Android Project AppointWell Application

SEG 2105A - Introduction to Software Engineering

Fall 2023
Faculty of Engineering
University of Ottawa

Course Coordinator: Hussein Al Osman

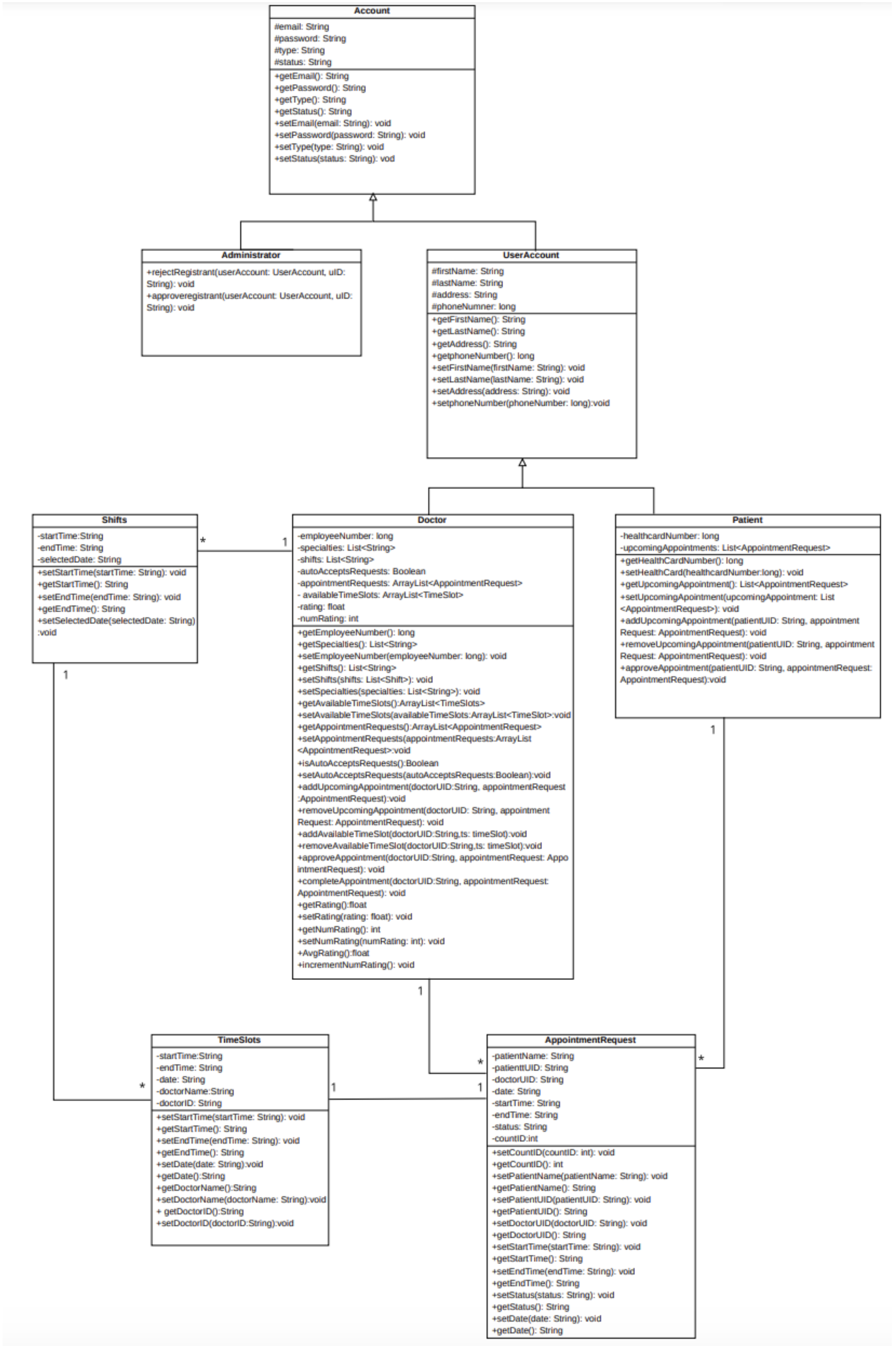
Duration: 09 Sep 2023 - 06 Dec 2023
Date of Submission: 04 December 2023

Authors:
Abbasi Mahreen, 300317493
Chan Kwong Lun Karen , 300280519
Leung-Pah-Hang Oceane, 300298116
Luchmun Rhadesh , 300184364
Perbhoo Vivek, 300241343
Pyndiah Tanya, 300241687

Overview

We developed Appointwell, an Android mobile application that implements a Healthcare Appointment Management System (HAMS) tailored for a telehealth clinic. AppointWell's primary objective is to optimize healthcare appointment scheduling and management. The system caters to three distinct user roles: Patient, Doctor, and Administrator and ensures seamless and user-friendly experience for all stakeholders involved.

