

Healthcare Appointment Management System

SEG 2105 - Introduction to Software Engineering

Fall 2023

University of Ottawa

Professor: Hussein Al Osman

Group 11:

Tanner Frisch

Michael Massaad

Christian Sampson

Jayson Seip

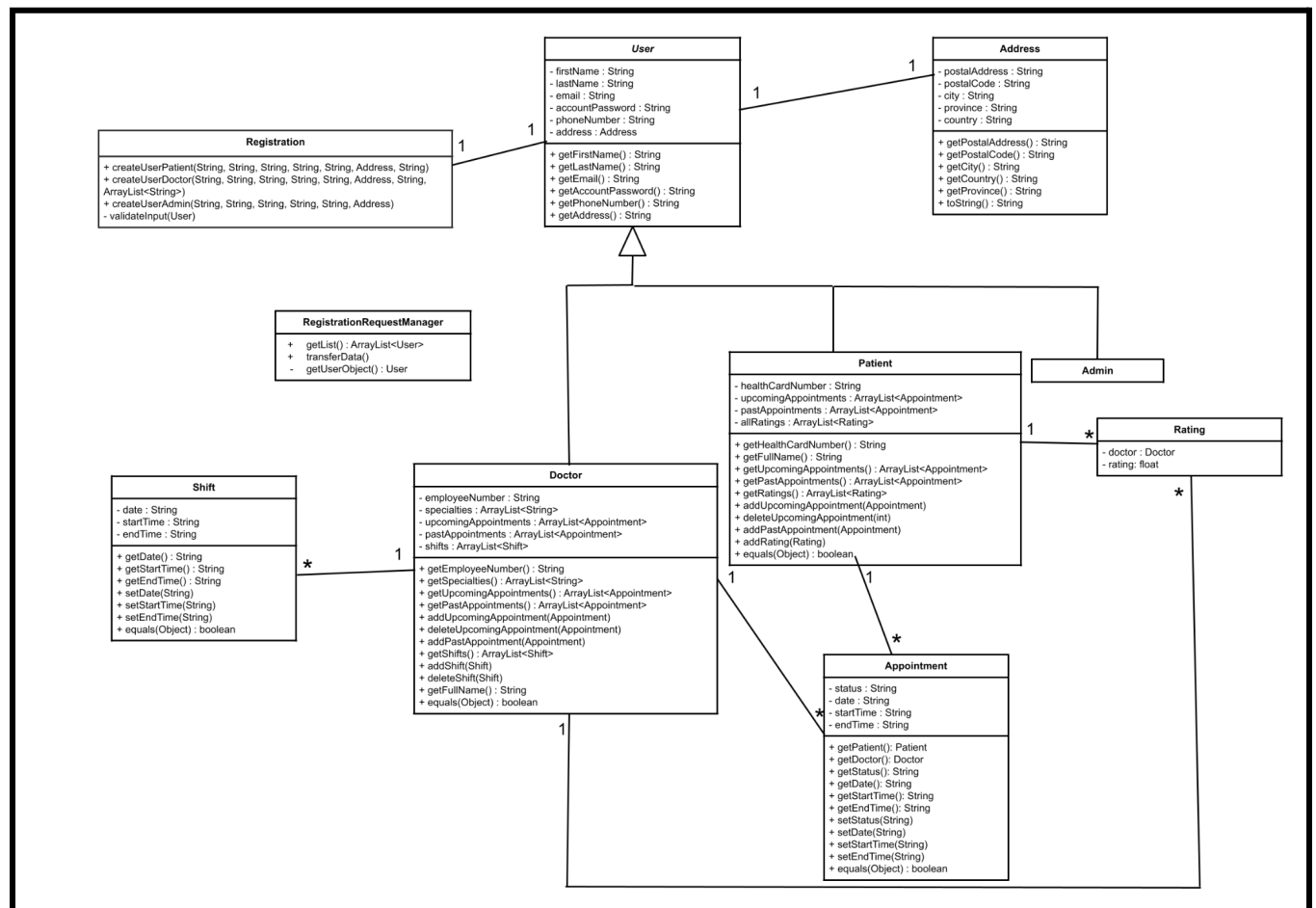
Aashish Suresh

Eric Zhou

Introduction

This project is an Android application developed with Java and Android Studio that provides functionality for an online healthcare appointment management system. The application allows telehealth clinics to better organize appointments between doctors and patients. It allows doctors and patients to register for accounts, patients to book appointments with doctors and view their upcoming and past appointments, and doctors to oversee shifts and view appointments with patients.

