UOW MALAYSIA KDU

SCHOOL OF COMPUTING & CREATIVE MEDIA

Please tick ✓ or click if using MS WORD

☐FOUNDATION ☐DIPLOMA ☑ DEGREE ☐ MASTER

Assignment Coversheet

Please complete all details required clearly. For softcopy submissions, please ensure this cover sheet is included at the start of your document or in the file folder.

Assignment & Course Details:

Subject Code: *(e.g. XCAT1234)*	**Subject Name** *(e.g. Fundamentals of Computing)*:
XBMC3024 / N	**Mobile Programming and Screen Design 2**

Course *(e.g. Bachelor in Computing)* :
Bachelor of Computer Science (Hons)

Lecturer Name:
Ts. Dr. Law Foong Li

Assessment Due Date: *(dd/mm/yy)*	**11 August 2022**	**Assessment Title:**	**Mobile Programming and Screen Design 2 Documentation**

I/We declare that:

- *This assignment is my/our own original work, except where I/we have appropriately cited the original source.*
- *This assignment or parts of it has not previously been submitted for assessment in this or any other subject.*
- *I/We allow the assessor of this assignment to test any work submitted by me/us, using text comparison software for plagiarism.*
  (For more information, Please read the Academic Integrity Guidelines)

Name : Calvin Goh Kai Boon **Student ID:** 0132005 **Email :** 0132005@kdu-online.com **Mobile No:** **016-4845592** **Signature:** **Date:** Cgkb	**Name :** Jonathan Ng Zhu Xiang **Student ID:** 0132477 **Email :** 0132477@kdu-online.com **Mobile No:** **016-553 2492** **Signature:** **Date:** 11 Aug 2022 johnathan	**Name :** Abdul Rahim Bin Mohamed Suhaimi **Student ID:** 0131352 **Email :** 0131352@kdu-online.com **Mobile No:** **012-505 1965** **Signature:** **Date:** 11 Aug 2022 abdulrahim
Name : **Student ID:** **Email :** **Mobile No:** **Signature:** **Date:**	**Name :** **Student ID:** **Email :** **Mobile No:** **Signature:** **Date:**	**Name :** **Student ID:** **Email :** **Mobile No:** **Signature:** **Date:**

For office use only – Lecturer comments (if applicable)

Marks Breakdown

UOW
MALAYSIA
KDU

PART OF THE UNIVERSITY
OF WOLLONGONG AUSTRALIA
GLOBAL NETWORK

Bachelor of Computer Science (Hons)

XBMC3024 Mobile Programming and Screen Design2

Ts. Dr Law Foong Li

Prepared By :

Calvin Goh Kai Boon 0132005

Jonathan Ng Zhu Xiang0132477

Abdul Rahim Bin Mohamed Suhaimi 0131352

Table of Contents

1.0 Introduction

In this day and age, with the prevalence of smartphones, everything we want to search for or need to know is at our fingertips. Smartphones have also allowed us to constantly track our health with built in applications. Therefore, for this project we would like to talk about our own application which also has the function of tracking our health and the ability to make appointments with associated clinics and hospitals that have collaborated with our application. The aim of our application is to make an application that fits an all-in-one system where people can do everything using it. For example, the ability to check their health, the ability to make appointments immediately and have consultations with doctors though the app itself for a diagnosis. With this new application, people can participate in their own health care seamlessly and effortlessly with everything located in just one place.

According to Pew Research Centre 97% of Americans owned a cell phone (2021) and that those that own a smartphone is 85% which is a huge leap compared to Pew Research Centre's first survey in 2011 where only 35% of Americans owned a smartphone. This is evident in surveys of healthcare professionals where many admit they own a smartphone which is used in both clinical practice and education (Aungst, 2013). One of the most important reasons why healthcare professionals use smartphones is because it allows for better communication and information resources (Mosa, Yoo and Sheets, 2012). On the average person's side however, it would lean more towards the fact that there have been rapid advancements in technology, which allows applications to have many more complex functions to accomplish a specific purpose thus leading to a rise in mobile device applications such as medical applications. Users now have a variety of applications to choose from catering to any of their needs (Mosa, Yoo and Sheets, 2012). Apart from that, smartphone compatible devices such blood pressure cuffs, and pulse oximeters can be found easily in a normal person's house (Batista and Gaglani, 2013). These smartphone compatible devices typically have a friendly user interface that are intuitive and provide health data which allows users to take their own healthcare into their own hands. They can also summarise the data into graphs that display the data in a concise form. Unfortunately, even with all these advances, they all still lack something, which is the ability to schedule/cancel an appointment, the ability to request prescriptions and easy access to medical records and only 11% of hospital apps offer one of those functionalities (georgiou, 2022).

1.1 Hypothesis & Objectives

Our hypothesis is will more people use our app if we were to offer functionalities like ability to schedule/cancel appointments, the ability to request prescriptions and easy access to medical records. Therefore, we aim to implement these functions into our application in a friendly user interface so that the user will have an easier time using our application. Delving into the objectives of developing the application and it is significance in the current day and age is to provide convenient access to medical services, the main focus being the ability for patients to book an appointment online with any type of medical professional to perform virtual consultations, with the assistance of modern day mobile technology at a push of a button; to add, it also aims to increase the exposure of people to essential medical knowledge and awareness.

2.0 Methodology

2.1 Tools Used

2.1.1 Android Studio

Figure 1.1

Android Studio is the official integrated development environment (IDE) for Android application development. Android Studio uses a Gradle-based build system, emulator, has code templates as well as a GitHub integration. It incorporates Java to edit codes

2.1.2 Java

Figure 1.2

Java is an object-oriented programming language released by Sun Microsystems in 1995. Now, 3 billion devices run Java (w3school, n.d.). Java is widely used because Java has provided constant testing, updating and consistency for as long as it has been around. Not only that, Java has also constantly been maintained and tested by a community of Java developers

2.1.3 Firebase

Figure 1.3

Firebase is Google's mobile application development that is used to help in the development of an application. What Firebase does is that it helps developers out by taking care of many services that developers would normally have to build themselves. For example, authentication services, databases, and file storages. All these services are hosted online

2.1.4 GitHub

Figure 1.4

GitHub is a website that allows developers to store and manage their code online. GitHub also allows developers to track and control changes to their code. GitHub comprises two connected principles, version control and git. Version control is the part that helps developers work through branching and merging. Branching is when the code is duplicated where changes to code can be made without affecting the main code. Merging is when the developer submits the code back into the main code. The next principle git is the distributed version control system where the entire code is available on every developers computer

2.1.5 Figma

Figure 1.5

Figma is a UI and UX design application to create websites or applications. Figma is also a cloud-based design application tool that can be accessed by separate users to work on projects together whenever and wherever. Figma also has many features and services included such as Vector Networks. These functionalities give users the chance to do whatever they need to do.

2.2 Techniques and Methodologies Used

2.2.1 Waterfall Methodology

The methodology that we implemented was the Waterfall Methodology. The Waterfall methodology has a chronological process and works based on fixed dates, requirements and outcomes. In this methodology, each phase must be completed fully before being able to move onto the next phase.

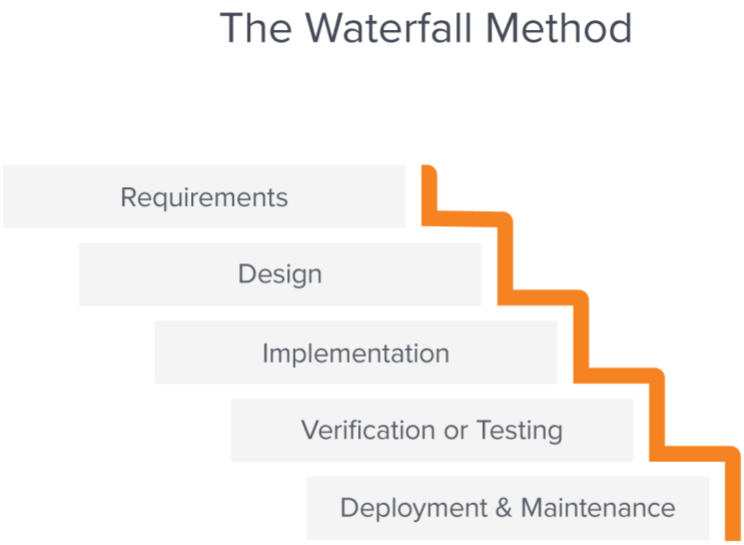


Figure 2.1 Waterfall Method

As seen in the picture above, before we started anything we had to complete the requirements phase. Here, we established whether all the project requirements could be gathered upfront where we did our best to understand what the assignment required.

After completing the requirements phase, we moved onto the design phase whereby we used a collaborative interface design tool known as Figma to come up with our mobile app designs. We spent a few days coming up with the design in order to come up with something because we had to come up with something that would fulfil the scope of the project. We also had to create scenarios and layouts of the project and how each component of the app would flow. After we all mutually agreed on the design, we moved onto the implementation phase.

In the implementation phase, we decided to split the project into smaller sections for each of us to complete as well as a due date for each of us to submit our work. Here, we coded the application based on the requirements and specifications. When we were done, we had to verify that the application was working meaning we had to conduct several instances of testing to ensure it

worked. If there were any changes, we would have had to go back to the design phase to do it properly.

Lastly, for the deployment and maintenance phase is when we would submit this application to the lecturer as it is already completed.

2.2.2 Data Binding Library

Data binding is the act of integrating data objects with views in an XML layout. The Data Binding Libraries responsibility is to generate the necessary classes for this method (Pandya, 2021). Data Binding-layout XML files, unlike other types of layout XML files, begin with a root layout tag followed by a data element. Then, each layout file is coupled with a Data Binding class created by the library (Android Developers, 2022b).

2.2.3 Fragment Navigation

Android 3.0 (API level 11) added fragments to facilitate more dynamic and adaptable UI designs on big displays, such as tablets. Because a tablet's screen is far more extensive than a smartphone's, there is more area to mix and swap UI elements. Fragments enable such designs without requiring the management of extensive modifications to the view hierarchy. By breaking an activity's layout into pieces, we may edit the activity's appearance at runtime and save those changes in a back stack maintained by the activity (Android Developers, 2018).

2.2.4 Hashmap

It is a data structure used to organise data such that instead of mapping index to values as in an array, keys are used to pinpoint each row of data in memory, supported with the use of a hash function to connect the key to the location of the memory; therefore, this enables for random insertion of data into the data structure as opposed to an array which requires insertion according to positions within the array indicated by its index (Android Developers, 2022).

2.3 Prototype Design

The Figures below shows the high-fidelity prototype we designed before carrying out the development process.

