Holy Angel University
School of Computing
Department

Subscription Management System
1st Semester, S.Y. 2024-2025

CASE STUDY: Welfare Records System

Submitted by:

Group 3

ONG, Edward

SESE, Jacob Mark

TAKUSHI, Brannon

CYB-201

Submitted to:

JOHN REY D CASINGAL

October 25, 2024

Manuscript

The healthcare management system starts with the Main class, asking the user to choose an option from a menu ranging from 1 to 7. It invokes methods from the Controller class depending on the chosen option, encapsulating core functionalities such as scheduling, updating, canceling, searching, filtering, and displaying appointments. The Controller controls the user input and coordinates the operations related to the Manager class in terms of storing and manipulating appointment data.

The Manager class contains a list of appointments and medical logs. It holds information regarding an appointment using the Appointment class, which enables the user to create a new appointment, update an existing one, or cancel it based on requirements. The filter method fetches appointments based on status, and another method known as search allows the user to find a specific appointment by their unique identifiers.

We had a class called Validation that validated the inputs entered by the users and thus prevented mistaken entries, making it more robust. We also have a central location named Displayer to put our print statements inside so that any change required for the output alone will be taken care of without affecting the core logic of the Controller.

In respect of object-oriented programming principles, we emphasize encapsulation in our base User class and all its derived classes, that is, doctor, nurse, and patient, for private attributes. The above ensures integrity in user data while still allowing public access methods. Our design embodies the Inheritance principle, whereby child classes inherit properties from a parent class, in our case the User class, and encourages coherent structure and code reuse.

Several interfaces namely, ISchedule, IUpdate, ICancel, ISearch, IFilter, and IDisplay specify the core functionality of the Controller class, encouraging Polymorphism through method overriding for the variety of user actions to be handled.

The Aggregation implementation is demonstrated by the Manager class, which depends on the input provided by the Controller and user classes. Such allows for effective management of appointments and medical logs. Each class can be designed with Cohesion in mind; for instance, an error-checking task is strictly managed by the Validation class while the Displayer deals only with the outputting of information. Such separation of responsibilities goes a long way in enhancing both readability and maintainability.

Finally, we balance between Tight Coupling and Loose Coupling. The Controller tightly couples with Manager and user classes and manages their interactions keenly. This is in direct conflict with the Main class, which is not tightly coupled with these components; therefore, it has minimal effects if changes occur. The complexities described above are well dealt with in our system by the design principles discussed above, enhancing the overall efficiency of operations and better care given to a patient.