UML Diagram


Account
#email: String
#password: String
#type: String
#status: String
+getEmail(): String
+getPassword(): String
+getType(): String
+getStatus(): String
+setEmail(email: String): void
+setPassword(password: String): void
+setType(type: String): void
+setStatus(status: String): vod
Administrator
+rejectRegistrant(userAccount: UserAccount, uID: String): void
+approveregistrant(userAccount: UserAccount, uID: String): void
UserAccount
#firstName: String
#lastName: String
#address: String
#phoneNumner: long
+getFirstName(): String
+getLastName(): String
+getAddress(): String
+getphoneNumber(): long
+setFirstName(firstName: String): void
+setLastName(lastName: String): void
+setAddress(address: String): void
+setphoneNumber(phoneNumber: long):void
Shifts
-startTime:String
-endTime: String
-selectedDate: String
+setStartTime(startTime: String): void
+getStartTime(): String
+setEndTime(endTime: String): void
+getEndTime(): String
+setSelectedDate(selectedDate: String) :void
*
1
Doctor
-employeeNumber: long
-specialties: List<String>
-shifts: List<String>
-autoAcceptsRequests: Boolean
-appointmentRequests: ArrayList<AppointmentRequest>
- availableTimeSlots: ArrayList<TimeSlot>
-rating: float
-numRating: int
+getEmployeeNumber(): long
+getSpecialties(): List<String>
+setEmployeeNumber(employeeNumber: long): void
+getShifts(): List<String>
+setShifts(shifts: List<Shift>): void
+setSpecialties(specialties: List<String>): void
+getAvailableTimeSlots():ArrayList<TimeSlots>
+setAvailableTimeSlots(availableTimeSlots:ArrayList<TimeSlot>:void
+getAppointmentRequests():ArrayList<AppointmentRequest>
+setAppointmentRequests(appointmentRequests:ArrayList <AppointmentRequest>:void
+isAutoAcceptsRequests():Boolean
+setAutoAcceptsRequests(autoAcceptsRequests:Boolean):void
+addUpcomingAppointment(doctorUID:String, appointmentRequest :AppointmentRequest):void
+removeUpcomingAppointment(doctorUID: String, appointment Request: AppointmentRequest): void
+addAvailableTimeSlot(doctorUID:String,ts: timeSlot):void
+removeAvailableTimeSlot(doctorUID:String,ts: timeSlot):void
+approveAppointment(doctorUID:String, appointmentRequest: Appo intmentRequest): void
+completeAppointment(doctorUID:String, appointmentRequest: AppointmentRequest): void
+getRating():float
+setRating(rating: float): void
+getNumRating(): int
+setNumRating(numRating: int): void
+AvgRating():float
+incrementNumRating(): void
Patient
-healthcardNumber: long
-upcomingAppointments: List<AppointmentRequest>
+getHealthCardNumber(): long
+setHealthCard(healthcardNumber:long): void
+getUpcomingAppointment(): List<AppointmentRequest>
+setUpcomingApointment(upcomingAppointment: List <AppointmentRequest>): void
+addUpcomingAppointment(patientUID: String, appointment Request: AppointmentRequest): void
+removeUpcomingAppointment(patientUID: String, appointment Request: AppointmentRequest): void
+approveAppointment(patientUID: String, appointmentRequest: AppointmentRequest):void
1
1
1
TimeSlots
-startTime:String
-endTime: String
-date: String
-doctorName:String
-doctorID: String
+setStartTime(startTime: String): void
+getStartTime(): String
+setEndTime(endTime: String): void
+getEndTime(): String
+setDate(date: String):void
+getDate():String
+getDoctorName():String
+setDoctorName(doctorName: String):void
+ getDoctorID():String
+setDoctorID(doctorID:String):void
*
1
1
*
*
AppointmentRequest
-patientName: String
-patienttUID: String
-doctorUID: String
-date: String
-startTime: String
-endTime: String
-status: String
-countID:int
+setCountID(countID: int): void
+getCountID(): int
+setPatientName(patientName: String): void
+getPatientName(): String
+setPatientUID(patientUID: String): void
+getPatientUID(): String
+setDoctorUID(doctorUID: String): void
+getDoctorUID(): String
+setStartTime(startTime: String): void
+getStartTime(): String
+setEndTime(endTime: String): void
+getEndTime(): String
+setStatus(status: String): void
+getStatus(): String
+setDate(date: String): void
+getDate(): String