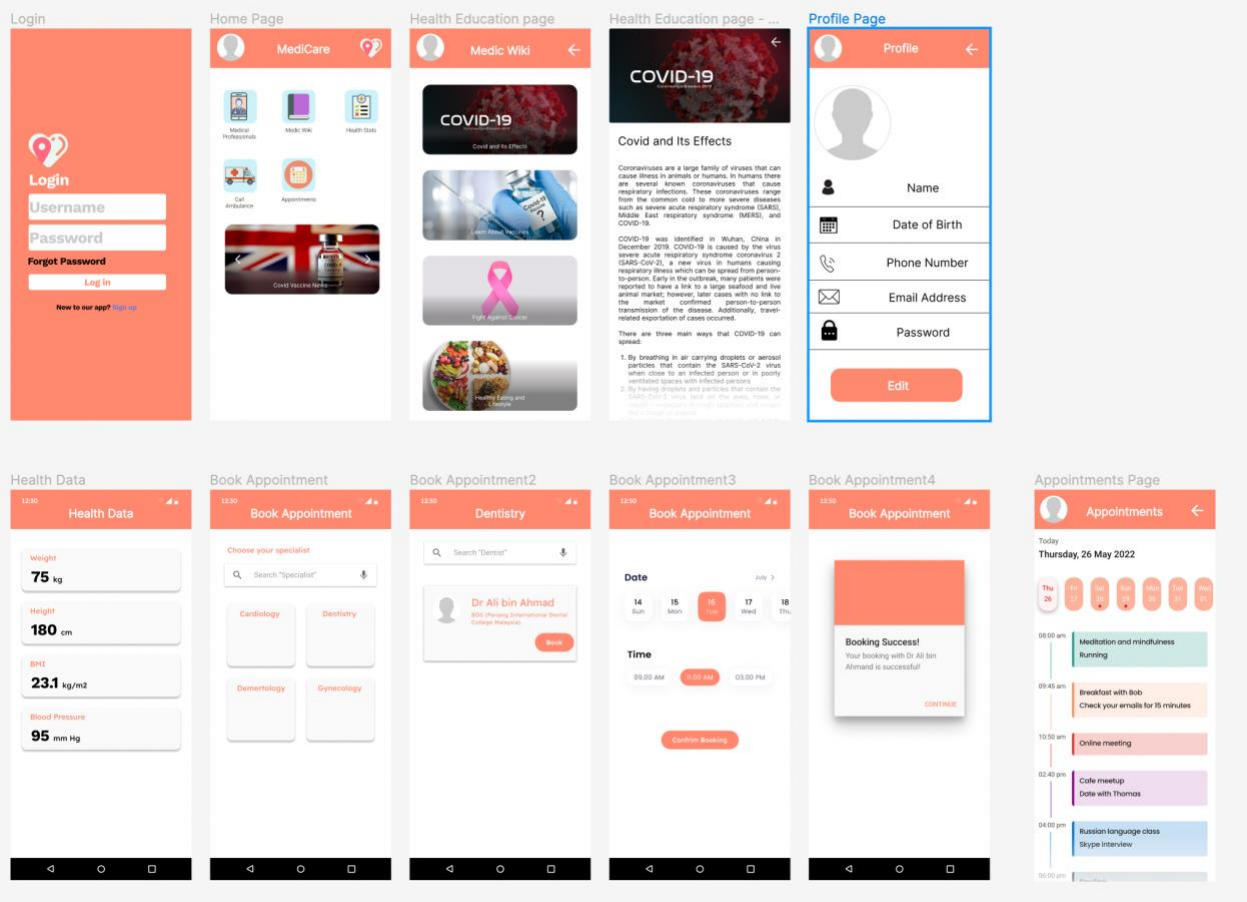


Figure 3.1

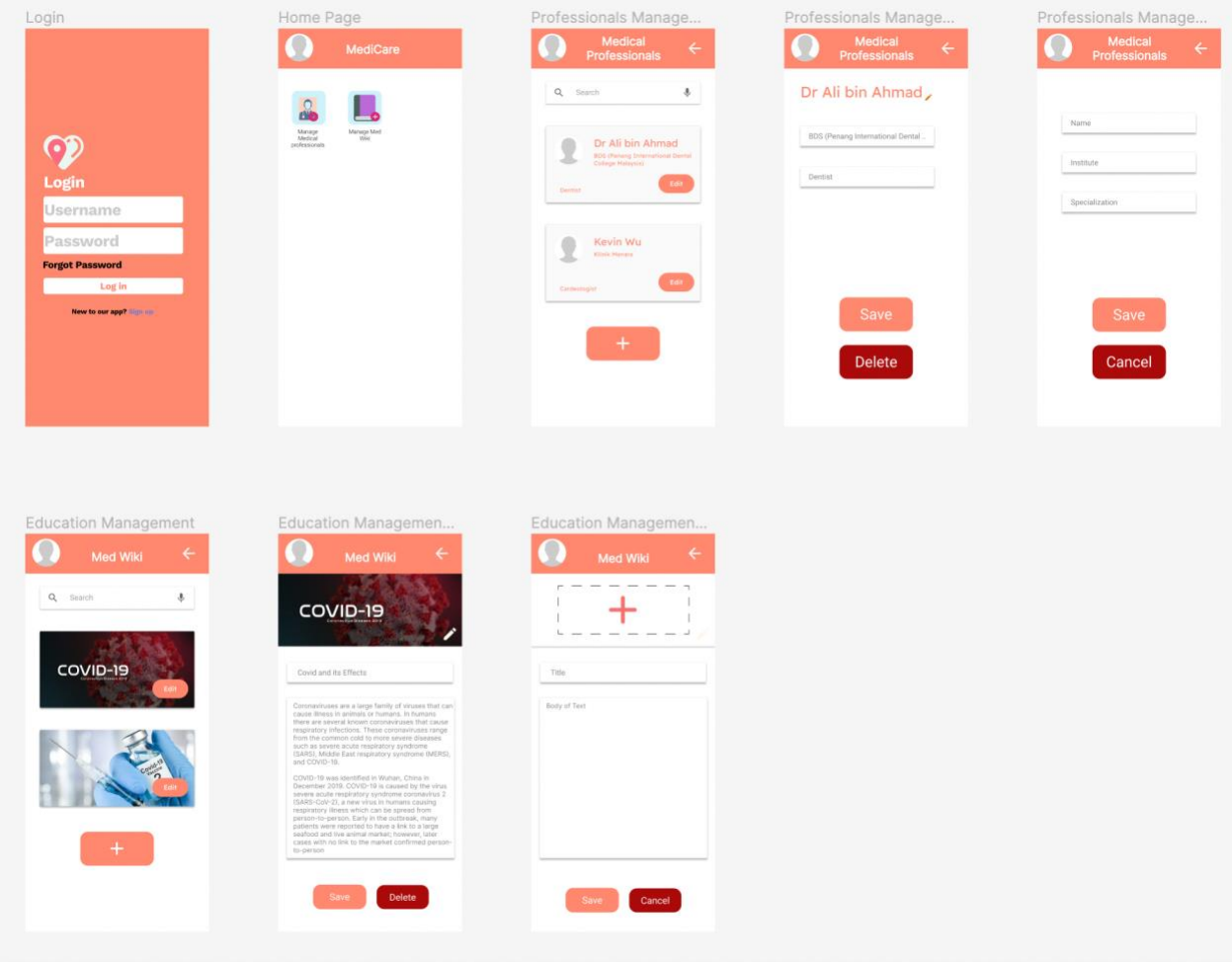

Login
Login
Username
Password
Forgot Password
Log in
New to our app? Sign up
Home Page
MediCare
Professionals Manage...
Medical Professionals
Search
Dr Ali bin Ahmad
Edit
Kevin Wu
Edit
Professionals Manage...
Medical Professionals
Dr Ali bin Ahmad
BDS (Penang International Dental ..
Dentist
Save
Delete
Professionals Manage...
Medical Professionals
Name
Institute
Specialization
Save
Cancel
Education Management
Med Wiki
Search
COVID-19
Edit
Edit
Education Managemen...
Med Wiki
COVID-19
Covid and its Effects
Save
Delete
Education Managemen...
Med Wiki
Title
Body of Text
Save
Cancel

Figure 3.2

3.0 Results

The purpose of our study of the problem is to determine whether more people would use our app if we were to offer functionalities like ability to schedule/cancel appointments, the ability to request prescriptions and easy access to medical records. Thus, we have compiled the functionalities of our application with a breakdown on what each page does

3.1 Functionalities

3.1.1 Admin

3.1.1.1 Home

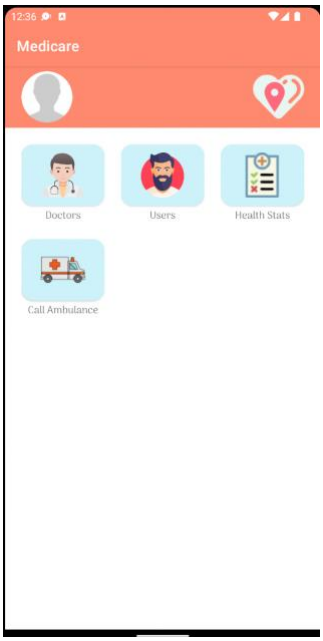


Figure 4.1

As the admin, when they access the main menu, they will see 4 options which are doctors, users, health stats and call an ambulance. Administrators will be able to select whichever button to conduct its corresponding activity

3.1.1.2 Add Doctors

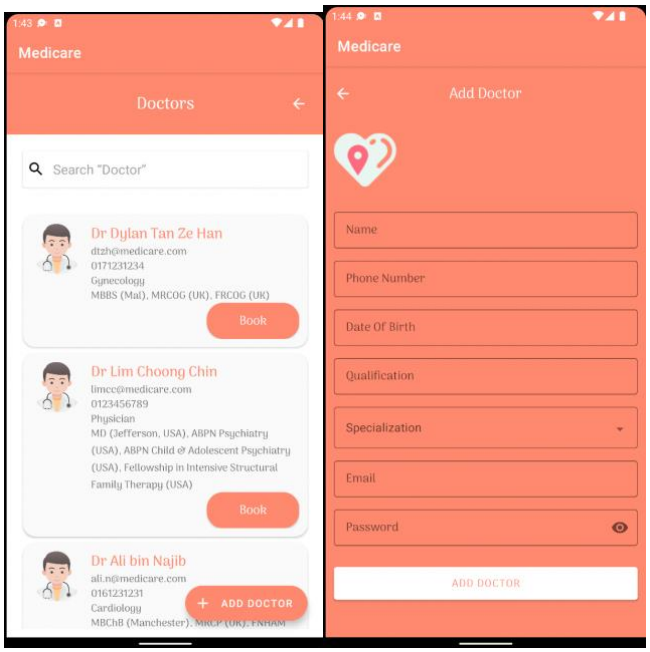


Figure 4.2

Administrators have the ability to add and register new doctors to the database. They can view all doctors that are registered into the system. In the other screen, they add doctors to the database and must include important things such as name, qualifications and specialisations.

3.1.1.3 Manage Users

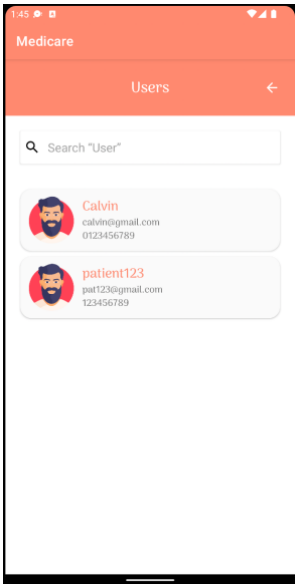


Figure 4.3

Next, administrators have the ability to search for any particular user which also displays all the user details

3.1.2 Users

3.1.2.1 Home

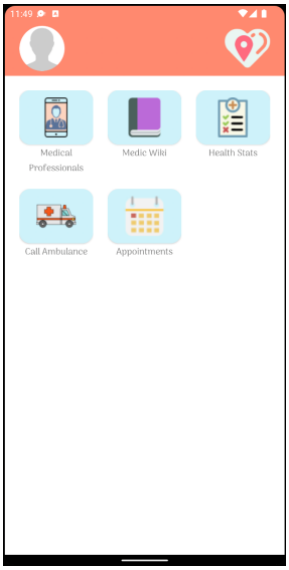


Figure 4.4

As the user, when we access the homepage, we have access to 5 options, which are medical professionals, medic wiki, health stats, call ambulance and appointments. Users will be able to click on any one of the options to conduct its corresponding activity