UML Class Diagram



Team Contributions

Table 1: Contributions for Deliverable 1

Tanner	Michael	Christian	Jayson	Aashish	Eric
Created Doctor class	Created Patient class	Implemented login for Doctor	Created User class	Validated postal code input	Created team on Github Classroom
Validated Registration input	Created UML Class Diagram	Implemented sign out	Created Address class		Added UI elements for MainActivity
	Modified the UI for the sign out button	Created UML Class Diagram	Created UML Class Diagram		Implemented registration for Patient and Doctor
					Implemented UI for Sign Up
					Connected Firebase to the project and stored data
					Created Admin class
					Implemented login for Patient and Doctor
					Created Welcome Page
					Validated Registration Input
					Implemented sign out
					Added Javadoc to every method
					Created UML Class Diagram

Table 2: Contributions for Deliverable 2

Tanner	Michael	Christian	Jayson	Aashish	Eric
Added comments to functions	Implemented clickable aspect to individual items in displayed lists for the admin inbox for the registrations of accounts	Added UI elements for individual item display	Created Admin inbox UI		Changed Registration to pending in database
Added unit tests	Created display for individual items in displayed lists of the admin inbox		Implemented login status events for Doctor and Patient (Rejection/Pending)		Implemented retrieval of pending and rejected requests from database
	Updated UML Class Diagram				Implemented transfer of data between categories in the database (pending, rejected, accepted)
					Displayed lists from database in the admin inbox
					Implemented events for accepting and rejecting requests
					Updated UML Class Diagram


Table 3: Contributions for Deliverable 3

Tanner	Michael	Christian	Jayson	Aashish	Eric
Implemented individual Appointment display	Implemented buttons for Doctor to reach Shifts and Appointments	Created Shift class	Created Appointment class		Implemented Shift display
Implemented events for accepting and rejecting Appointments	Updated UML Class Diagram	Implemented Appointment list display	Implemented Approve all toggle switch		Implemented Shift adding and deleting
		Implemented layout to display patient information			Validated Shift-adding inputs
		Attempted DoctorAppointmentsActivity implementation, Implemented by Eric.			Implemented Appointment list display
					Implemented individual Appointment display
					Updated UML Class Diagram

Table 4: Contributions for Deliverable 4

Tanner	Michael	Christian	Jayson	Aashish	Eric
Implemented display for patient appointment details	Implemented the search bar for the patient to search for available appointments based on doctor specialty	Aided Michael in the implementation of the BookAppointment Activity	Implemented the patient listview for upcoming and past appointments	Helped Tanner debug the PatientDisplay activity; fixed emulator issues with canceling appointments	Implemented check for appointment conflicting with shift
Added more unit tests	Created UI elements for the booking activity and displaying available appointments of 30 minute time period	Aided Jayson in determining implementation for upcoming and past appointments listview	Assisted Tanner in rating system for doctor	Contributed to application screenshots descriptions	Assisted with appointment searching by specialty
Implemented a system for patients to rate doctors	Implemented the patient being able to book appointment, which updates the firebase for upcoming appointments	Aided Tanner in determining implementation for doctor rating activity			Assisted with appointment booking
	Updated UML diagram				Implemented events for accepting and rejecting /canceling appointments
	Assisted Eric in debugging				Updated UML Class Diagram

Application Screenshots

Screenshot	Description of Screenshot
	<p>This page is displayed when the user opens the application. It has a sign up button where if the user clicks on it, it will send them to set up an account. The log in button also allows users, when clicked, to log into their account.</p>



This page is displayed once a user clicks the “Sign Up” button, which allows them to specify whether they want to create a doctor (by clicking the “Sign Up as Doctor” button) or patient account (by clicking the “Sign Up as Patient” button).