Contributions of team members

I. Deliverable 1

Team Member	Contribution
Mahreen	● AppointWell Splash Screen XML designing ● Patient and Doctor classes and their associated functions (setters and getters) ● UML Class Diagram
Karen	● Created GitHub Classroom Repo ● Main Page Welcome Screen upon successful login and signup
Rhadesh	● Login Page XML designing ● Login feature implementation via Firebase Authentication implementation ● Logoff functionality
Tanya	● Registration Page Sign up Button Functionality, Patient & Doctor Toggle Button functionality ● Registration: Validation of variables of First name, Last name, Email address, Password, Specialties ● Administrator and User Class
Vivek	● Registration Page XML designing ● Registration: Validation of variables of Phone number, Address, Health card number, Employee number ● Linked views and associated activity together ● Submitted APK and UML Diagram
Oceane	● Registration EditText elements error message implementation ● Firebase RealTime Database implementation of users data

II. Deliverable 2

Team Member	**Contribution**
Tanya	● Reworked on the Firebase RealTime Database to make Approved Users and Pending requests be two separate nodes with each patient/doctor associated info in the appropriate node ● Single Registration Request View XML designing ● Inflater implementation to show each registration request ● Administrator Pending View: accept and reject functionality (retrieving Patient/Doctor node from Pending Requests and redirecting it to appropriate parent node)
Vivek	● Administrator Requests List Page XML designing ● Implementation of the toggle Pending and Rejected Toggle Button ● Administrator Reject View: accept functionality (retrieving Patient/Doctor node from Reject and redirecting it to Approved Users node) ● Linked views and associated activity together ● Submitted APK and UML Diagram
Oceane	● Slider View containing info associated for a single request XML designing ● Animation of Slider View implementation (On click of Registration Request View, the Slider View slide up from the bottom) ● Updated Patient & Doctor Activity to notify them of their request status (Approved, Rejected or Processing) ● UML Class Diagram ● (Attempted to do the email bonus but was not able to finish it due to Google no longer allowing less secure apps)

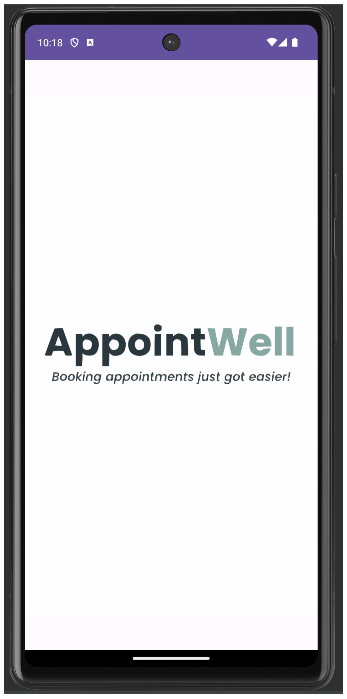
III. Deliverable 3

Team Member	**Contribution**
Mahreen	● Doctor Appointment Request Page: used recycler view to see list of upcoming appointments ● Doctor Appointment Request Page: approve or reject an appointment implementation ● Doctor Appointment Request Page: used recycler view to see a list of past appointments
Karen	● Doctor & Patient Main Page XML designing ● Doctor Shift Page XML designing ● Doctor Shift Page: used recycler view to see list of upcoming shifts they are working on ● Doctor Shift Page: add a new shift by specifying the date, start-time, and end-time of the shift ● Doctor Shift Page: validation for the creation of a new shift implementation ● Doctor Shift Page: delete an existing shift implementation ● UML Diagram
Rhadesh	● Doctor Appointment Requests Page XML designing ● Doctor auto-approve patient requests implementation ● Doctor Appointment Request Page: used Slider View previously created to view info of patient (On click of Appointment Request View, the Slider View slide up from the bottom) ● Doctor Appointment Request Page: cancel a previously approved appointment implementation
Tanya, Vivek & Oceane	● Linked views and associated activity together ● Helped in the debugging ● Submitted APK and UML Diagram