3.1.2.2 Medical Professionals

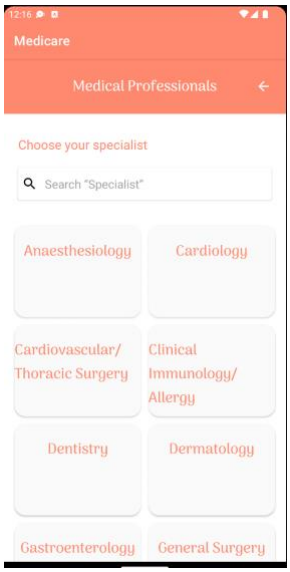


Figure 4.5

In the medical professionals' option, users will be able to search for any specialist they wish to see. In the image above, users can also scroll through the numbers of specialisations if they are unsure onto what they are looking for

3.1.2.3 Health Data

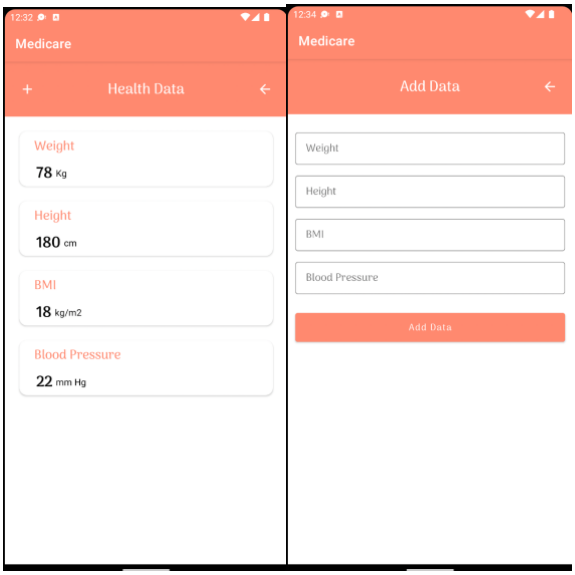
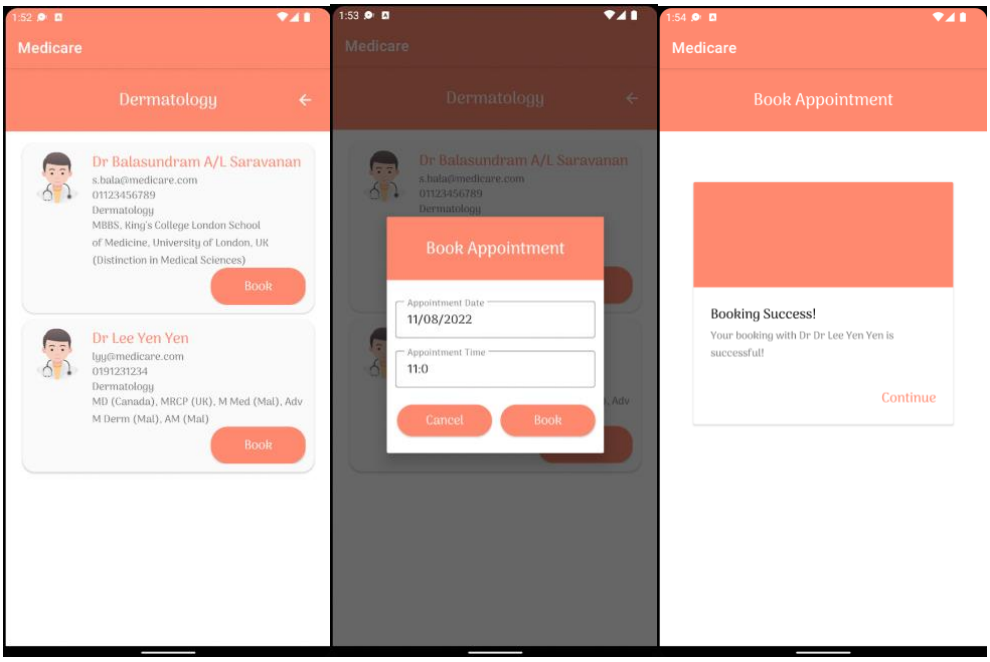


Figure 4.6

In the health data tab, users will be able to enter their weight, height, body mass index (BMI) and blood pressure. They can constantly track their progression this way

3.1.2.4 Book Appointment



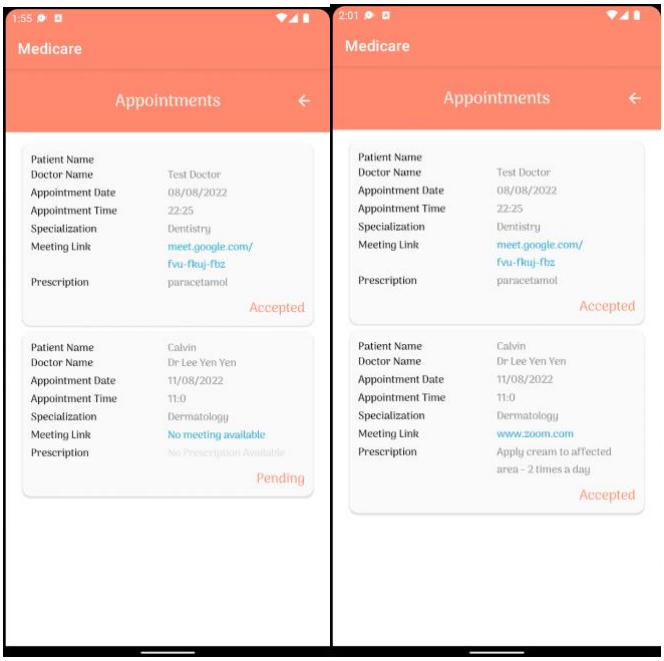


Figure 4.6

Users will be able to book appointments using our application with our built-in appointment registration system. First, from the medical professionals' page, they will select a specialisation. For example, dermatology. After picking dermatology, they will see a list of doctors who specialise in that particular field. Users will pick a doctor they wish to see and will then pick a time and date for their appointment. Patients will be informed that their appointment was registered successfully. Later, they can check on all their appointments which will specify important things such as appointment date and time as well as a link to an online meeting for their appointment. After the appointment, their prescription will be updated to the given prescription by the doctor along with the number of times they need to take it

3.1.3 Doctors

3.1.3.1 Home

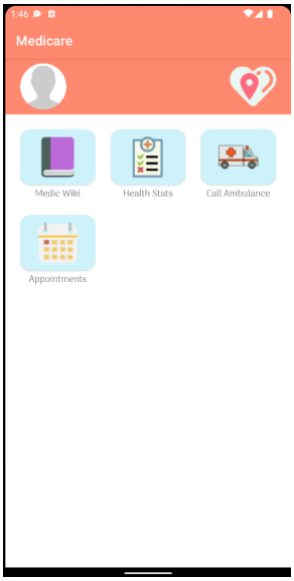


Figure 4.7

For the doctors, when they enter the main menu, they will also have 4 options which are medic wiki, health stats, call ambulance and appointments.

3.1.3.2 Appointment

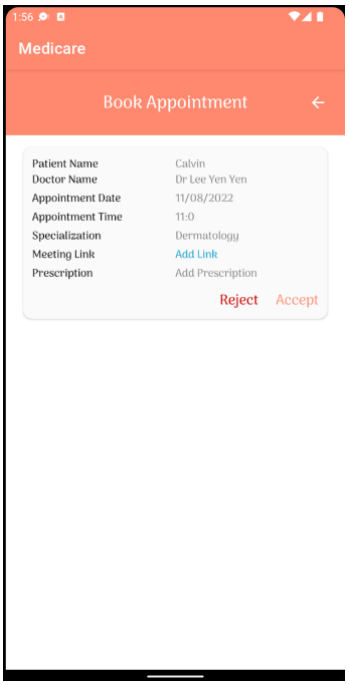


Figure 4.8

When the doctor clicks on the appointments option, it will bring him to this page where they can view any appointment that a user has made with them. The doctor will be able to reject or accept the appointment if it clashes with another appointment. The appointment card will also list details such as patient name

3.1.3.3 Create Virtual Consultation

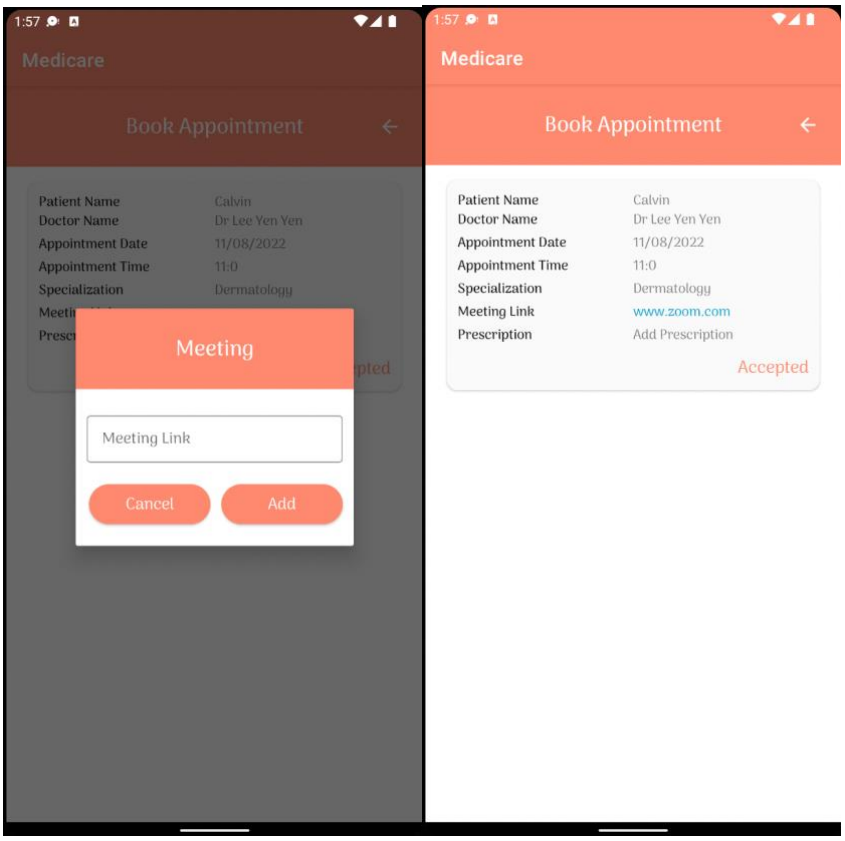


Figure 4.9

If the doctor does accept the appointment booking, a popup will be displayed in front of them where they have to add a meeting link for the patient to use. After adding the meeting link, the appointment card will change to an accepted status to show the doctor their accepted appointments