Two side-by-side smartphone screens. The left screen shows the 'Patient Registration Form' with input fields for First Name, Last Name, Email, Password, Phone, Postal Address, Address, Postal Code, City/Town, Province/State, and Country. The right screen shows the same form with additional fields for Health Card Number and a red 'Submit' button at the bottom.

This page is displayed when the user clicks the “Sign Up as Patient” button. In this scrollable page, the user can fill the required information to create the patient account and click the “Submit” button to finalize and create the account. If the account was able to be created, a display text will indicate that the account has been create. If the account was not able to be created, it will notify the user with a display text if a specific information has been not properly inputted.

A mobile app screen titled "Doctor Registration Form" with a blue background. It contains a scrollable list of input fields: First Name, Last Name, Email, Password, Phone, Postal Address, Address, Postal Code, City/Town, Province/State, Country, and Employee Number. Each field has a white text input area.A mobile app screen showing the continuation of the "Doctor Registration Form". It contains input fields for Postal Address, Address, Postal Code, City/Town, Province/State, Country, Employee Number, and Specialties. At the bottom, there is a red "Submit" button.

This page is displayed when the user clicks the “Sign Up as Doctor” button. In this scrollable page, the user can fill the required information to create the doctor account and click the “Submit” button to finalize and create the account. If the account was able to be create, a display text will indicate that the account has been create. If the account was not able to be created, it will notify the user with a display text if a specific information has been not properly inputted.

A mobile app screen titled "Log In" with a blue background. It contains two input fields: "Email Address" and "Password". Below the fields is a red "Log In" button.

This page is displayed when the user clicks the “Log In” button on the first page that was displayed. The user can login to an existing account with their account email and password. If the account was able to be retrieved from the firebase with the authenticator and the user is logged in, a display text will indicate that the account was able to log in. If the user could not log into the account, it will notify the user with a display text if a specific information has been not properly inputted.



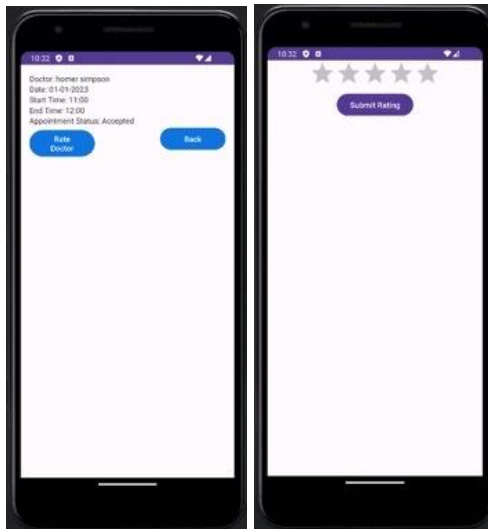
This page is displayed when a user logs into a patient account, where it welcomes the user by their name that is linked to their account, as well as the type of account they have logged into. Once this page is displayed, the user is able to access all the activities of a patient account. If the user wants to view all the appointments linked to the account, they can click the “Appointments” button. If the user would like to book an appointment, they can click the “Book Appointment” button. If the user would like to sign out of the account and return to the initial page, they can click the “Sign Out” button.