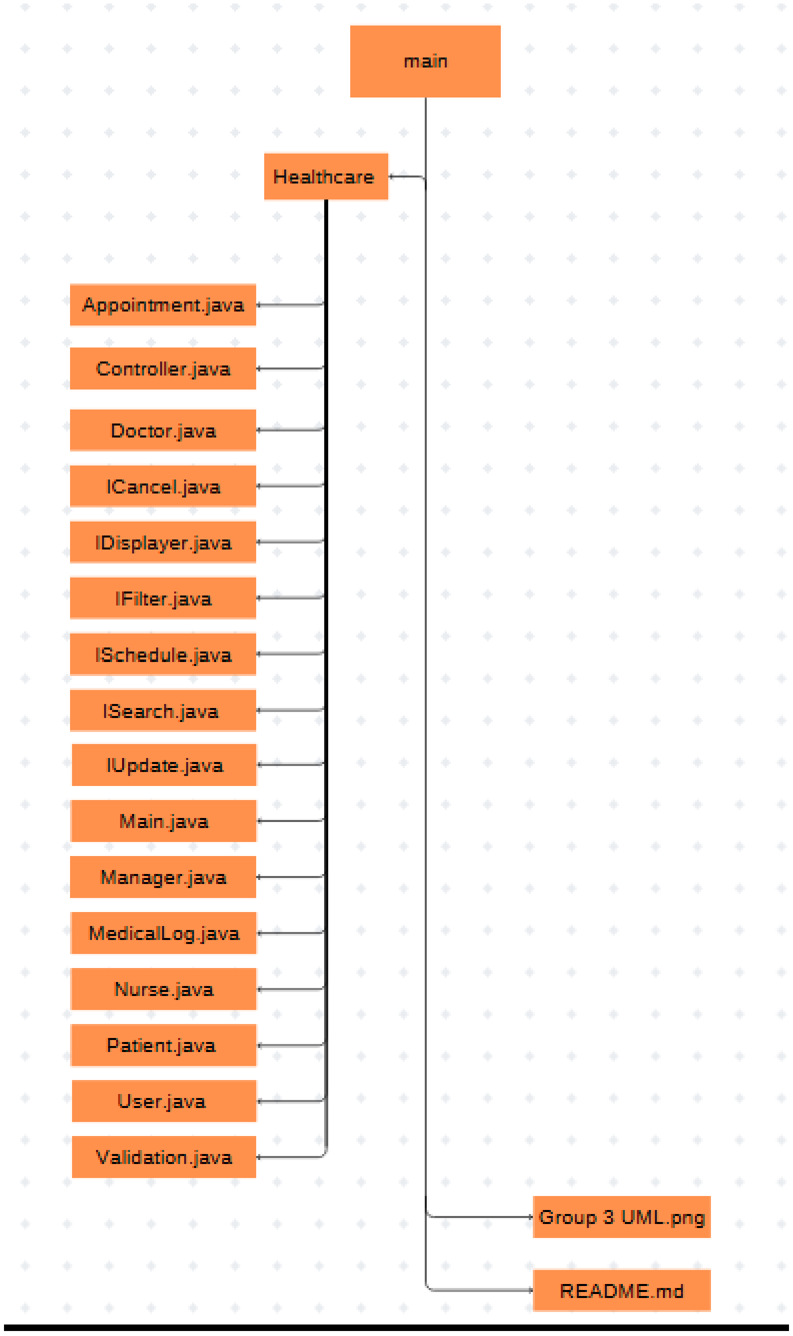
Project Description

It is a console-based healthcare management application that allows users to schedule, update, or cancel appointments besides searching and filtering appointment records. This system holds various healthcare professionals such as doctors and nurses handling multiple patient information and medical logs. For instance, every class, such as Doctor, Nurse, and Patient, has specific attributes that determine their functions within the healthcare system. This is built with encapsulation, inheritance, and polymorphism from object-oriented principles such that there would be very high modularity and maintainability of the codebase. It will, therefore, have the capacity to orchestrate user interactions such as managing appointments through method calls, and will use a manager class for efficient storage and retrieval of the data.

Scope and Limitations

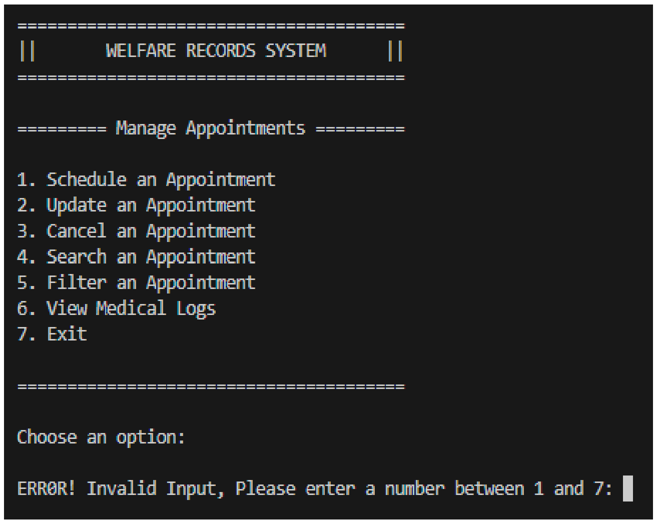
A console-based application to manage health appointments has been successfully tested using IntelliJ and Visual Studio Code. The core functionalities of the product include scheduling, updating, canceling, searching, and filtering the status of appointments. The role-based application is created for three types of users: doctors, nurses, and patients. The interface is text-based only, it totally consoles print statements, which ensures a clear layout. Although this makes implementation easier, it constrains the system's visual appeal, as well as interactivity among other aspects. Persistent storage of data is not part of the program; all information exists only in memory, so it loses data when the application is closed. For this reason, it is not sufficient for long use. Which makes it hard to use for long term

Code-base Structure

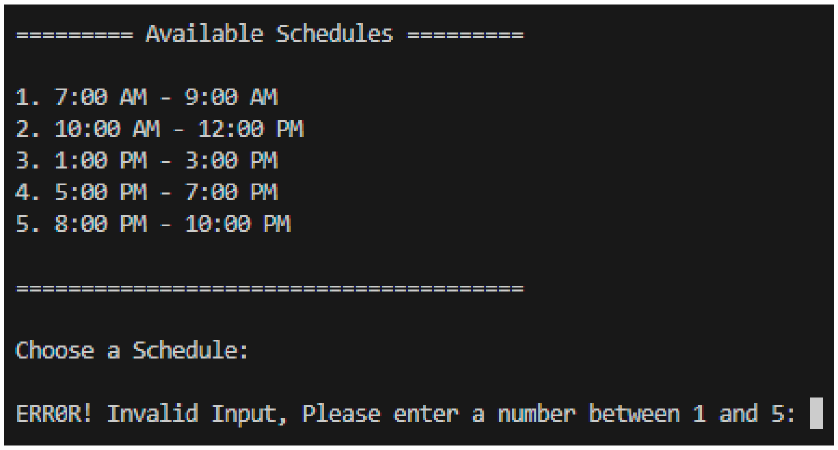


Appendices

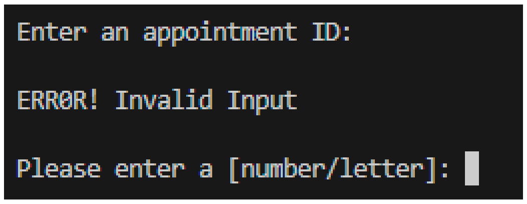
Error Handling and Validation of Inputs:



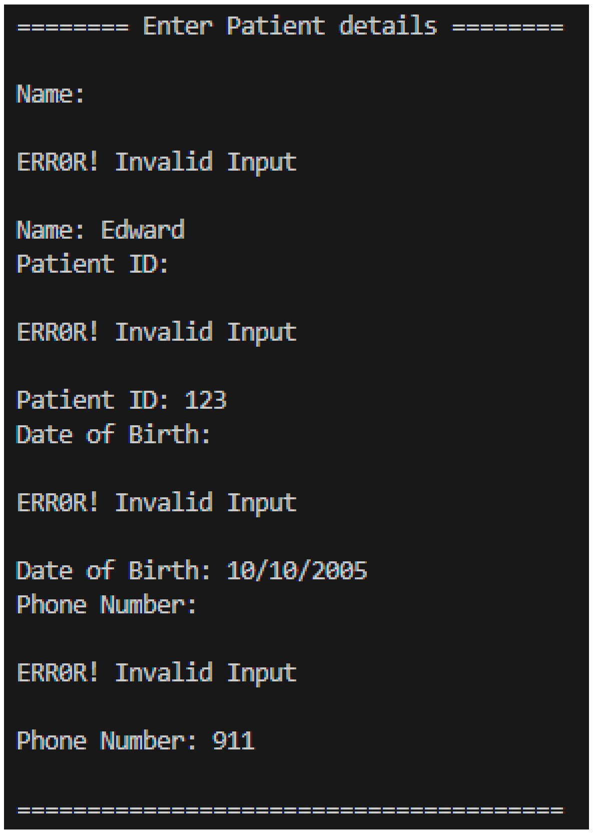
```
=======================================
||       WELFARE RECORDS SYSTEM      ||
=======================================

========= Manage Appointments =========

1. Schedule an Appointment
2. Update an Appointment
3. Cancel an Appointment
4. Search an Appointment
5. Filter an Appointment
6. View Medical Logs
7. Exit

=======================================

Choose an option:

ERR0R! Invalid Input, Please enter a number between 1 and 7:
```



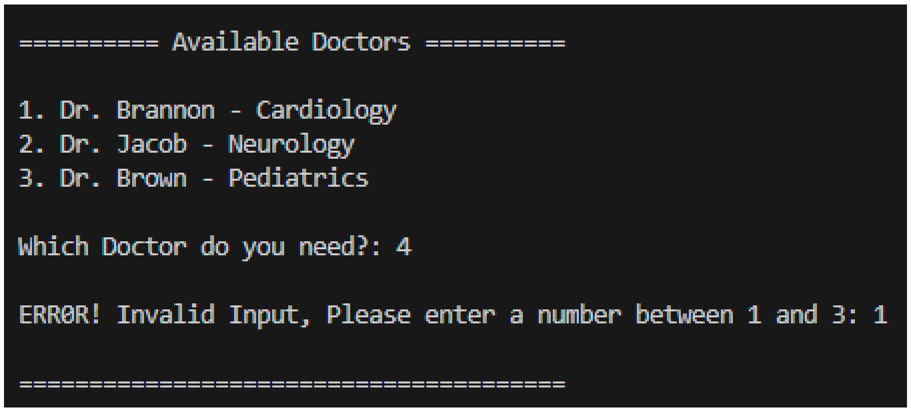
```
========= Available Schedules =========

1. 7:00 AM - 9:00 AM
2. 10:00 AM - 12:00 PM
3. 1:00 PM - 3:00 PM
4. 5:00 PM - 7:00 PM
5. 8:00 PM - 10:00 PM

=======================================

Choose a Schedule:

ERR0R! Invalid Input, Please enter a number between 1 and 5:
```



```
Enter an appointment ID:

ERR0R! Invalid Input

Please enter a [number/letter]:
```