3.1.3.4 Prescriptions

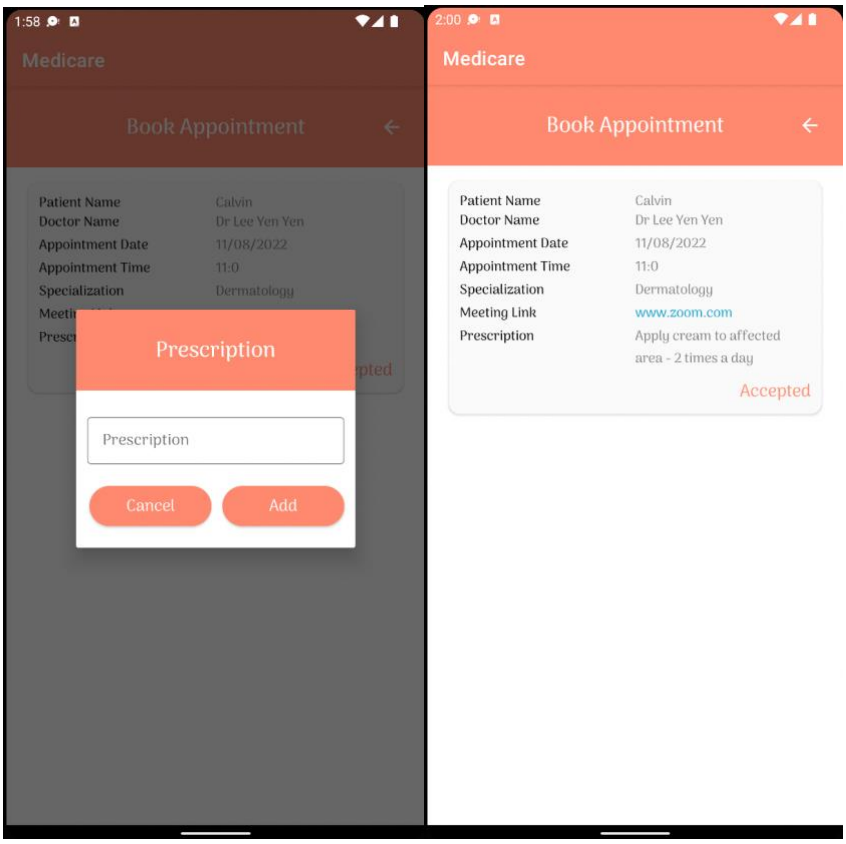


Figure 4.10

After the appointment and diagnosing the patient, the doctor will give a prescription to the page. Once again, a popup will be displayed in front of the doctor where he will enter the prescription. After adding it, the appointment card will be update with the prescription so that the patient will be reminded of the prescription whenever they check the application

3.1.4 Common Functions

3.1.4.1 Login

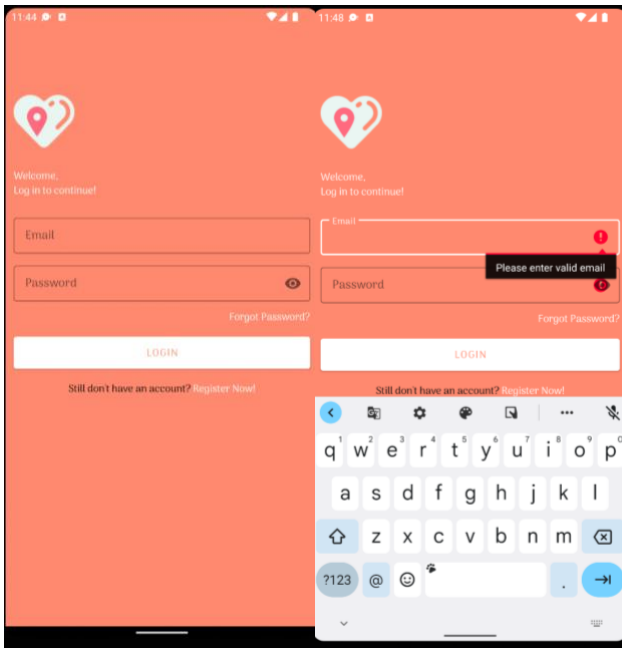


Figure 4.11

This is the login page for the application where users have to input email and password. If the user fails to enter an email or password, they will be alerted to enter an email. The login screen also allows users to view their password in case they mistyped something

3.1.4.2 Profile

Figure 4.12

The profile page for the user. Users will enter their name, phone number, date of birth and their email which will all be used when making an appointment with a doctor

3.1.4.3 MedicWiki

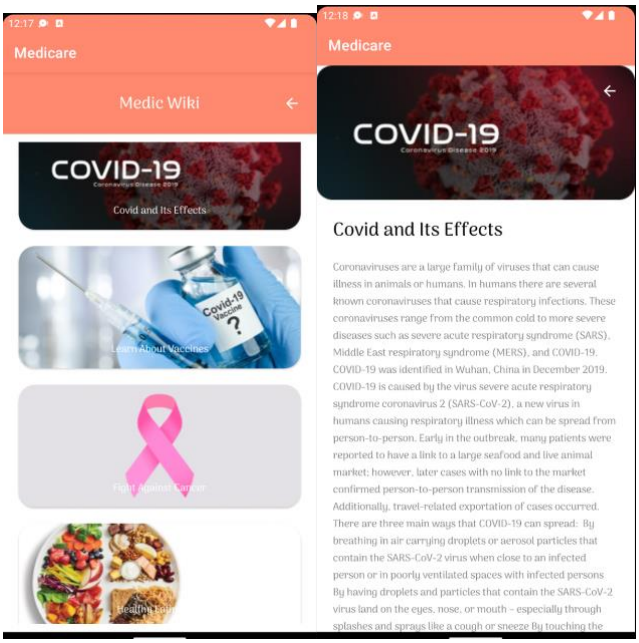


Figure 4.13

Users can also check on the medical wiki page where it displays things like the current Covid-19 situation along with its effects, vaccinations, or healthy eating. Users can look for self-improvement options here

3.1.4.4 Emergency Call

Figure 4.14

Users can register a number for an emergency call as well as call emergency services

3.1.4.5 Loading Screen

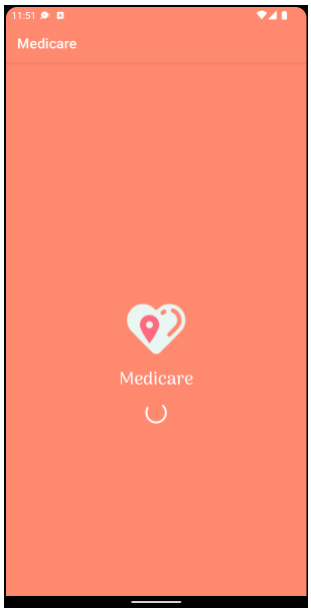


Figure 4.15

A simple loading screen to indicate when a page is loading.

Summary

Our applications main features are its easy-to-use functionalities on the users, doctors and administrators sides where everything is located in one place. The application also features a very clean design that makes navigating through easy and convenient to use. Along with its ability to track the user's health, users will be inclined to use the features provided by our application

4.0 Implementation

With regards to development, we used Android Studio for this project. We created a GitHub project and then imported the project to our Android Studio so that we could share any changes among us seamlessly and easily. This way, any changes made would also be documented.

4.1 Design

4.1.1 Edit Text View

Figure 5.1

The Edit Text View is used wherever an editable text field is required such as in the case of the Registration or Login pages.

4.1.2 Text View

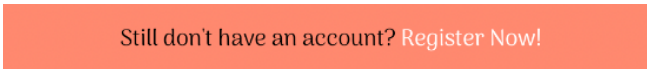


Figure 5.2

The normal Text View is used to display a static text such as the one shown in the figure above, this can range from a short text to label an icon or even a whole paragraph of text to convey a message to the user. This component is used throughout the application

4.1.3 RecyclerView

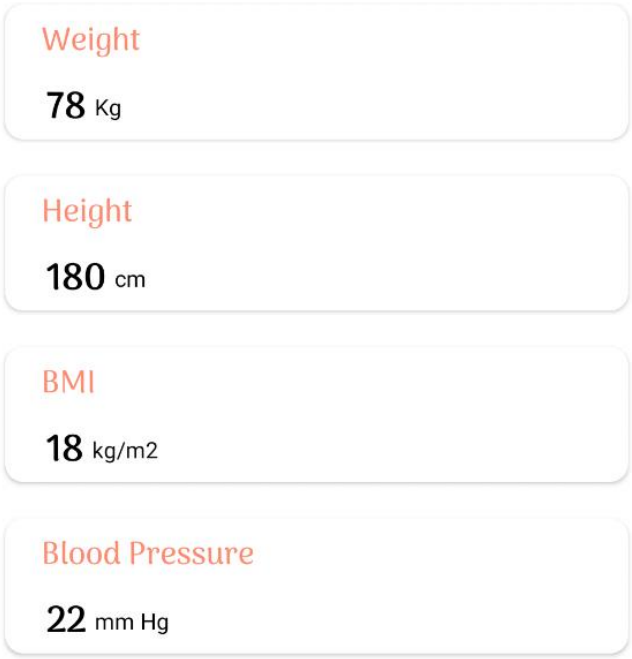


Figure 5.3

The RecyclerView is basically an extension of 2 existing views available in the android native development environment, ListView and GridView, however it provides the ability to react dynamically to changes in the list. Hence when an element is added or moved in the list it will reflect in the RecyclerView; such a component is used in the Health Data page.

4.1.4 Card View

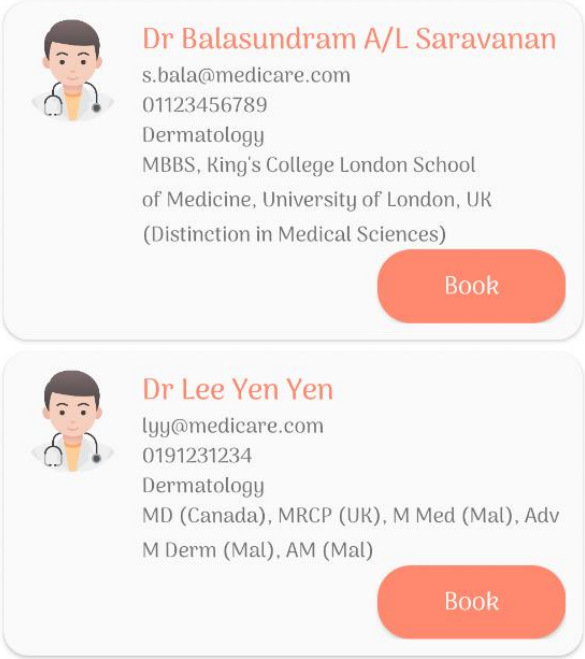


Figure 5.4

The CardView allows the developer to organise information for respective medical professionals into cards for easier browsing and searching by the patient user; such a component is mostly used in the medical professionals' page.

4.1.5 Image View

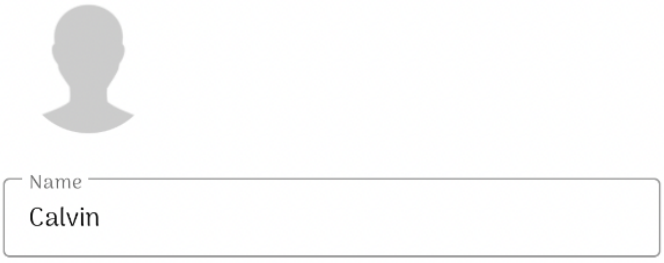


Figure 5.5

The ImageView is used to display images in any area of the pages by referencing the image source to a png or jpg; and this component can be seen almost throughout the app such as in the Edu Wiki Page and Medical Professionals page.

4.1.6 Button

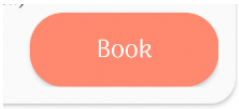


Figure 5.6

The button allows the user of the app to invoke a function from the app such as redirecting to another page or submitting a request, i.e. when booking an appointment in the Medical professionals page.

4.1.7 Floating Button

Figure 5.7

Serves the same purpose as the button component but is given the attribute of floating over other elements within the page such as can be seen in the admin's side Doctors page.

4.1.8 Dialog

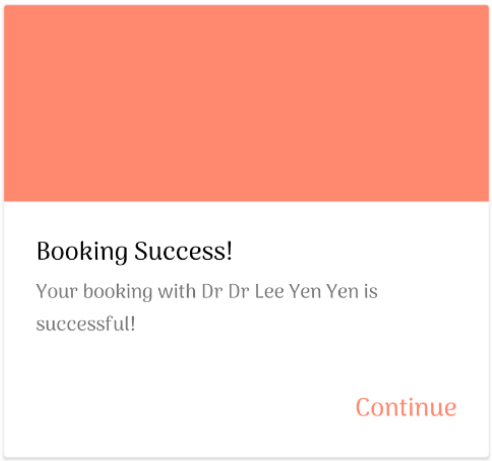


Figure 5.8

The dialog box is basically a component to prompt the user with some message as can be seen in the page after the patient user has clicked on any of the “Book” buttons.