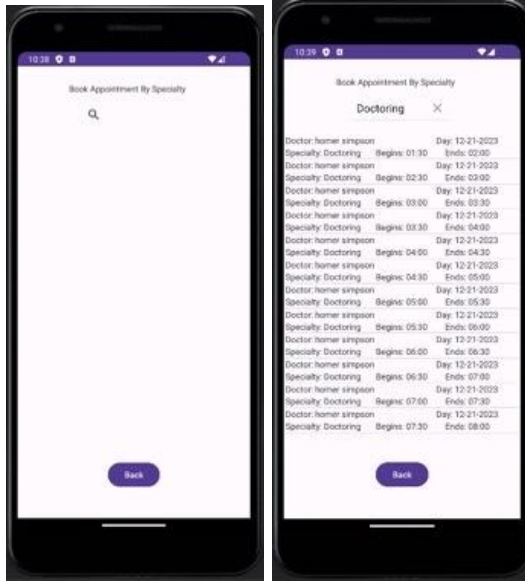
This page is displayed when the user of a patient account clicks the “Appointments” button. In this page, the user can access all the appointments that they have booked, which have not passed yet (by clicking on the “Upcoming Appointments” tab) and the ones that have already occurred (by clicking on the “Previous Appointments” tab). In both tabs of appointments, all the appointments linked to the account are listed displaying the information of that specific appointment, specifying the Patient that booked the appointment, the date of the appointment and the start and end time of the appointment. If the user clicks on a specific appointment in the list, it will display more information about the appointment. If the user wants to return to the welcome page, they can click the “Return” button.



This page is displayed when the user of a patient account clicks on a specific appointment in their list of upcoming appointments. In this page, the information of the clicked appointment is displayed, being the Doctor linked to the account, the date, start time and end time of the appointment and also the status of the appointment (if the doctor has accepted or rejected the appointment). If the user would like to cancel an upcoming appointment, the user can click on the “Cancel” button. If the user wants to return to the list of appointments page, they can click the “Back” button.



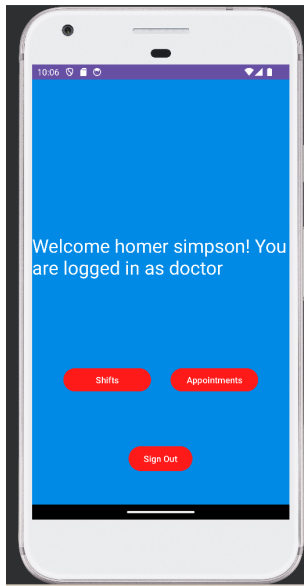
This page is displayed when the user of a patient account clicks on a specific appointment in their list of past appointments. In this page, the information of the clicked appointment is displayed, being the Doctor linked to the account, the date, start time and end time of the appointment and also the status of the appointment (if the doctor has accepted or rejected the appointment). If the user would like to rate the doctor linked to a past appointment, the user can click on the “Rate Doctor” button, where it sends them to the rating page (Screenshot on right), where the user can set a rating (out of 5 stars) and submit that rating with the “Submit Rating” button, which will then return them to the information page. If the user wants to return to the list of appointments page, they can click the “Back” button that is on the information page.



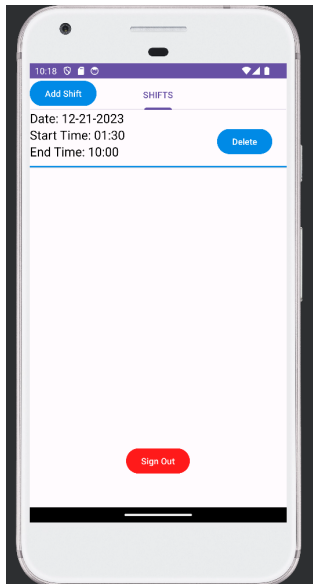
This page is displayed when the user of a patient account clicks on the “Book Appointment” button in the welcome page. In this page, the user can search for an available appointment by searching by a doctor specialty, which will then display a list of available appointments based on the shifts of the doctors, which are each a 30 minute period. For each available appointment, the doctor assigned to that potential appointment, the specialty desired, the date, the start time and the end time of the appointment. If the user wants to book or view more information on the appointment, they can click on a specific available appointment. If the user would like to go back to the welcome page, they can click on the “Back” button.