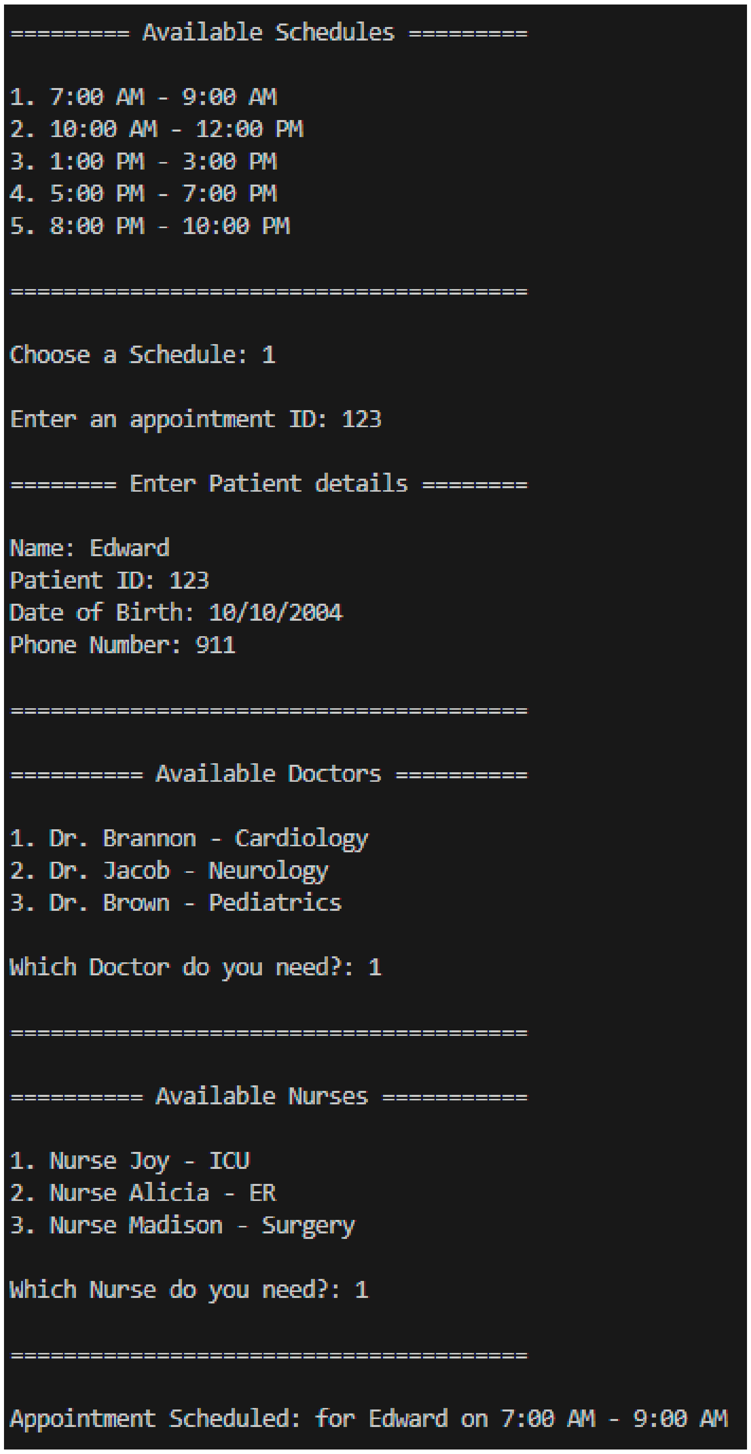


```
======== Enter Patient details ========

Name:

ERR0R! Invalid Input

Name: Edward
Patient ID:

ERR0R! Invalid Input

Patient ID: 123
Date of Birth:

ERR0R! Invalid Input

Date of Birth: 10/10/2005
Phone Number:

ERR0R! Invalid Input

Phone Number: 911

=======================================
```



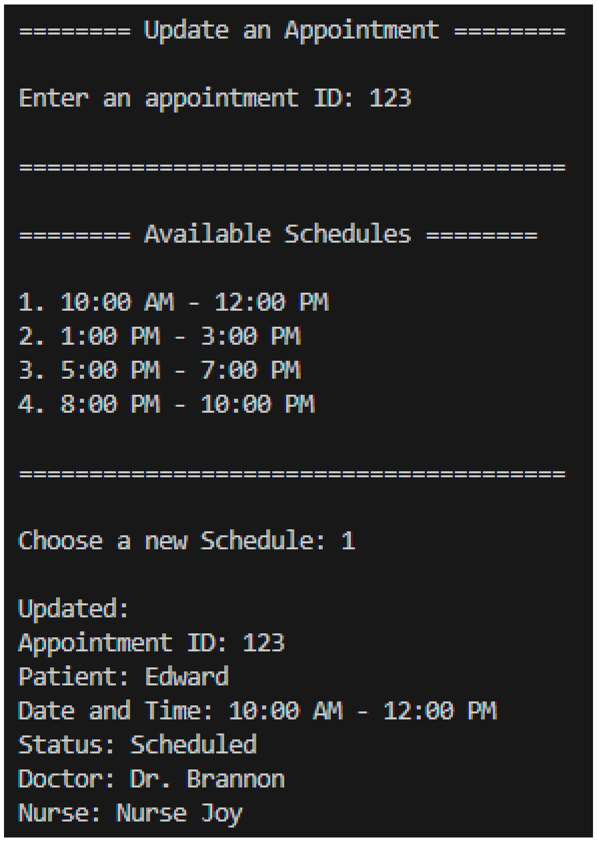
```
========== Available Doctors ==========

1. Dr. Brannon - Cardiology
2. Dr. Jacob - Neurology
3. Dr. Brown - Pediatrics

Which Doctor do you need?: 4

ERR0R! Invalid Input, Please enter a number between 1 and 3: 1

=======================================
```