4.1.9 Progress Indicator

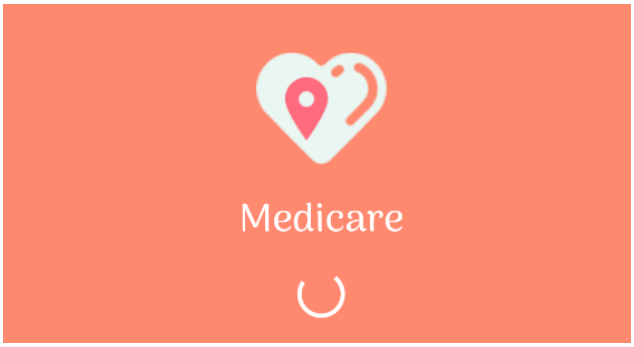


Figure 5.9

The indicator is to indicate to the user that the app is still loading so that the user knows whether the app is still running as intended while trying to load a page.

4.1.10 Calendar

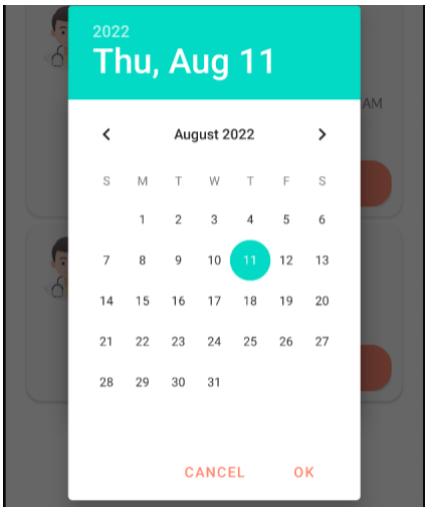


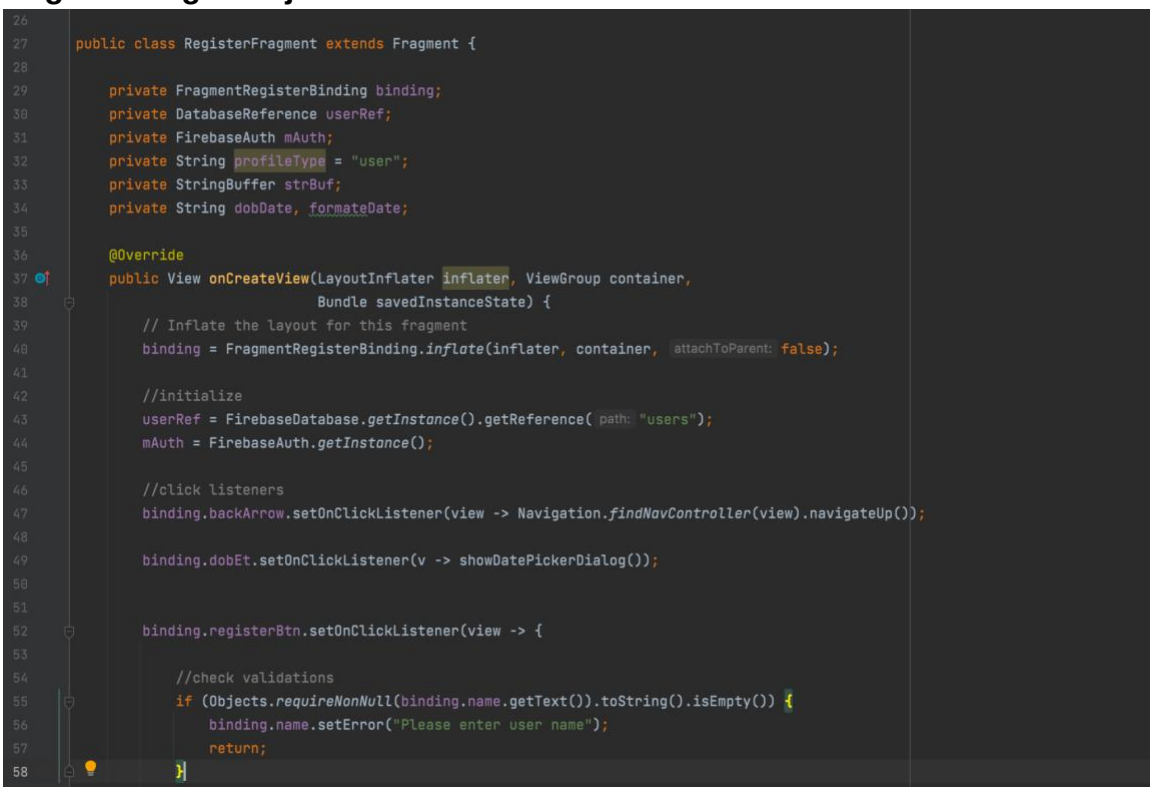
Figure 5.10

The calendar component basically facilitates the action of picking a date so that the user may have a more intuitive and efficient user experience; this component is used in the Booking page when the patient user is about to book an appointment with any medical professional.

4.2 Coding/Development

4.2.1 Register User Account

RegisterFragment.java

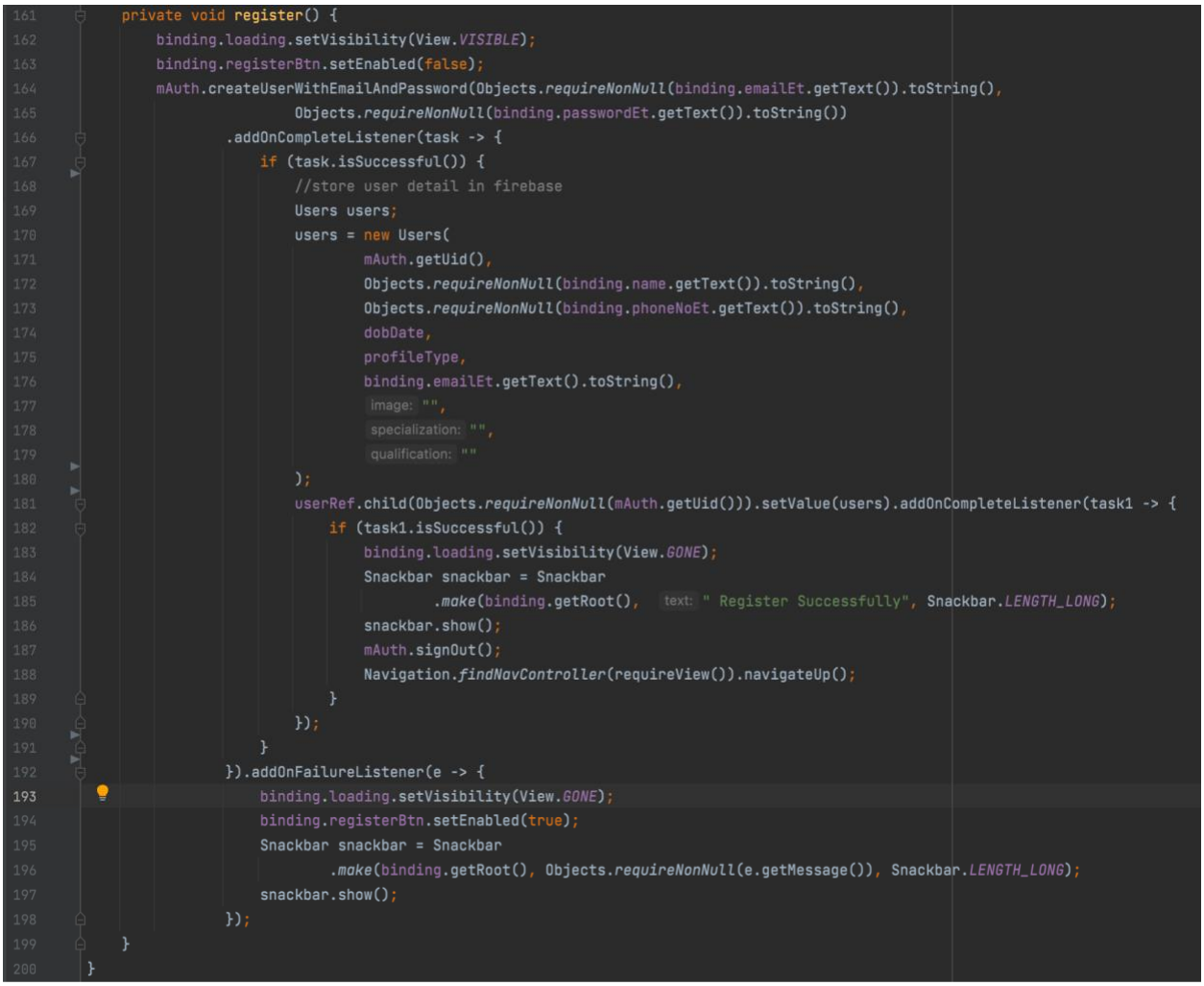


```
public class RegisterFragment extends Fragment {

    private FragmentRegisterBinding binding;
    private DatabaseReference userRef;
    private FirebaseAuth mAuth;
    private String profileType = "user";
    private StringBuffer strBuf;
    private String dobDate, formateDate;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        binding = FragmentRegisterBinding.inflate(inflater, container, attachToParent: false);

        //initialize
        userRef = FirebaseDatabase.getInstance().getReference( path: "users");
        mAuth = FirebaseAuth.getInstance();

        //click listeners
        binding.backArrow.setOnClickListener(view -> Navigation.findNavController(view).navigateUp());

        binding.dobEt.setOnClickListener(v -> showDatePickerDialog());


        binding.registerBtn.setOnClickListener(view -> {

            //check validations
            if (Objects.requireNonNull(binding.name.getText()).toString().isEmpty()) {
                binding.name.setError("Please enter user name");
                return;
            }
```

```
            if (Objects.requireNonNull(binding.emailEt.getText()).toString().isEmpty()) {
                binding.emailEt.setError("Please enter valid email");
                return;
            }
            if (Objects.requireNonNull(binding.passwordEt.getText()).toString().isEmpty()) {
                binding.passwordEt.setError("Please enter password");
                return;
            }

            if (Objects.requireNonNull(binding.dobEt.getText()).toString().isEmpty()) {
                binding.dobEt.setError("Please select date of birth");
                return;
            }
            //register user
            register();
        });
        return binding.getRoot();
    }


    private void showDatePickerDialog() {
        // Get open DatePickerDialog button.
        // Create a new OnDateSetListener instance. This listener will be invoked when user click ok button in DatePickerDialog.
        DatePickerDialog.OnDateSetListener onDateSetListener = (datePicker, year, month, dayOfMonth) -> {
            strBuf = new StringBuffer();
            // strBuf.append("You select date is ");


            if ((month + 1) < 10) {
                strBuf.append("0" + (month + 1));
            } else {
                strBuf.append(month + 1);
            }
```

```
        DatePickerDialog datePickerDialog = new DatePickerDialog(requireContext(), onDateSetListener, year, month, day);
        now.add(Calendar.YEAR,  amount: -18);

        datePickerDialog.getDatePicker().setMaxDate(now.getTimeInMillis());
        datePickerDialog.show();
    }

    //to get date of birth
@   private String getAge(int year, int month, int day) {

        Calendar dob = Calendar.getInstance();
        Calendar today = Calendar.getInstance();
        int todaymonth = today.get(Calendar.MONTH);
        int todayyear = today.get(Calendar.YEAR);
        int today_date = today.get(Calendar.DAY_OF_MONTH);
        today.set(todayyear,  month: todaymonth + 1,  date: today_date + 1);
        dob.set(year, month, day);
        int age = today.get(Calendar.YEAR) - dob.get(Calendar.YEAR);

        System.out.println("DateFormat1 " + today.get(Calendar.YEAR) + "---" + dob.get(Calendar.YEAR));
        System.out.println("DateFormat2 " + today.get(Calendar.DAY_OF_YEAR) + "---" + dob.get(Calendar.DAY_OF_YEAR));

        if (today.get(Calendar.DAY_OF_YEAR) < dob.get(Calendar.DAY_OF_YEAR)) {
            age--;
        }

        Integer ageInt = new Integer(age);
        String ageS = ageInt.toString();
        return ageS;
    }
```

```
96          strBuf.append("-");
97
98          if (dayOfMonth < 10) {
99              strBuf.append("0" + dayOfMonth);
100         } else {
101             strBuf.append(dayOfMonth);
102         }
103         strBuf.append("-");
104
105         strBuf.append(year);
106
107         String ages = getAge(year,  month: month + 1, dayOfMonth);
108
109         dobDate = strBuf.toString();
110
111         try {
112             DateFormat inputFormat = new SimpleDateFormat( pattern: "MM-dd-yyyy");
113             DateFormat outputFormat = new SimpleDateFormat( pattern: "MMM-dd-yyyy");
114             String inputDateStr = strBuf.toString();
115             Date date = inputFormat.parse(inputDateStr);
116             formateDate = outputFormat.format(date);
117         } catch (ParseException e) {
118             e.printStackTrace();
119         }
120
121
122         binding.dobEt.setText(ages + " " + "Years" + " / " + formateDate);
123     };
124     // Get current year, month and day.
125     Calendar now = Calendar.getInstance();
126     int year = now.get(Calendar.YEAR);
127     int month = now.get(Calendar.MONTH);
128     int day = now.get(Calendar.DAY_OF_MONTH);
```



```
161    private void register() {
162        binding.loading.setVisibility(View.VISIBLE);
163        binding.registerBtn.setEnabled(false);
164        mAuth.createUserWithEmailAndPassword(Objects.requireNonNull(binding.emailEt.getText()).toString(),
165                        Objects.requireNonNull(binding.passwordEt.getText()).toString())
166                .addOnCompleteListener(task -> {
167                    if (task.isSuccessful()) {
168                        //store user detail in firebase
169                        Users users;
170                        users = new Users(
171                                mAuth.getUid(),
172                                Objects.requireNonNull(binding.name.getText()).toString(),
173                                Objects.requireNonNull(binding.phoneNoEt.getText()).toString(),
174                                dobDate,
175                                profileType,
176                                binding.emailEt.getText().toString(),
177                                 image: "",
178                                 specialization: "",
179                                 qualification: ""
180                        );
181                        userRef.child(Objects.requireNonNull(mAuth.getUid())).setValue(users).addOnCompleteListener(task1 -> {
182                            if (task1.isSuccessful()) {
183                                binding.loading.setVisibility(View.GONE);
184                                Snackbar snackbar = Snackbar
185                                        .make(binding.getRoot(),  text: " Register Successfully", Snackbar.LENGTH_LONG);
186                                snackbar.show();
187                                mAuth.signOut();
188                                Navigation.findNavController(requireView()).navigateUp();
189                            }
190                        });
191                    }
192                }).addOnFailureListener(e -> {
193                    binding.loading.setVisibility(View.GONE);
194                    binding.registerBtn.setEnabled(true);
195                    Snackbar snackbar = Snackbar
196                            .make(binding.getRoot(), Objects.requireNonNull(e.getMessage()), Snackbar.LENGTH_LONG);
197                    snackbar.show();
198                });
199    }
200 }
```

