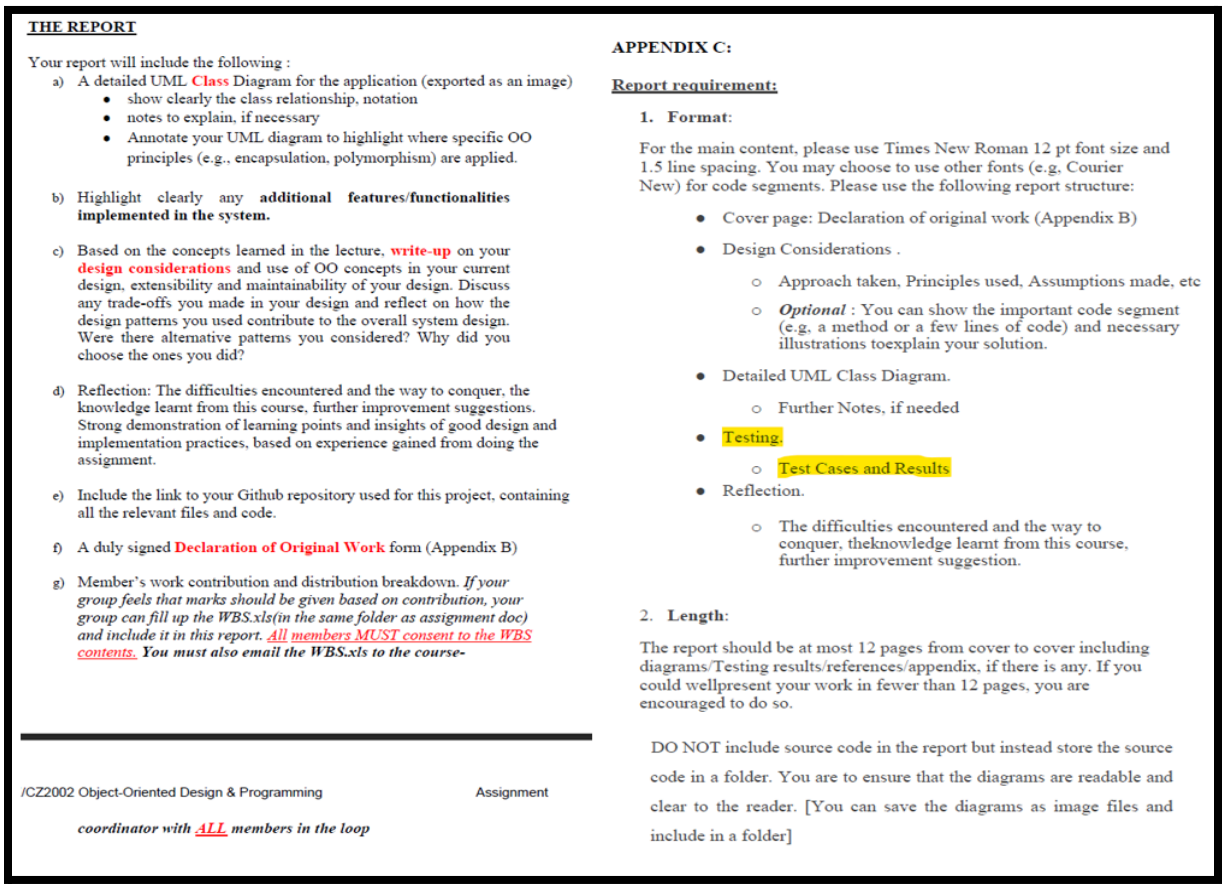
Reply Reply all Forward

Tue 8/10/2024 10:0

Hi

You can change those files.

Best regards,
Li fang



THE REPORT

Your report will include the following :

a) A detailed UML **Class** Diagram for the application (exported as an image)
- show clearly the class relationship, notation
- notes to explain, if necessary
- Annotate your UML diagram to highlight where specific OO principles (e.g., encapsulation, polymorphism) are applied.

b) Highlight clearly any **additional features/functionalities implemented in the system.**

c) Based on the concepts learned in the lecture, **write-up** on your **design considerations** and use of OO concepts in your current design, extensibility and maintainability of your design. Discuss any trade-offs you made in your design and reflect on how the design patterns you used contribute to the overall system design. Were there alternative patterns you considered? Why did you choose the ones you did?

d) Reflection: The difficulties encountered and the way to conquer, the knowledge learnt from this course, further improvement suggestions. Strong demonstration of learning points and insights of good design and implementation practices, based on experience gained from doing the assignment.

e) Include the link to your Github repository used for this project, containing all the relevant files and code.

f) A duly signed **Declaration of Original Work** form (Appendix B)

g) Member's work contribution and distribution breakdown. *If your group feels that marks should be given based on contribution, your group can fill up the WBS.xls(in the same folder as assignment doc) and include it in this report. All members MUST consent to the WBS contents.* ***You must also email the WBS.xls to the course-***

/CZ2002 Object-Oriented Design & Programming Assignment

coordinator with ALL members in the loop

APPENDIX C:

Report requirement:

1. **Format**:

For the main content, please use Times New Roman 12 pt font size and 1.5 line spacing. You may choose to use other fonts (e.g. Courier New) for code segments. Please use the following report structure:

- Cover page: Declaration of original work (Appendix B)
- Design Considerations .
  - Approach taken, Principles used, Assumptions made, etc
  - ***Optional*** : You can show the important code segment (e.g. a method or a few lines of code) and necessary illustrations toexplain your solution.
- Detailed UML Class Diagram.
  - Further Notes, if needed
- Testing.
  - Test Cases and Results
- Reflection.
  - The difficulties encountered and the way to conquer, theknowledge learnt from this course, further improvement suggestion.

2. **Length**:

The report should be at most 12 pages from cover to cover including diagrams/Testing results/references/appendix, if there is any. If you could wellpresent your work in fewer than 12 pages, you are encouraged to do so.

DO NOT include source code in the report but instead store the source code in a folder. You are to ensure that the diagrams are readable and clear to the reader. [You can save the diagrams as image files and include in a folder]