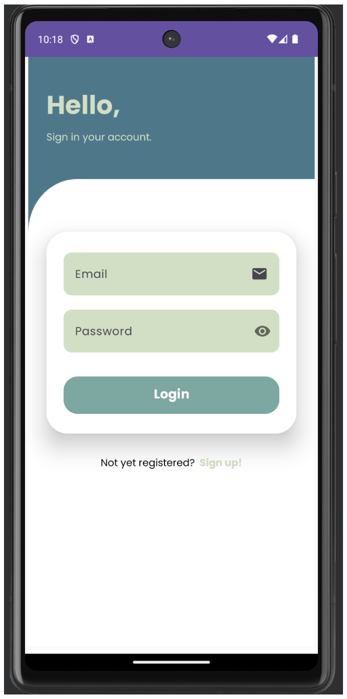
IV. Deliverable 4

Team Member	**Contribution**
Mahreen	● Designed the 4 Unit test cases ● Patient Feature: rate a doctor (after appointment) ● Patient Appointment Page: used a recycler view to see a list of past appointments
Karen	● UML Diagram ● Doctor Shift Page: updated the deleteShift method so that doctor cannot delete an appointment if it a Patient or more has already booked an appointment during that shift
Rhadesh	● Patient Appointment Page XML designing ● Patient Appointment Page: used a recycler view to see a list of upcoming appointments when appointment is booked ● Patient Appointment Page: cancel an upcoming appointment implementation
Tanya	● Doctor Shift Page: updated the addshift method such that it adds the shift hours as 30min appointments in the database as a list under the associated doctor node ● Patient Browsing Page: remove selected appointment time slot from available appointments view when this particular appointment is booked implementation (move the time slot from availableAppointments to upcomingAppointments in database)
Vivek	● Patient Browsing Page XML designing ● Patient Browsing Page: used a recycler view to see a list of available booking hours depending on specialty specified in search bar ● Patient Browsing Page: created adapter and associated xml file for the recycler view item
Oceane	● Report writing