Figure 6.1

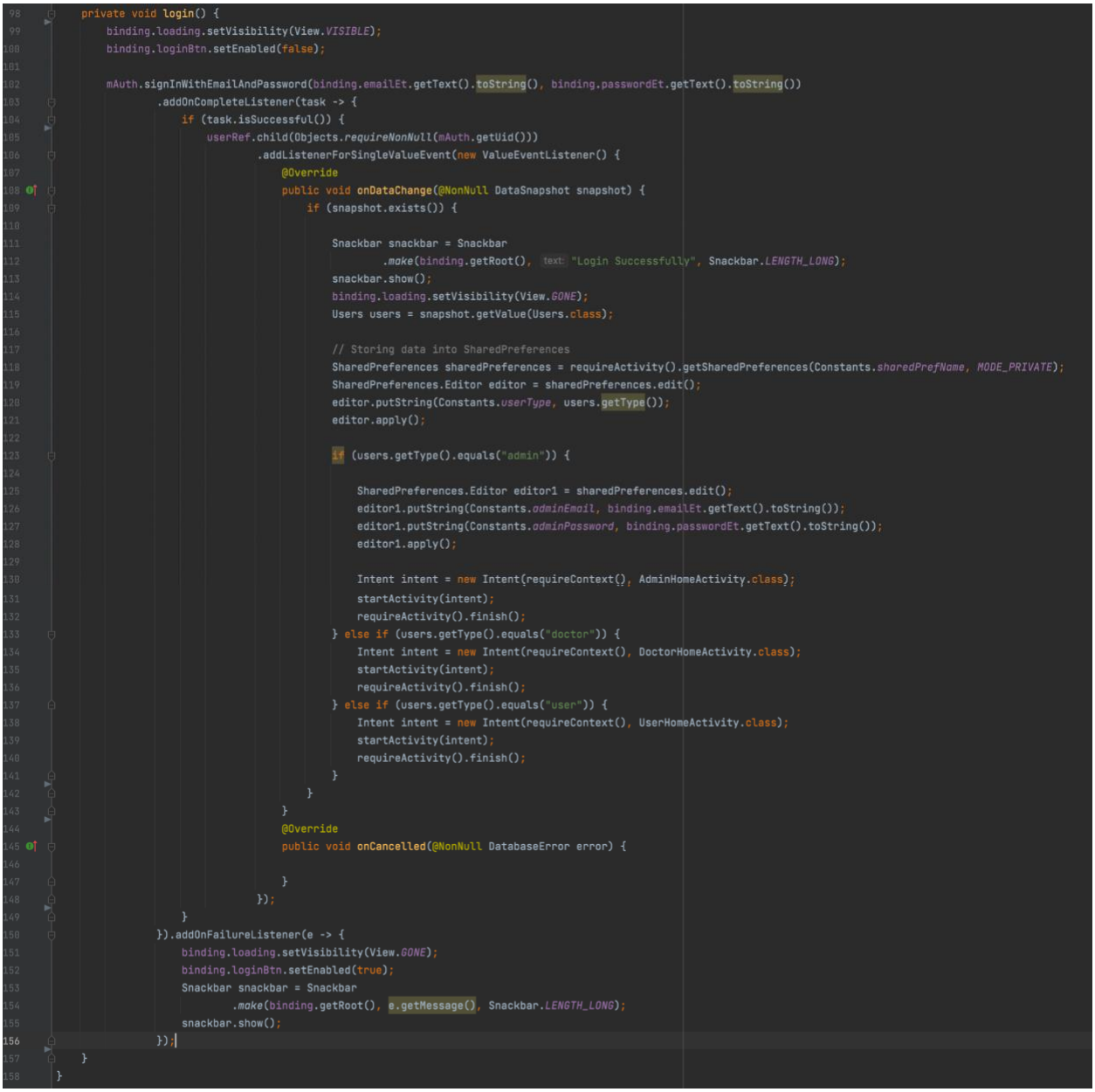
The figure above shows the codes for account registration using createUserWithEmailAndPassword method provided by Firebase Authentication. As stated in the previous chapter, View Binding is used for this project. For each XML layout file existing in the module, it creates a binding class. All views with an ID in the relevant layout are directly referenced in an instance of a binding class.For example, in line 40, call the static inflate() method included in the generated binding class. This creates an instance of the binding class for the activity to use instead of using findViewById. The edit text (Et) basically would accept email as the username and the respective password. When registration is successful, the snackbar will display the ‘Successful’ toast message.

4.2.2 Login

LoginFragment.java

```java
public class LoginFragment extends Fragment {
    private FragmentLoginBinding binding;
    private FirebaseAuth mAuth;
    private DatabaseReference userRef;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        binding = FragmentLoginBinding.inflate(inflater, container,  attachToParent: false);


        //initialize
        mAuth = FirebaseAuth.getInstance();

        userRef = FirebaseDatabase.getInstance().getReference( path: "users");


        //listeners
        binding.forgotPasswordTv.setOnClickListener(view -> Navigation.findNavController(view).navigate(R.id.action_loginFragment_to_forgotPasswordFragment));
        binding.registerTv.setOnClickListener(view -> Navigation.findNavController(view).navigate(R.id.action_loginFragment_to_registerFragment));


        //permissions
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            try {
                requestPermissions(new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE,
                        Manifest.permission.READ_EXTERNAL_STORAGE, Manifest.permission.CAMERA},  requestCode: 555);
            } catch (Exception e) {
                Log.e( tag: "userPer", e.toString());
            }
        }
```

```java
        binding.loginBtn.setOnClickListener(view -> {
            //checking validations
            if (Objects.requireNonNull(binding.emailEt.getText()).toString().isEmpty()) {
                binding.emailEt.setError("Please enter valid email");
                return;
            }

            if (Objects.requireNonNull(binding.passwordEt.getText()).toString().isEmpty()) {
                binding.passwordEt.setError("Please enter password");
                return;
            }

            //login
            login();
        });


        return binding.getRoot();
    }


    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
```



```
98      private void login() {
99          binding.loading.setVisibility(View.VISIBLE);
100         binding.loginBtn.setEnabled(false);
101
102         mAuth.signInWithEmailAndPassword(binding.emailEt.getText().toString(), binding.passwordEt.getText().toString())
103                 .addOnCompleteListener(task -> {
104                     if (task.isSuccessful()) {
105                         userRef.child(Objects.requireNonNull(mAuth.getUid()))
106                                 .addListenerForSingleValueEvent(new ValueEventListener() {
107                                     @Override
108                                     public void onDataChange(@NonNull DataSnapshot snapshot) {
109                                         if (snapshot.exists()) {
110
111                                             Snackbar snackbar = Snackbar
112                                                     .make(binding.getRoot(),  text: "Login Successfully", Snackbar.LENGTH_LONG);
113                                             snackbar.show();
114                                             binding.loading.setVisibility(View.GONE);
115                                             Users users = snapshot.getValue(Users.class);
116
117                                             // Storing data into SharedPreferences
118                                             SharedPreferences sharedPreferences = requireActivity().getSharedPreferences(Constants.sharedPrefName, MODE_PRIVATE);
119                                             SharedPreferences.Editor editor = sharedPreferences.edit();
120                                             editor.putString(Constants.userType, users.getType());
121                                             editor.apply();
122
123                                             if (users.getType().equals("admin")) {
124
125                                                 SharedPreferences.Editor editor1 = sharedPreferences.edit();
126                                                 editor1.putString(Constants.adminEmail, binding.emailEt.getText().toString());
127                                                 editor1.putString(Constants.adminPassword, binding.passwordEt.getText().toString());
128                                                 editor1.apply();
129
130                                                 Intent intent = new Intent(requireContext(), AdminHomeActivity.class);
131                                                 startActivity(intent);
132                                                 requireActivity().finish();
133                                             } else if (users.getType().equals("doctor")) {
134                                                 Intent intent = new Intent(requireContext(), DoctorHomeActivity.class);
135                                                 startActivity(intent);
136                                                 requireActivity().finish();
137                                             } else if (users.getType().equals("user")) {
138                                                 Intent intent = new Intent(requireContext(), UserHomeActivity.class);
139                                                 startActivity(intent);
140                                                 requireActivity().finish();
141                                             }
142                                         }
143                                     }
144                                     @Override
145                                     public void onCancelled(@NonNull DatabaseError error) {
146
147                                     }
148                                 });
149                     }
150                 }).addOnFailureListener(e -> {
151                     binding.loading.setVisibility(View.GONE);
152                     binding.loginBtn.setEnabled(true);
153                     Snackbar snackbar = Snackbar
154                             .make(binding.getRoot(), e.getMessage(), Snackbar.LENGTH_LONG);
155                     snackbar.show();
156                 });
157     }
158 }
```

Figure 6.2

The figure above shows the codes for Login Function. The username and password requires validation from the firebase in order to login users into their respective account. Hence, we implement mAuth.signInWithEmail linked to the firebase where if the validation is successful, the firebase would fetch users' accounts according to their username and password. Users will be directed to the home page after login. However, if the validation process fails, a toast message will pop up to inform the users that their username or password is invalid. From line 123, it will also check on the user's type before directing the respective users to their respective home page. Besides, addOnFailureListener() is to ensure users' meet the requirements of inputs as the validation would fail and an error message will be displayed if either textbox is left blank.