Schedule Feature:



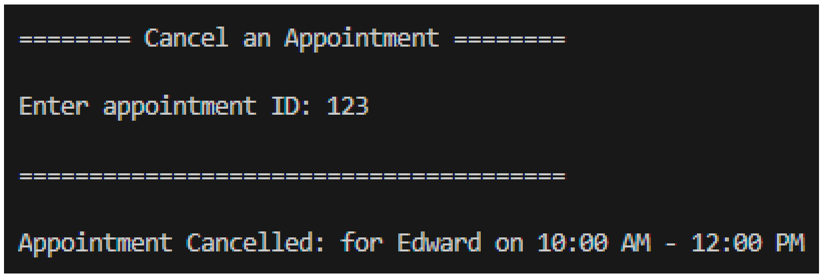
```
========= Available Schedules =========

1. 7:00 AM - 9:00 AM
2. 10:00 AM - 12:00 PM
3. 1:00 PM - 3:00 PM
4. 5:00 PM - 7:00 PM
5. 8:00 PM - 10:00 PM

=======================================

Choose a Schedule: 1

Enter an appointment ID: 123

======== Enter Patient details ========

Name: Edward
Patient ID: 123
Date of Birth: 10/10/2004
Phone Number: 911

=======================================

========== Available Doctors ==========

1. Dr. Brannon - Cardiology
2. Dr. Jacob - Neurology
3. Dr. Brown - Pediatrics

Which Doctor do you need?: 1

=======================================

========== Available Nurses ===========

1. Nurse Joy - ICU
2. Nurse Alicia - ER
3. Nurse Madison - Surgery

Which Nurse do you need?: 1

=======================================

Appointment Scheduled: for Edward on 7:00 AM - 9:00 AM
```

Update Feature:



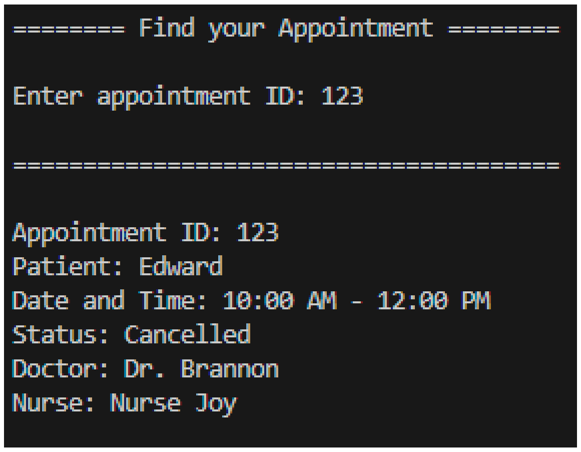
```
======== Update an Appointment ========

Enter an appointment ID: 123

=======================================

======== Available Schedules ========

1. 10:00 AM - 12:00 PM
2. 1:00 PM - 3:00 PM
3. 5:00 PM - 7:00 PM
4. 8:00 PM - 10:00 PM

=======================================

Choose a new Schedule: 1

Updated:
Appointment ID: 123
Patient: Edward
Date and Time: 10:00 AM - 12:00 PM
Status: Scheduled
Doctor: Dr. Brannon
Nurse: Nurse Joy
```

Cancel Feature:



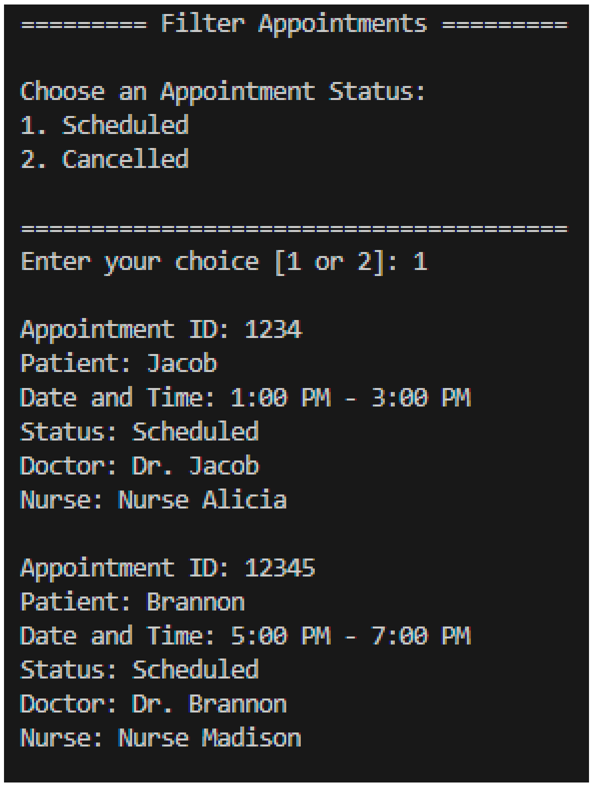
```
======== Cancel an Appointment ========

Enter appointment ID: 123

=======================================

Appointment Cancelled: for Edward on 10:00 AM - 12:00 PM
```

Search Feature:

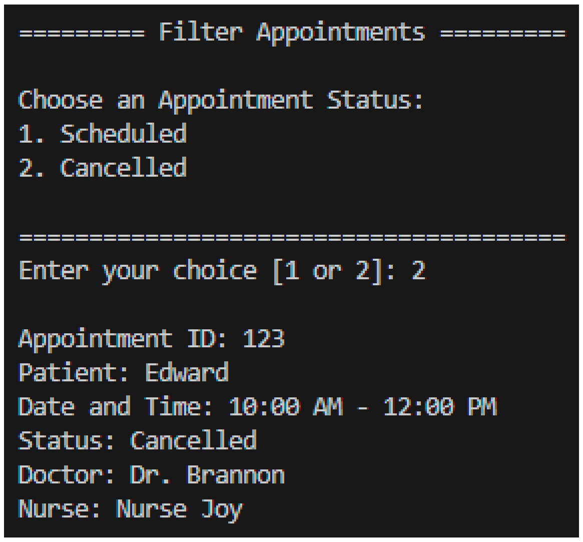


```
======== Find your Appointment ========

Enter appointment ID: 123

=======================================

Appointment ID: 123
Patient: Edward
Date and Time: 10:00 AM - 12:00 PM
Status: Cancelled
Doctor: Dr. Brannon
Nurse: Nurse Joy
```

Filter Feature:

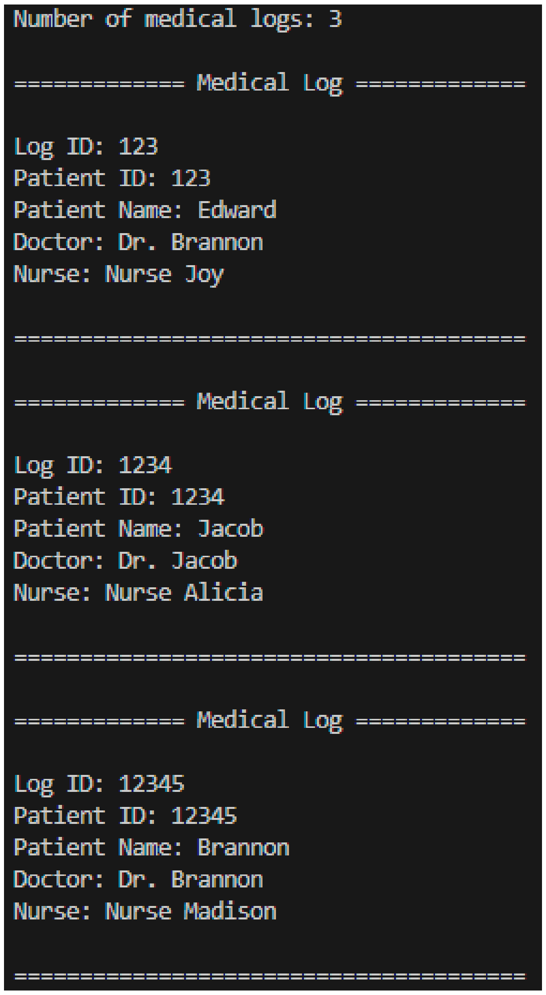


```
========= Filter Appointments =========

Choose an Appointment Status:
1. Scheduled
2. Cancelled

=======================================
Enter your choice [1 or 2]: 1

Appointment ID: 1234
Patient: Jacob
Date and Time: 1:00 PM - 3:00 PM
Status: Scheduled
Doctor: Dr. Jacob
Nurse: Nurse Alicia

Appointment ID: 12345
Patient: Brannon
Date and Time: 5:00 PM - 7:00 PM
Status: Scheduled
Doctor: Dr. Brannon
Nurse: Nurse Madison
```