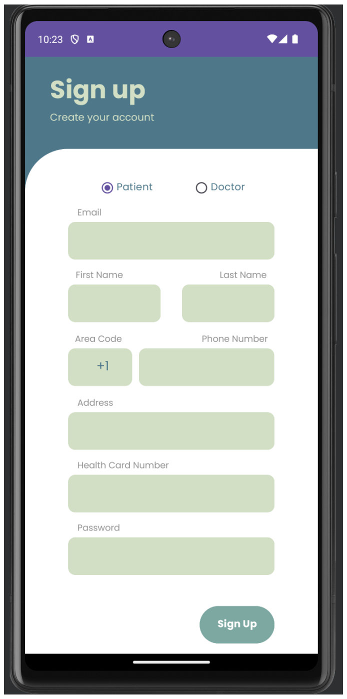
User Interface



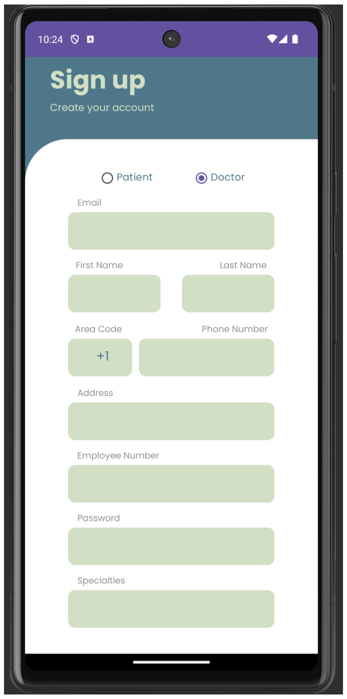
1 Splash Screen



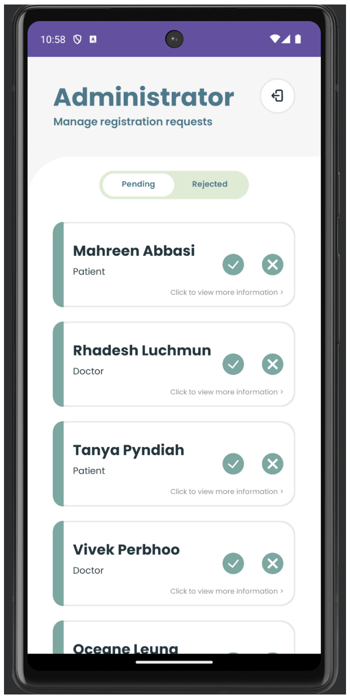
2 Login



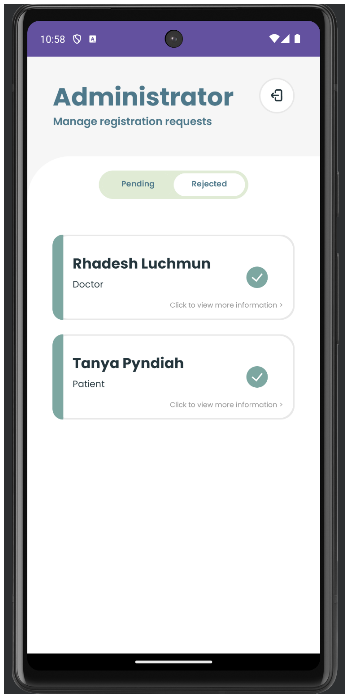
3 Sign Up Patient



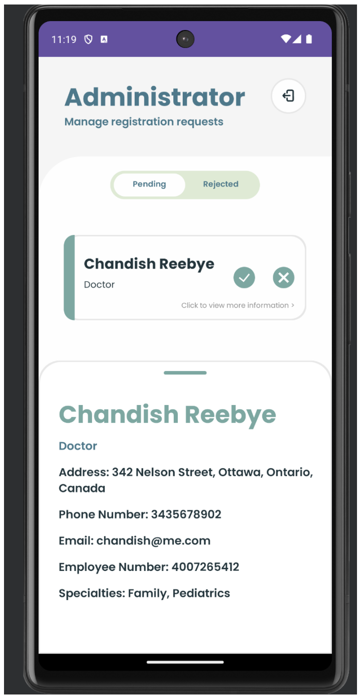
4 Sign Up Doctor


10:58
Administrator
Manage registration requests
Pending
Rejected
Mahreen Abbasi
Patient
Click to view more information ›
Rhadesh Luchmun
Doctor
Click to view more information ›
Tanya Pyndiah
Patient
Click to view more information ›
Vivek Perbhoo
Doctor
Click to view more information ›
Oceane Leung

5 Admin Pending Requests


10:58
Administrator
Manage registration requests
Pending
Rejected
Rhadesh Luchmun
Doctor
Click to view more information ›
Tanya Pyndiah
Patient
Click to view more information ›

6 Admin Rejected Requests


11:19
Administrator
Manage registration requests
Pending
Rejected
Chandish Reebye
Doctor
Click to view more information ›