4.2.3 Home

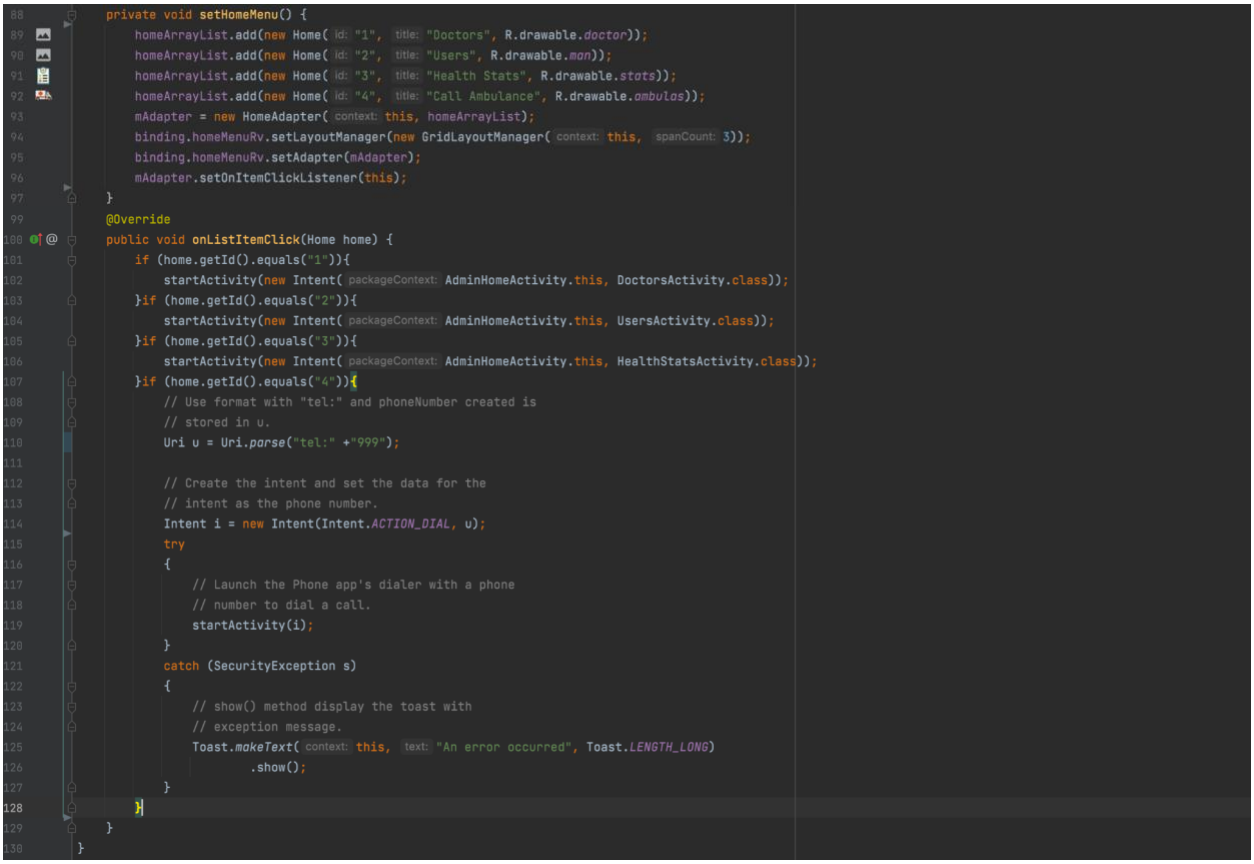
AdminHomeActivity.java

UserHomeActivity.java

DoctorHomeActivity.java

```java
public class AdminHomeActivity extends AppCompatActivity implements HomeAdapter.OnItemClickListener {
    private ActivityAdminHomeBinding binding;
    private HomeAdapter mAdapter;
    private ArrayList<Home> homeArrayList = new ArrayList<>();
    private FirebaseAuth mAuth;
    private DatabaseReference userRef;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityAdminHomeBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        //initialize
        mAuth = FirebaseAuth.getInstance();
        userRef = FirebaseDatabase.getInstance().getReference( path: "users");
        binding.userImage.setOnClickListener(v -> startActivity(new Intent( packageContext: AdminHomeActivity.this, ProfileActivity.class)));

        setHomeMenu();

    }
    @Override
    protected void onStart() {
        super.onStart();
        getUserData();

    }
```

```java
    private void getUserData() {
        userRef.child(Objects.requireNonNull(mAuth.getUid()))
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot snapshot) {
                        if (snapshot.exists()) {


                            Users users = snapshot.getValue(Users.class);

                            assert users != null;
                            if (!users.getImage().equals("")) {
                                Glide.with( activity: AdminHomeActivity.this)
                                        .load(users.getImage())
                                        .into(binding.userImage);
                            }
                        }
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError error) {
                        Snackbar snackbar = Snackbar
                                .make(binding.getRoot(), error.getMessage(), Snackbar.LENGTH_LONG);
                        snackbar.show();
                    }
                });

    }
```

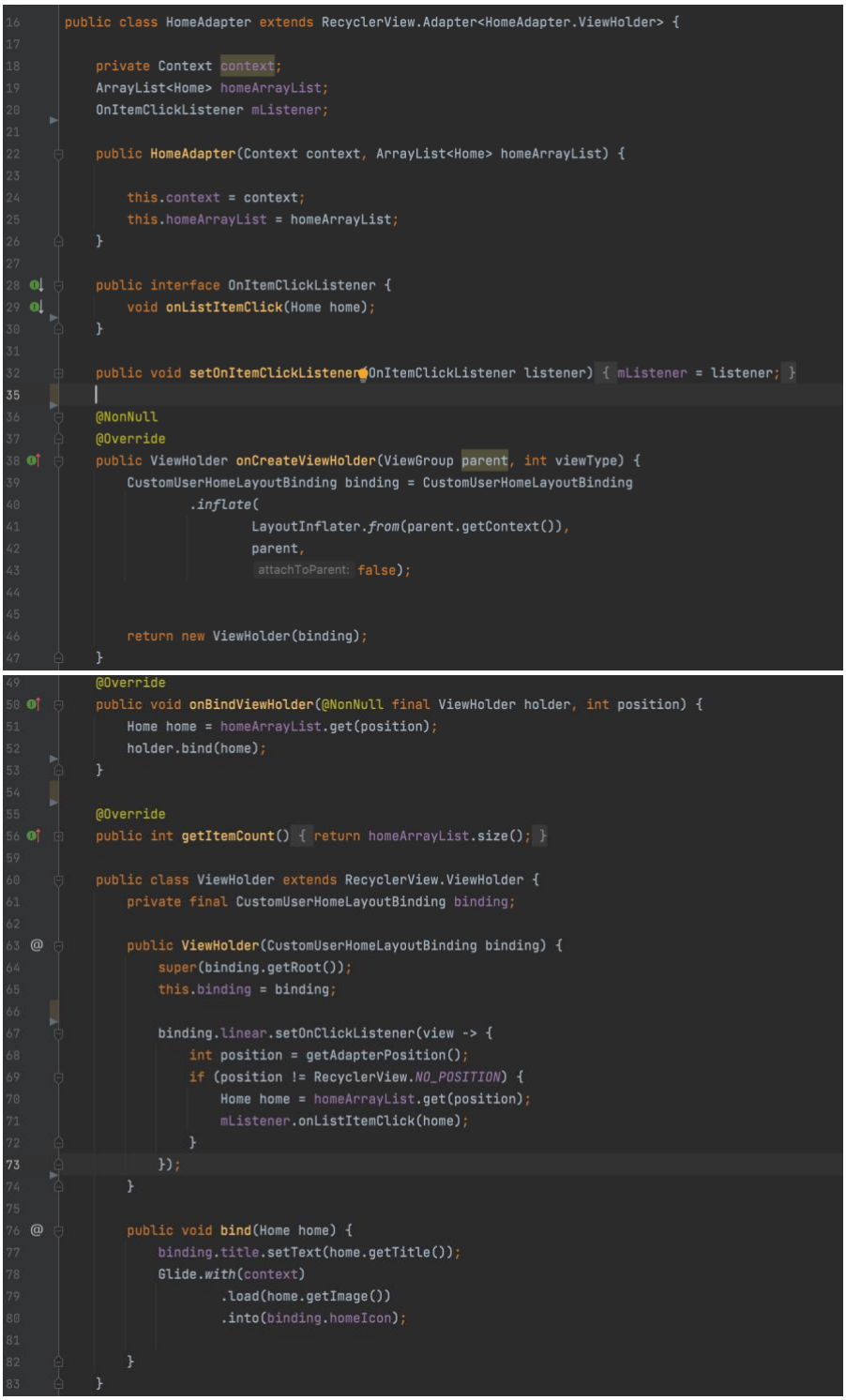


```
    private void setHomeMenu() {
        homeArrayList.add(new Home( id: "1",  title: "Doctors", R.drawable.doctor));
        homeArrayList.add(new Home( id: "2",  title: "Users", R.drawable.man));
        homeArrayList.add(new Home( id: "3",  title: "Health Stats", R.drawable.stats));
        homeArrayList.add(new Home( id: "4",  title: "Call Ambulance", R.drawable.ambulos));
        mAdapter = new HomeAdapter( context: this, homeArrayList);
        binding.homeMenuRv.setLayoutManager(new GridLayoutManager( context: this,  spanCount: 3));
        binding.homeMenuRv.setAdapter(mAdapter);
        mAdapter.setOnItemClickListener(this);
    }
    @Override
    public void onListItemClick(Home home) {
        if (home.getId().equals("1")){
            startActivity(new Intent( packageContext: AdminHomeActivity.this, DoctorsActivity.class));
        }if (home.getId().equals("2")){
            startActivity(new Intent( packageContext: AdminHomeActivity.this, UsersActivity.class));
        }if (home.getId().equals("3")){
            startActivity(new Intent( packageContext: AdminHomeActivity.this, HealthStatsActivity.class));
        }if (home.getId().equals("4")){
            // Use format with "tel:" and phoneNumber created is
            // stored in u.
            Uri u = Uri.parse("tel:" +"999");

            // Create the intent and set the data for the
            // intent as the phone number.
            Intent i = new Intent(Intent.ACTION_DIAL, u);
            try
            {
                // Launch the Phone app's dialer with a phone
                // number to dial a call.
                startActivity(i);
            }
            catch (SecurityException s)
            {
                // show() method display the toast with
                // exception message.
                Toast.makeText( context: this,  text: "An error occurred", Toast.LENGTH_LONG)
                        .show();
            }
        }
    }
}
```

Figure 6.3

The figure above shows the codes for the admin home page but for users and doctors home page, the codes will be similar to the admin. In the home page, there will be menus for users to navigate to another page. These menu items are stored into homeArrayList and set into respective ids. The menu items are then added to the onListItemClick and assign intents to their respective layout.

