SEG2105 Project Report

David Chen: 300283888

Zilin Liu: 300310845

Anjelika Babayan: 300308106

Arnav Chaturvedi: 300304150

Date: December 4, 2023

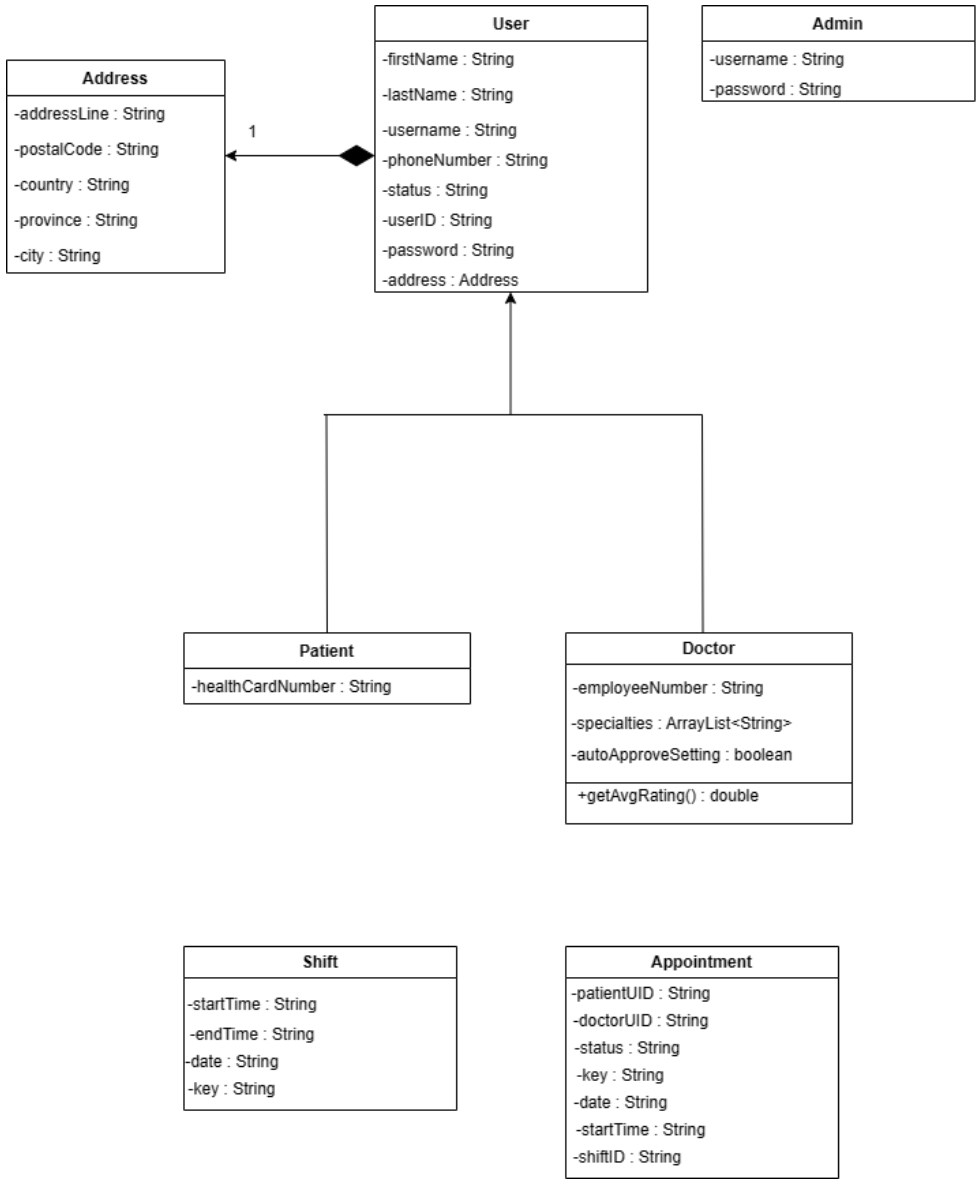
Professor: Hussein Al Osman

Introduction

In the past couple of months, our team has developed a Healthcare Appointment Management System app using Android Studio. To use the app, users must submit a registration form and be approved by an administrator. . Upon approval, patients can book available appointments based on doctor specialties, and doctors can enter their shifts and approve patient appointment requests. User information, shifts, appointments and doctor ratings are all stored in realtime database and queried as needed.