Chandish Reebye

Doctor

Address: 342 Nelson Street, Ottawa, Ontario, Canada

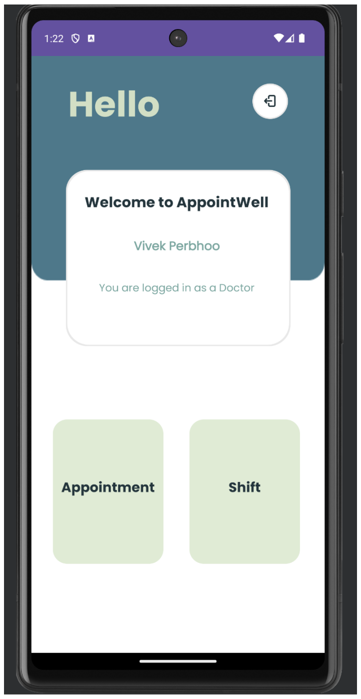
Phone Number: 3435678902

Email: chandish@me.com

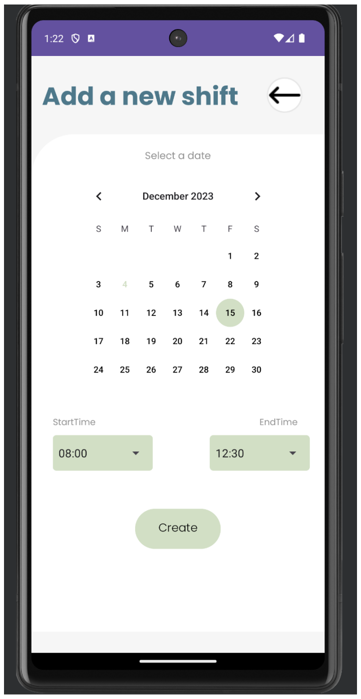
Employee Number: 4007265412

Specialties: Family, Pediatrics

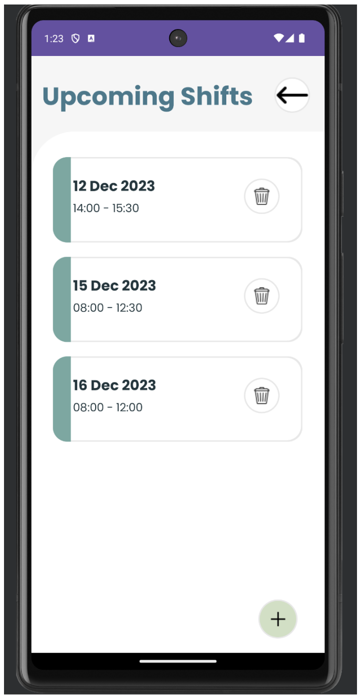
7 Admin Slider View



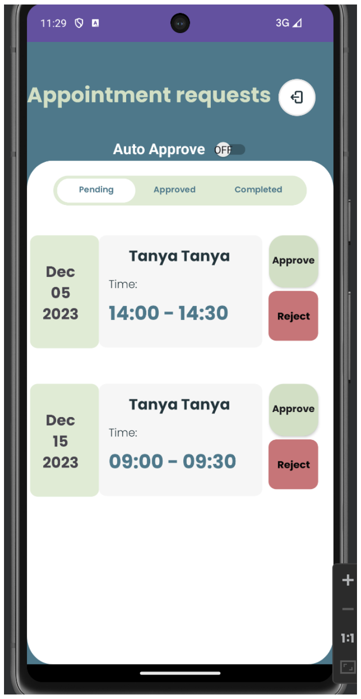
8 Doctor Main Page



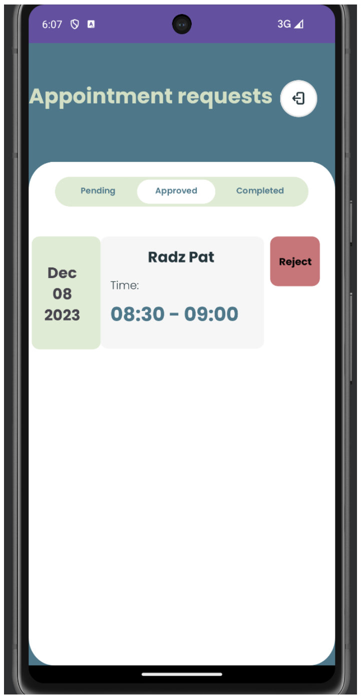
9 Doctor Shift Adder



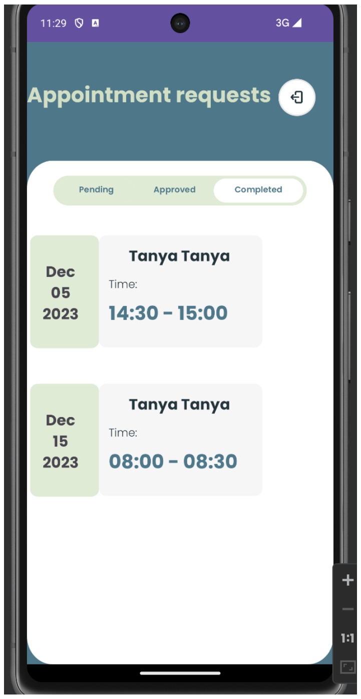
10 Doctor Shift List



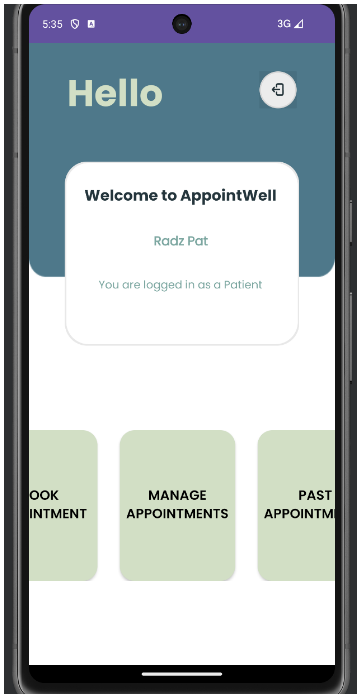
11 Doctor Pending Appointment



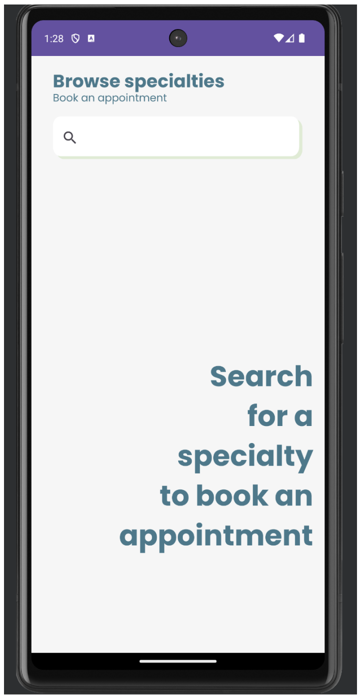
12 Doctor Approved Appointment



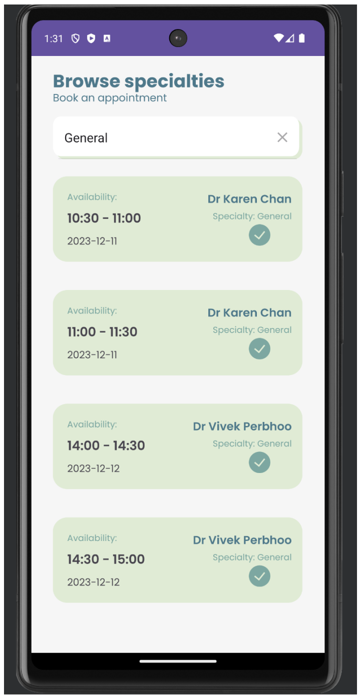
13 Doctor Completed Appointment



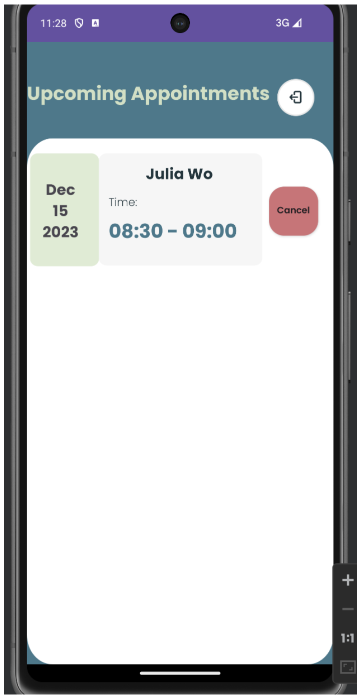
14 Patient Main Page



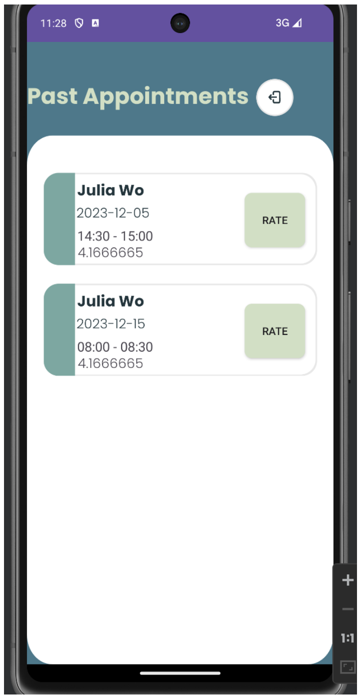
15 Patient Browsing Page



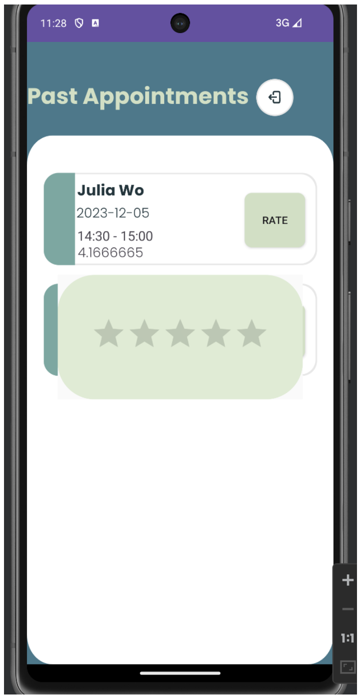
16 Patient Booking Search



17 Patient Manage Appointment



18 Patient Rate a Doctor



19 Patient Rate a Doctor

Lesson Learnt

i) We learnt a lot about Git and Github. Git helped us track changes and supported local experimentation by letting each team member work on different parts of the project at the same time. We learnt how to add, commit, push, pull, stash, merge and reset our code. It tremendously helped us collaborate, and work faster.

ii) It was our first time working on a mobile application,and thus our first time working with Android Studio. We discovered the difference between UI and backend and how to link them together. We also uncovered the different types of layouts, views and other elements in android studio.

iii) We learnt how to use Figma, a cloud-based designing tool, where we would create our UI before doing its corresponding XML file. We did this so that everyone had a clear idea of what had to be done.