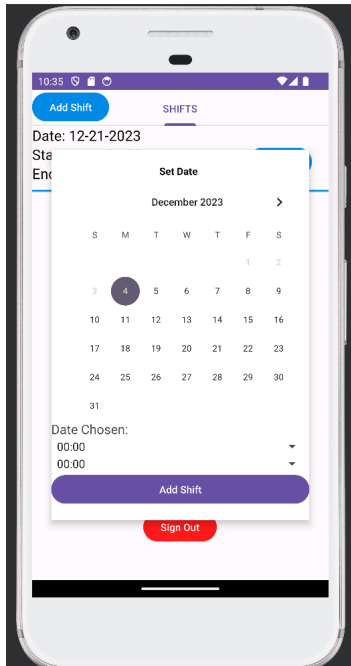
This page is displayed when the user of a patient account clicks on a specific appointment in the list of available appointments. In this page, the information of the clicked appointment is displayed, being the doctor assigned to that potential appointment, the specialty desired, the date, the start time and the end time of the appointment. If the user would like to book that available appointment, the user can click on the “Book” button. If the user wants to return to the search page for the available appointments, they can click the “Back” button.



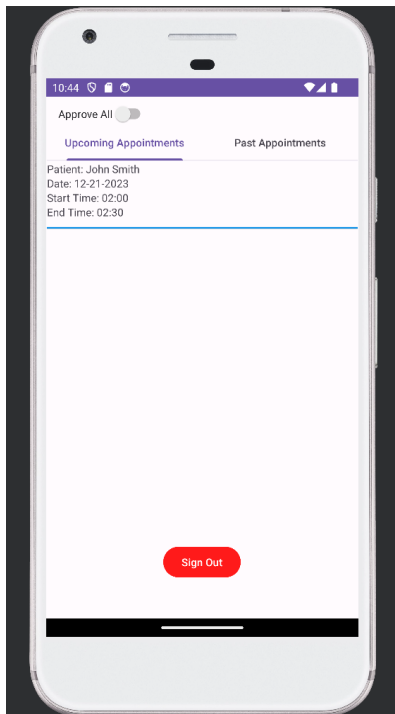
This is the welcome page activity for when a user of type doctor logs into the system. It displays a textview containing a welcome message, as well as the doctor's name (stored using a getter and setter in the doctor class.) This activity also includes 3 onClick buttons, the "Shifts" button taking the user to an activity displaying the shifts associated with the doctor object, the "Appointments" button taking the user to an activity displaying all upcoming and past appointments associated with that doctor object, and finally a "Sign Out" button, that will log the user out of the database.



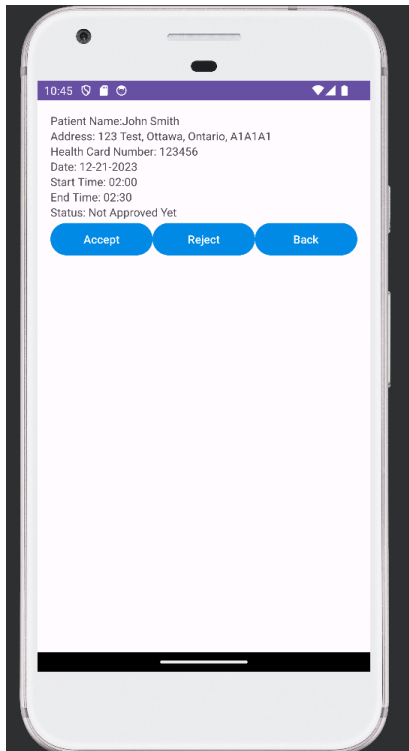
This is the "Shifts" activity, for when the user clicks the "Shifts" OnClick button from the welcome page. This displays a listview of all of the upcoming shifts the doctor user has, with 3 additional OnClick methods. One of the OnClicks allows the user to sign out, the second allows the user to delete a shift on the listview, and the third allows the user to add a shift to the listview and to the database. When clicking the "Delete" button, the shift associated with that button will be removed from the listview and the database, and accompanied with a toast indicating success or failure.