In the following pages, a UML diagram, pictures of our app, as well as each team member's contribution towards the project will be found.

UML Class Diagram

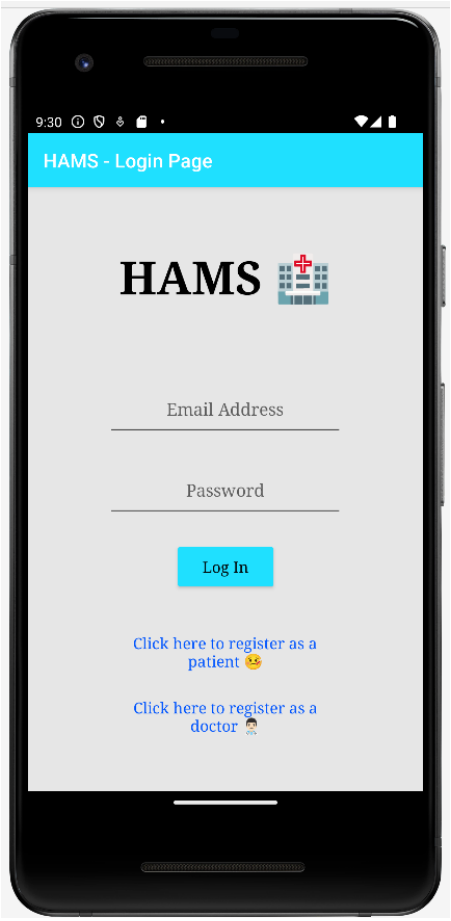


Contribution Table

	Deliverable 1	**Deliverable 2**	**Deliverable 3**	**Deliverable 4**
David	Created UI for doctor registration form. Implemented functionality for doctor registration form.	Created the lists to hold the pending and rejected registration requests. Created UI for admin screen. Created list adapter for users.	Created UI for the doctor welcome screen and the doctor shifts screen. Created list adapters for shift and appointments.	Created UI for the main patient welcome screen. Added rating attribute to Doctors. Implemented adding appointment slots.
Arnav	Implemented functionality for storing user data on registration and authentication using firebase	Implemented database functionality for retrieving users based on approval status and updating their status when needed	Implemented functionality for adding and retrieving shifts, retrieving and updating appointments.	Ensured appointments occurring within an hour cannot be deleted, stored ratings in the database, implemented searching based on specialties, functionality for creating appointments, ensured appointments corresponding to taken shifts cannot be deleted.
Anjelika	Implemented Patient.java, Admin.java, User.java, and Doctor.java	Creating Address.java class, fixing database hierarchy. Implemented email functionality	Implemented deleteshift functionality, Approve and Reject appointments functionality, and add shift conflict functionality for doctor.	Implement the search doctor specialty dropdown. Fix error checking when the doctor adds shifts.
Zilin	Create UI for patient registration form. Implemented functionality for patient registration form.	UML	UML and implement the button that approves all requests of the welcome screen doctor.	UML and implement 4 Unit test cases.

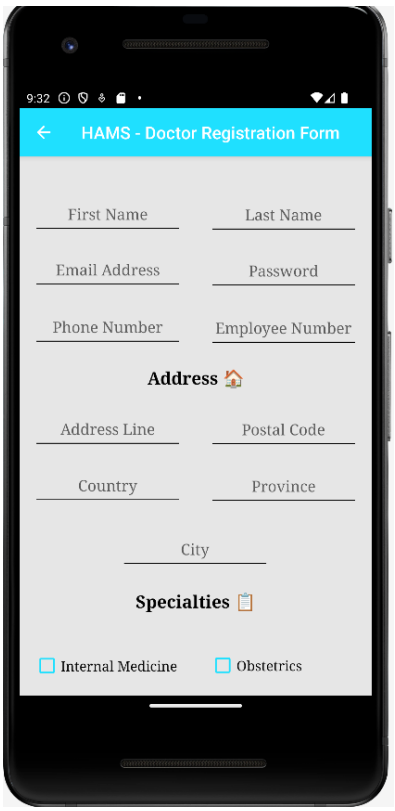
Screenshots

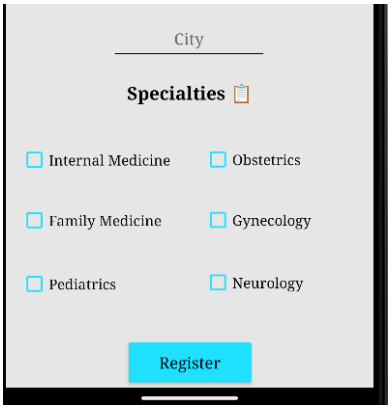
Main Login Page:



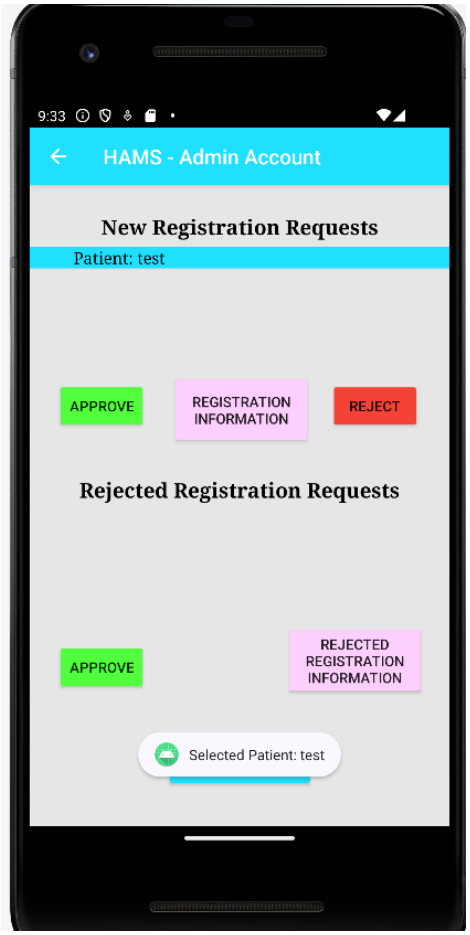
Patient Registration Form:

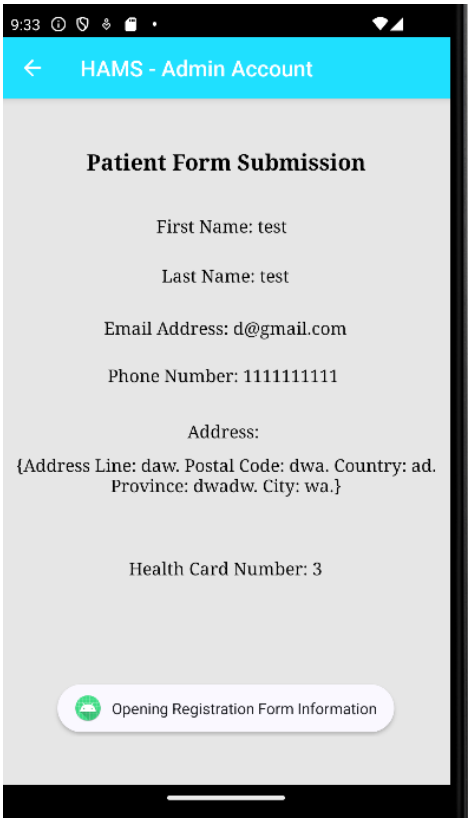
Doctor Registration Form:



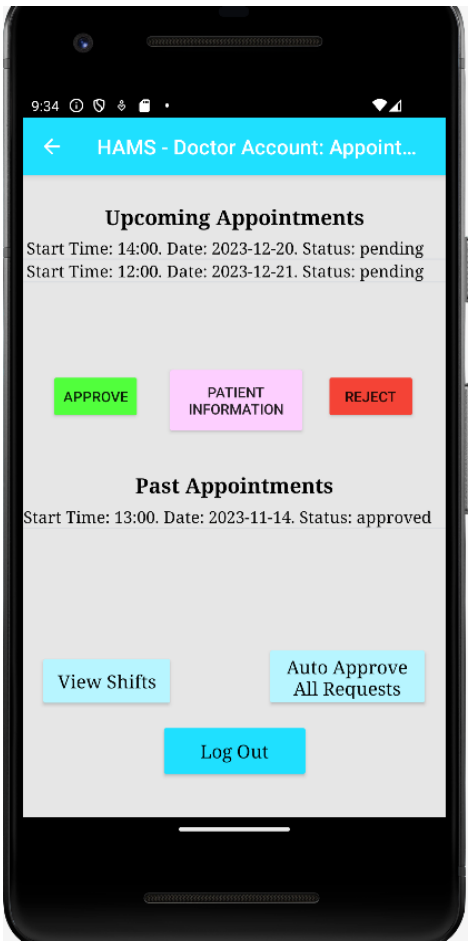


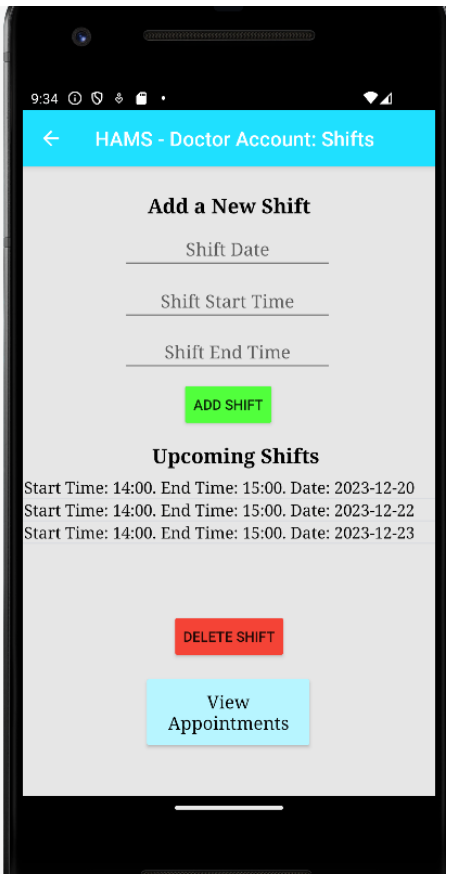
Admin Account:



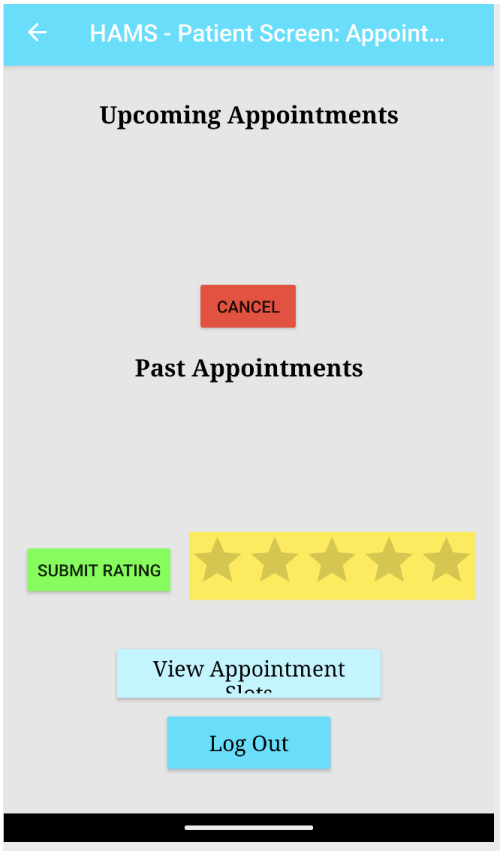


Doctor Account:

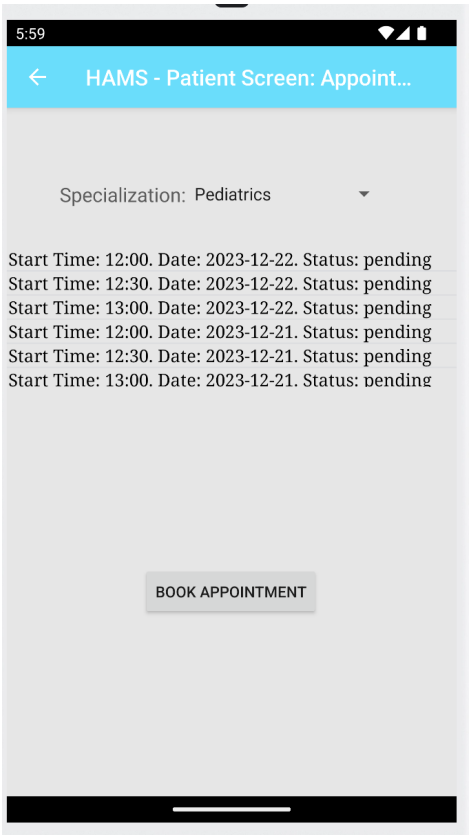

9:34
HAMS - Doctor Account: Appoint...
Upcoming Appointments
Start Time: 14:00. Date: 2023-12-20. Status: pending
Start Time: 12:00. Date: 2023-12-21. Status: pending
APPROVE
PATIENT INFORMATION
REJECT
Past Appointments
Start Time: 13:00. Date: 2023-11-14. Status: approved
View Shifts
Auto Approve All Requests
Log Out