iv) We acquired knowledge on Adapters and RecyclerViews. We learnt that adapters provide a bridge between the data (in our case from our database) and the RecyclerView in a way that enables our UI to display that data.

v) As it was our first time using Firebase, we learnt a lot about firebase: its syntax, how to read data (by attaching a listener to a database reference, addValueEventListener), how to write data (by using the push() or setValue() method on a database reference) and how to store data efficiently. We also gained valuable insights from bugs and run time errors that initially appeared insurmountable.

`addListenerForSingleValueEvent`: We could not understand why our database was not updating even though we called the addListenerForSingleValueEvent multiple times. After hours of debugging we figured out that the addListenerForSingleValueEvent fetches the data once and then detaches the listener and that calling it multiple times will still retrieve the same snapshot of the data as the method will retrieve the snapshot from the local cache of data after detaching itself. The solution to our problem was thus to use the keepSynced(true) function so that Firebase keeps a synchronized local cache of the data at the specified database reference.

`addValueEventListener`: We did not know why our app kept going back to the login page when we were creating shifts for doctors in the deliverable 3. We later discovered that it was because the onDataChange() method of the addValueEventListener of the Sign Up activity was being called. This onDataChange() method was designed to redirect users to login page after patient/doctor sign up.

This happened because the addValueEventListener in Firebase is designed to listen for changes in the specified location in real-time, and it remains active until it is explicitly removed. The solution was to call removeEventListener in the appropriate lifecycle method, to ensure that the listener is removed when the activity is no longer active.

vi) We learnt about regular expression (regex) and how to use it efficiently while doing our password validation.

vii) We gained insight on how to make animations in Android Studio while making a SliderView for our Patient/Doctor “more information” implementation.

viii) We became familiar with 0-based and 1-based formats while working with the LocalDate class in our Deliverable 3. We learnt that when working with the LocalDate class in Java, months are represented in a 1-based format which means that January is represented as 1, February as

2, and so on. But if the data you're working with (such as the date & month from the date picker from our dateTimeShiftSelector class) uses a 0-based format, you need to add 1 to convert it to the 1-based format expected by LocalDate.

Conclusion

Our project journey was a constant learning experience that spanned a variety of tools, technologies, and concepts in the realm of software development. Git and Github became indispensable collaborators, guiding us through version control intricacies and empowering us to work collaboratively. Android Studio introduced us to the dynamic interplay between UI and backend, shedding light on elements crucial for mobile application development. Firebase unfolded as a powerful tool for real-time data management.

Through overcoming challenges, debugging runtime errors, and gaining insights from each stumbling block, the AppointWell mobile application became a testament to the iterative nature of development — a process of continuous learning and improvement.