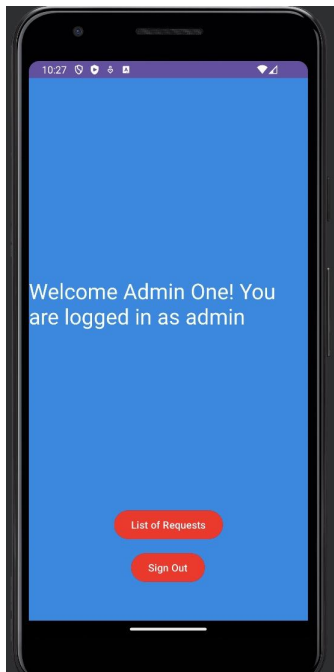
This is the view of the application when clicking the “Add Shift” button, displaying a calendar dropdown, a date display, 2 additional dropdowns, and a “Add Shift” button at the bottom of the dropdown. When the user selects a date on the calendar, that date will appear on the “Date Chosen” section. The two additional dropdowns allows the user to choose a start time and end time to the shift, in military time. Clicking “Add Shift” will add the created shift and its attributes to the previously displayed listview, and the database.



This page shows the upcoming appointments that exist, and also provides important patient information such as Name, Date of appointment, as well as Start and end time of the appointment itself.



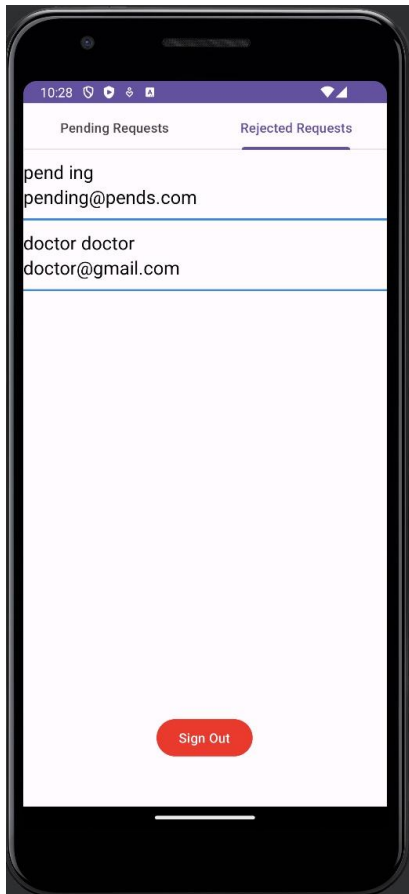
Upon clicking a listview instance of an upcoming appointment, it brings the doctor user to this activity, displaying the patient information from the associated patient object, including the name, address, health card number, and the upcoming appointment information. On top of the details of the appointment, it has 3 buttons that give the option to Accept or Reject the appointment. If the user chooses, they can also go back to the previous page in the application.



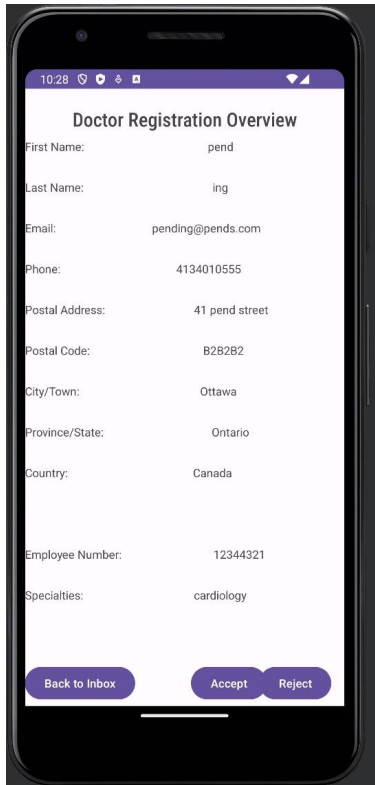
This is the login page activity for the admin. The text view in the middle features the admin's name and letting the admin know they are logged in as an admin. The admin is then able to either press a button to sign out or view the list of pending requests to accept or reject user types in the database (patient or doctor objects.)