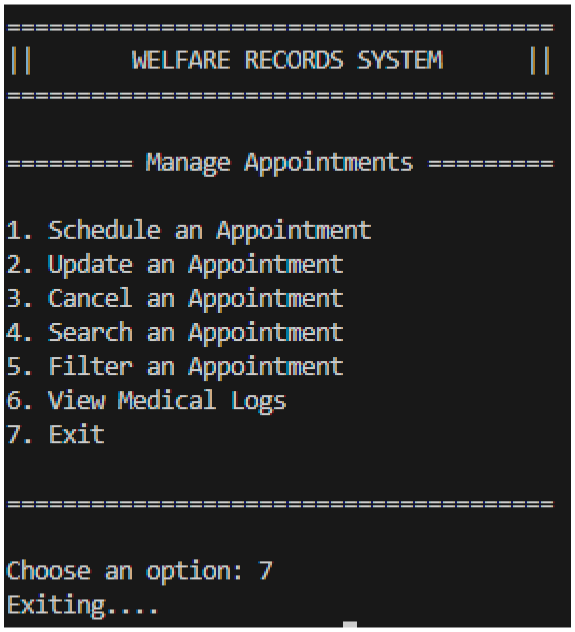
```
========= Filter Appointments =========

Choose an Appointment Status:
1. Scheduled
2. Cancelled

=======================================
Enter your choice [1 or 2]: 2

Appointment ID: 123
Patient: Edward
Date and Time: 10:00 AM - 12:00 PM
Status: Cancelled
Doctor: Dr. Brannon
Nurse: Nurse Joy
```

View Medical Log Feature:



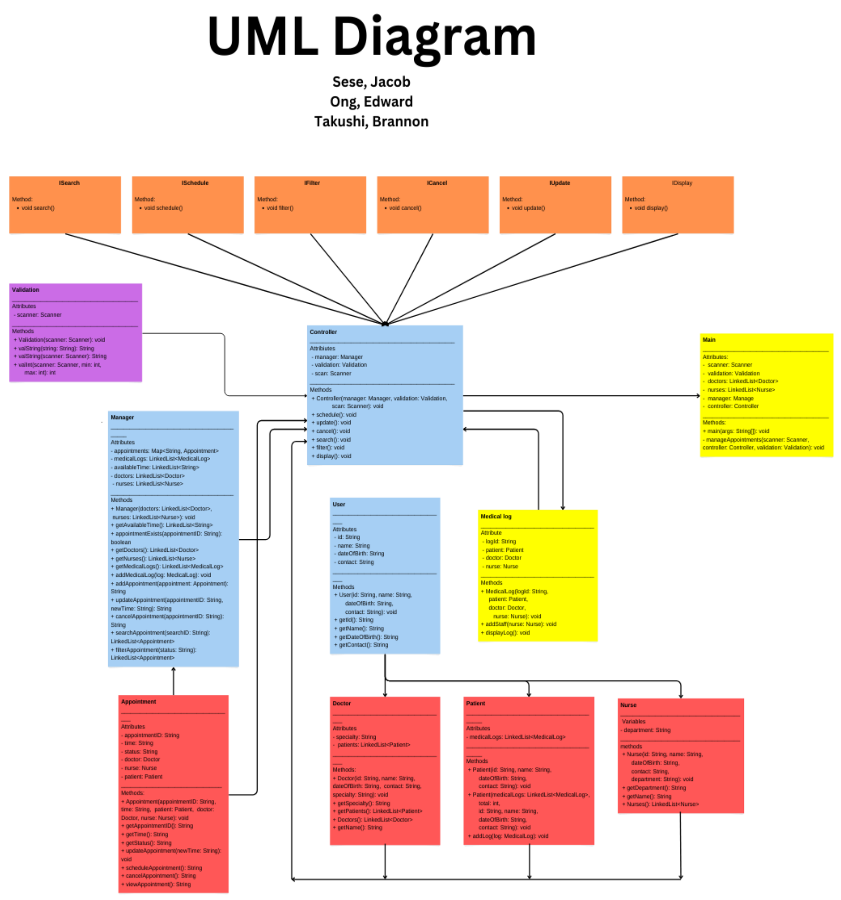
```
Number of medical logs: 3

============= Medical Log =============

Log ID: 123
Patient ID: 123
Patient Name: Edward
Doctor: Dr. Brannon
Nurse: Nurse Joy

=======================================

============= Medical Log =============

Log ID: 1234
Patient ID: 1234
Patient Name: Jacob
Doctor: Dr. Jacob
Nurse: Nurse Alicia

=======================================

============= Medical Log =============

Log ID: 12345
Patient ID: 12345
Patient Name: Brannon
Doctor: Dr. Brannon
Nurse: Nurse Madison

=======================================
```