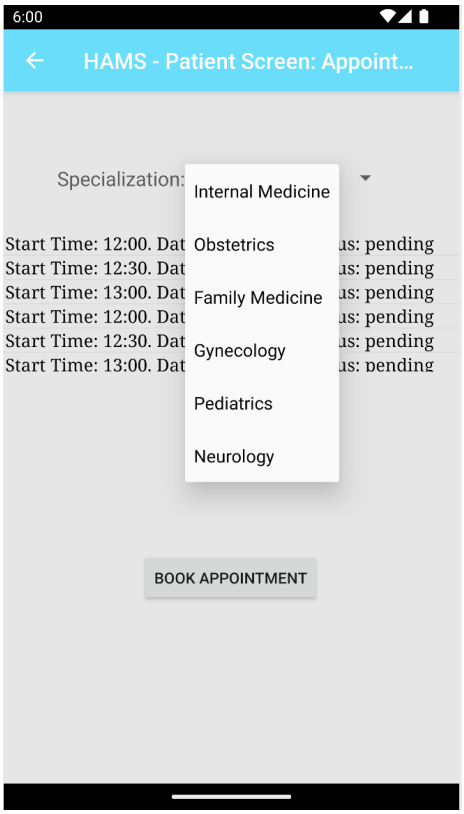

9:34
HAMS - Doctor Account: Shifts
Add a New Shift
Shift Date
Shift Start Time
Shift End Time
ADD SHIFT
Upcoming Shifts
Start Time: 14:00. End Time: 15:00. Date: 2023-12-20
Start Time: 14:00. End Time: 15:00. Date: 2023-12-22
Start Time: 14:00. End Time: 15:00. Date: 2023-12-23
DELETE SHIFT
View Appointments

Patient Account:



Patient Screen Appointment Slots


5:59
HAMS - Patient Screen: Appoint...
Specialization: Pediatrics
Start Time: 12:00. Date: 2023-12-22. Status: pending
Start Time: 12:30. Date: 2023-12-22. Status: pending
Start Time: 13:00. Date: 2023-12-22. Status: pending
Start Time: 12:00. Date: 2023-12-21. Status: pending
Start Time: 12:30. Date: 2023-12-21. Status: pending
Start Time: 13:00. Date: 2023-12-21. Status: pending
BOOK APPOINTMENT


6:00
HAMS - Patient Screen: Appoint...
Specialization:
Internal Medicine
Obstetrics
Family Medicine
Gynecology
Pediatrics
Neurology
BOOK APPOINTMENT

Lessons Learned

Design Patterns For Database Functionality

Throughout deliverables 2, 3 and 4 our firebase code drastically grew in complexity. Due to this, writing, debugging and reading code in activity classes became increasingly difficult. To prevent this, we could have used the facade pattern to implement general database functionality. For example, methods to add elements to the database, update the database and more. This would have made our code more readable, and even allowed us to switch to a more appropriate database as our querying became more complex.

Furthermore, we could have made use of the proxy pattern to create a more organised structure in our program while retaining the freedom to only load parts of the user's information that were needed. A proxy class and better organised database would have simplified querying and made our program more readable by increasing readability and modularity.

Database Structure

The structure of our database was an important consideration throughout the development process. Using firebase's documentation, we learned that data structures should be kept as flat as it is easier and nested data should be avoided. We followed this principle by moving appointments and shifts to separate parts of the database. This removed the need to instantiate full doctor or patient objects to get related appointments and shifts and made our queries faster.

UI

When it came to the UI, we learned that we could have made the design better by using navigation tabs or drawers instead of creating more empty view activities. This would have been helpful because all 3 users need to see at least more than 2 different screens to see all the information in their account.

Finally, we could have used more methods and static classes to avoid code duplication. We ended up copying and pasting code for UI elements many times which could have been avoided by using this approach. Specifically, listviews were used a lot throughout the app and their code may have been simpler with such a change.