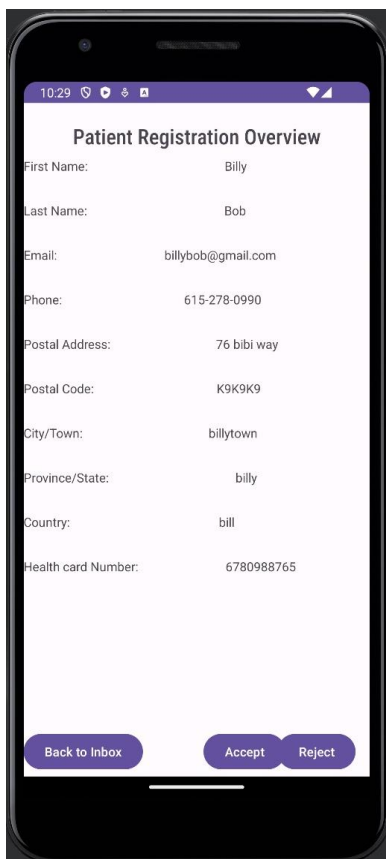
This is the admin inbox activity that displays a list of pending requests of newly registered users. It allows the Admin to click on a request and view the information to either approve or reject the user.



This is a screenshot that depicts the rejected requests of some users who attempted to create an account on the HAMS system, but was rejected for one reason or another. This is the view from the admin's side that allows them to approve or deny account requests.



This activity displays the user's information so the admin can choose to accept or reject the user based on this information. This example screenshot shows the information for a pending/rejected doctor user.



This activity displays the users information so the admin can choose to accept or reject the user based on this information. This example screenshot shows the information for a patient.

Lessons Learned:

- How to develop UML class diagrams for a practical application
- How to connect and manipulate databases (i.e. Firebase) for an Android project
- How to incorporate the usage of synchronous and asynchronous functions together
- How to effectively communicate with team members to troubleshoot and solve bugs encountered in our implementations
- How everyone must be responsible and well organized in order to complete their tasks for the set deadline of a deliverable
- Introduced to and learned the basics of a user interface system, and created a corresponding backend component to work together with the UI.
- How to expand our knowledge for coding concepts by doing research online or through discussing with other team members
- Adapt better coding habits, such as writing comments in our implementations to allow other team members to understand your code, which can help when debugging and solving issues

Resources Used:

-TabItems:

[How to get tab click event in activity on TabLayout android - Stack Overflow](#)

-RecyclerView

[\(833\) RecyclerView Android Studio Tutorial | 2023 - YouTube](#)

-Callbacks for async functions

[android - Return value from ValueEventListener java - Stack Overflow](#)

-Firebase Real-Time Database

[Read and Write Data on Android | Firebase Realtime Database \(google.com\)](#)

[Connect your App to Firebase | Firebase Realtime Database \(google.com\)](#)

[How to Retrieve Data from the Firebase Realtime Database in Android? - GeeksforGeeks](#)

[How to Save Data to the Firebase Realtime Database in Android? - GeeksforGeeks](#)

-Firebase authentication

[Authenticate with Firebase using Password-Based Accounts on Android | Firebase Authentication \(google.com\)](#)

-Calendar view

[Android | Creating a Calendar View app - GeeksforGeeks](#)

-Detect user input in real time

[android - Avoid EditText TextWatcher firing events on every character change - Stack Overflow](#)

-Setting minimum date in Calendar view

[android - CalendarView minimum date - Stack Overflow](#)

-Spinner for drop-down menu

[Spinner in Android with Example - GeeksforGeeks](#)

[Add spinners to your app | Android Developers](#)

-ListView

[Custom ArrayAdapter with ListView in Android - GeeksforGeeks](#)

-Time overlapping

[How to check a timeperiod is overlapping another time period in java - Stack Overflow](#)

-OnClickListener for request lists (clicking on a request to look at the info and to accept/reject

<https://www.youtube.com/watch?v=69C1ljfDvI0>