Exit Feature:



```
========================================
||       WELFARE RECORDS SYSTEM       ||
========================================


========= Manage Appointments =========

1. Schedule an Appointment
2. Update an Appointment
3. Cancel an Appointment
4. Search an Appointment
5. Filter an Appointment
6. View Medical Logs
7. Exit

========================================

Choose an option: 7
Exiting....
```

UML Diagram


UML Diagram
Sese, Jacob
Ong, Edward
Takushi, Brannon
ISearch
Method:
• void search()
ISchedule
Method:
• void schedule()
IFilter
Method:
• void filter()
ICancel
Method:
• void cancel()
IUpdate
Method:
• void update()
IDisplay
Method:
• void display()
Validation
Attributes
- scanner: Scanner
Methods
+ Validation(scanner: Scanner): void
+ valString(string: String): String
+ valString(scanner: Scanner): String
+ valInt(scanner: Scanner, min: int, max: int): int
Controller
Attributes
- manager: Manager
- validation: Validation
- scan: Scanner
Methods
+ Controller(manager: Manager, validation: Validation, scan: Scanner): void
+ schedule(): void
+ update(): void
+ cancel(): void
+ search(): void
+ filter(): void
+ display(): void
Main
Attributes:
- scanner: Scanner
- validation: Validation
- doctors: LinkedList<Doctor>
- nurses: LinkedList<Nurse>
- manager: Manage
- controller: Controller
Methods:
+ main(args: String[]): void
- manageAppointments(scanner: Scanner, controller: Controller, validation: Validation): void
Manager
Attributes
- appointments: Map<String, Appointment>
- medicalLogs: LinkedList<MedicalLog>
- availableTime: LinkedList<String>
- doctors: LinkedList<Doctor>
- nurses: LinkedList<Nurse>
Methods
+ Manager(doctors: LinkedList<Doctor>, nurses: LinkedList<Nurse>): void
+ getAvailableTime(): LinkedList<String>
+ appointmentExists(appointmentID: String): boolean
+ getDoctors(): LinkedList<Doctor>
+ getNurses(): LinkedList<Nurse>
+ getMedicalLogs(): LinkedList<MedicalLog>
+ addMedicalLog(log: MedicalLog): void
+ addAppointment(appointment: Appointment): String
+ updateAppointment(appointmentID: String, newTime: String): String
+ cancelAppointment(appointmentID: String): String
+ searchAppointment(searchID: String): LinkedList<Appointment>
+ filterAppointment(status: String): LinkedList<Appointment>
User
Attributes
- id: String
- name: String
- dateOfBirth: String
- contact: String
Methods
+ User(id: String, name: String, dateOfBirth: String, contact: String): void
+ getId(): String
+ getName(): String
+ getDateOfBirth(): String
+ getContact(): String
Medical log
Attribute
- logId: String
- patient: Patient
- doctor: Doctor
- nurse: Nurse
Methods
+ MedicalLog(logId: String, patient: Patient, doctor: Doctor, nurse: Nurse): void
+ addStaff(nurse: Nurse): void
+ displayLog(): void
Appointment
Attributes
- appointmentID: String
- time: String
- status: String
- doctor: Doctor
- nurse: Nurse
- patient: Patient
Methods:
+ Appointment(appointmentID: String, time: String, patient: Patient, doctor: Doctor, nurse: Nurse): void
+ getAppointmentID(): String
+ getTime(): String
+ getStatus(): String
+ updateAppointment(newTime: String): void
+ scheduleAppointment(): String
+ cancelAppointment(): String
+ viewAppointment(): String
Doctor
Attributes
- specialty: String
- patients: LinkedList<Patient>
Methods:
+ Doctor(id: String, name: String, dateOfBirth: String, contact: String, specialty: String): void
+ getSpecialty(): String
+ getPatients(): LinkedList<Patient>
+ Doctors(): LinkedList<Doctor>
+ getName(): String
Patient
Attributes
- medicalLogs: LinkedList<MedicalLog>
Methods
+ Patient(id: String, name: String, dateOfBirth: String, contact: String): void
+ Patient(medicalLogs: LinkedList<MedicalLog>, total: int, id: String, name: String, dateOfBirth: String, contact: String): void
+ addLog(log: MedicalLog): void
Nurse
Variables
- department: String
methods
+ Nurse(id: String, name: String, dateOfBirth: String, contact: String, department: String): void
+ getDepartment(): String
+ getName(): String
+ Nurses(): LinkedList<Nurse>

GITHUB LINK:

https://github.com/Voulch/Finals-Health-Care-Management-Group-3

VIDEO LINK:

https://drive.google.com/file/d/1zI8qx38l3FO80VgGZQqBmkJsClfQL8WE/view?usp=sharing