HomeAdapter.java



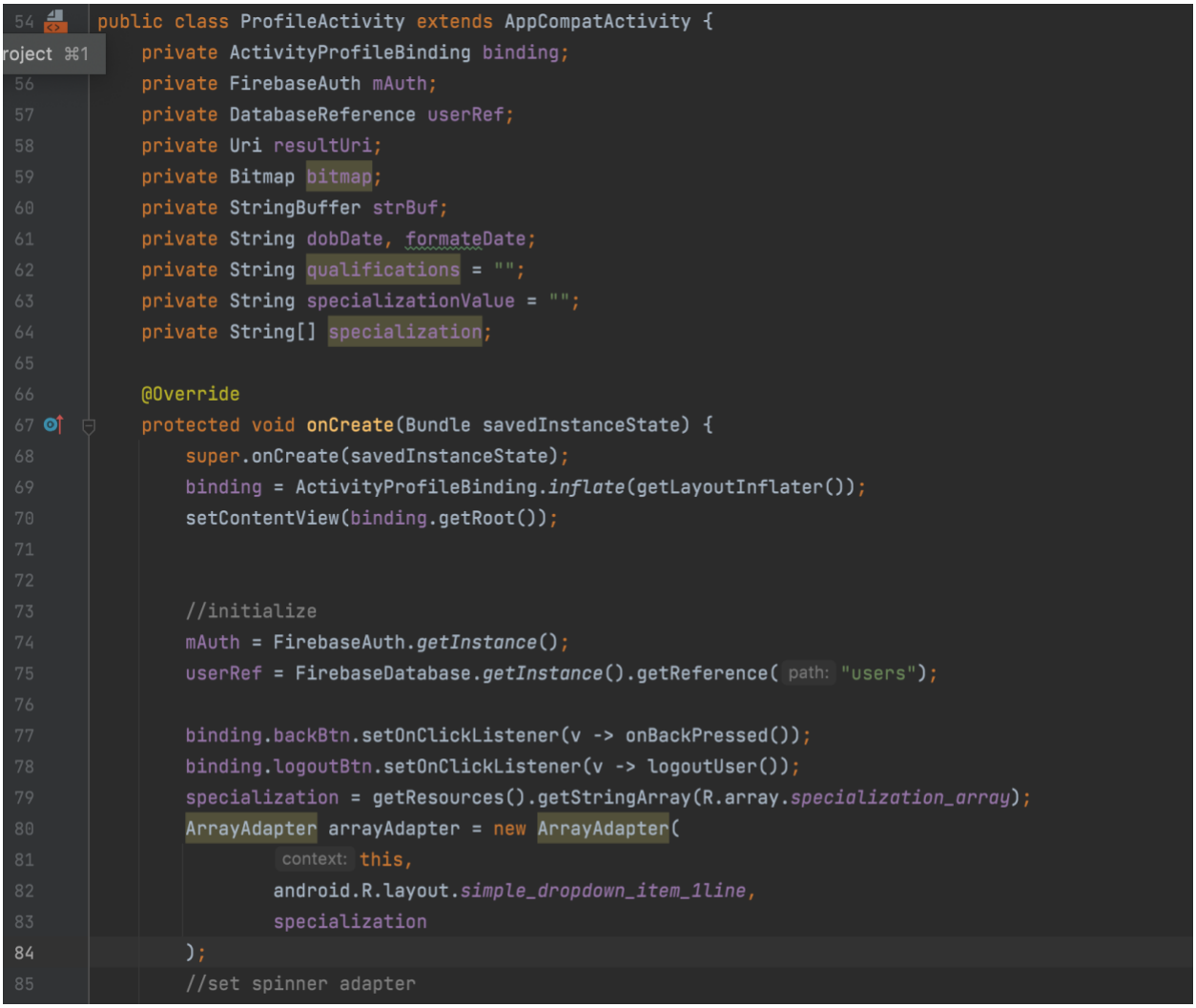
```
public class HomeAdapter extends RecyclerView.Adapter<HomeAdapter.ViewHolder> {

    private Context context;
    ArrayList<Home> homeArrayList;
    OnItemClickListener mListener;

    public HomeAdapter(Context context, ArrayList<Home> homeArrayList) {

        this.context = context;
        this.homeArrayList = homeArrayList;
    }

    public interface OnItemClickListener {
        void onListItemClick(Home home);
    }

    public void setOnItemClickListener(OnItemClickListener listener) { mListener = listener; }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        CustomUserHomeLayoutBinding binding = CustomUserHomeLayoutBinding
                .inflate(
                        LayoutInflater.from(parent.getContext()),
                        parent,
                        attachToParent: false);


        return new ViewHolder(binding);
    }
    @Override
    public void onBindViewHolder(@NonNull final ViewHolder holder, int position) {
        Home home = homeArrayList.get(position);
        holder.bind(home);
    }

    @Override
    public int getItemCount() { return homeArrayList.size(); }

    public class ViewHolder extends RecyclerView.ViewHolder {
        private final CustomUserHomeLayoutBinding binding;

        public ViewHolder(CustomUserHomeLayoutBinding binding) {
            super(binding.getRoot());
            this.binding = binding;

            binding.linear.setOnClickListener(view -> {
                int position = getAdapterPosition();
                if (position != RecyclerView.NO_POSITION) {
                    Home home = homeArrayList.get(position);
                    mListener.onListItemClick(home);
                }
            });
        }

        public void bind(Home home) {
            binding.title.setText(home.getTitle());
            Glide.with(context)
                    .load(home.getImage())
                    .into(binding.homeIcon);

        }
    }
```

Figure 6.4

Figure above shows the codes for Home Adapter. It is to display the menu items into the recycler view using ViewHolder.

4.2.3 Profile & Logout

ProfileActivity.java



```
public class ProfileActivity extends AppCompatActivity {
    private ActivityProfileBinding binding;
    private FirebaseAuth mAuth;
    private DatabaseReference userRef;
    private Uri resultUri;
    private Bitmap bitmap;
    private StringBuffer strBuf;
    private String dobDate, formateDate;
    private String qualifications = "";
    private String specializationValue = "";
    private String[] specialization;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityProfileBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());


        //initialize
        mAuth = FirebaseAuth.getInstance();
        userRef = FirebaseDatabase.getInstance().getReference( path: "users");

        binding.backBtn.setOnClickListener(v -> onBackPressed());
        binding.logoutBtn.setOnClickListener(v -> logoutUser());
        specialization = getResources().getStringArray(R.array.specialization_array);
        ArrayAdapter arrayAdapter = new ArrayAdapter(
                context: this,
                android.R.layout.simple_dropdown_item_1line,
                specialization
        );
        //set spinner adapter
```

```
        //set spinner adapter
        binding.specializationSpinner.setAdapter(arrayAdapter);
        //getting profile type
        binding.specializationSpinner.setOnItemSelectedListener(new MaterialSpinner.OnItemSelectedListener() {
            @Override
            public void onItemSelected(@NonNull MaterialSpinner materialSpinner, @Nullable View view, int i, long l) {
                specializationValue = Objects.requireNonNull(materialSpinner.getSelectedItem()).toString();
            }

            @Override
            public void onNothingSelected(@NonNull MaterialSpinner materialSpinner) {

            }
        });

        binding.editBtn.setOnClickListener(v -> {
            if (binding.editBtn.getText().toString().equals("Edit")) {

                binding.name.setEnabled(true);
                binding.dobEt.setEnabled(true);
                binding.phoneNoEt.setEnabled(true);
                binding.userImage.setEnabled(true);
                binding.specializationSpinner.setEnabled(true);
                binding.qualificationEt.setEnabled(true);
                binding.editBtn.setText("Update");
            } else {

                //check validations
                if (Objects.requireNonNull(binding.name.getText()).toString().isEmpty()) {
                    binding.name.setError("Please enter user name");
                    return;
                }
                if (Objects.requireNonNull(binding.phoneNoEt.getText()).toString().isEmpty()) {
                    binding.phoneNoEt.setError("Please enter phone number");
                    return;
                }

                if (Objects.requireNonNull(binding.dobEt.getText()).toString().isEmpty()) {
                    binding.dobEt.setError("Please select date of birth");
                    return;
                }
                updateProfile();
            }
        });

        binding.userImage.setOnClickListener(v -> checkAndroidVersion());
        binding.dobEt.setOnClickListener(v -> showDatePickerDialog());
        getUserData();

    }
```

```java
private void logoutUser() {
    new AlertDialog.Builder( context: this)
            .setTitle("Logout")
            .setMessage("Do you really want to logout?")
            .setIcon(android.R.drawable.ic_dialog_alert)
            .setPositiveButton(android.R.string.yes, (dialog, whichButton) -> {
                SharedPreferences settings = getSharedPreferences(Constants.sharedPrefName, Context.MODE_PRIVATE);
                settings.edit().clear().apply();
                mAuth.signOut();
                Intent intent = new Intent( packageContext: ProfileActivity.this, AuthActivity.class);
                startActivity(intent);
                finishAffinity();
            })
            .setNegativeButton(android.R.string.no,  listener: null).show();
}
```

```java
private void showDatePickerDialog() {
    // Create a new OnDateSetListener instance. This listener will be invoked when user click ok button in DatePickerDialog.
    DatePickerDialog.OnDateSetListener onDateSetListener = (datePicker, year, month, dayOfMonth) -> {
        strBuf = new StringBuffer();
        // strBuf.append("You select date is ");
        if ((month + 1) < 10) {
            strBuf.append("0" + (month + 1));
        } else {
            strBuf.append(month + 1);
        }
        strBuf.append("-");

        if (dayOfMonth < 10) {
            strBuf.append("0" + dayOfMonth);
        } else {
            strBuf.append(dayOfMonth);
        }
        strBuf.append("-");

        strBuf.append(year);

        String ages = getAge(year,  month: month + 1, dayOfMonth);

        dobDate = strBuf.toString();

        try {
            DateFormat inputFormat = new SimpleDateFormat( pattern: "MM-dd-yyyy");
            DateFormat outputFormat = new SimpleDateFormat( pattern: "MMM-dd-yyyy");
            String inputDateStr = strBuf.toString();
            Date date = inputFormat.parse(inputDateStr);
            formateDate = outputFormat.format(date);
        } catch (ParseException e) {
```

```
                e.printStackTrace();
            }
            binding.dobEt.setText(ages + " " + "Years" + " / " + formateDate);
        };
        // Get current year, month and day.
        Calendar now = Calendar.getInstance();
        int year = now.get(Calendar.YEAR);
        int month = now.get(Calendar.MONTH);
        int day = now.get(Calendar.DAY_OF_MONTH);
        DatePickerDialog datePickerDialog = new DatePickerDialog( context: ProfileActivity.this, onDateSetListener, year, month, day);
        now.add(Calendar.YEAR,  amount: -18);

        datePickerDialog.getDatePicker().setMaxDate(now.getTimeInMillis());
        datePickerDialog.show();
    }

    private String getAge(int year, int month, int day) {

        Calendar dob = Calendar.getInstance();
        Calendar today = Calendar.getInstance();
        int todaymonth = today.get(Calendar.MONTH);
        int todayyear = today.get(Calendar.YEAR);
        int today_date = today.get(Calendar.DAY_OF_MONTH);
        today.set(todayyear,  month: todaymonth + 1,  date: today_date + 1);
        dob.set(year, month, day);
        int age = today.get(Calendar.YEAR) - dob.get(Calendar.YEAR);

        System.out.println("DateFormat1 " + today.get(Calendar.YEAR) + "---" + dob.get(Calendar.YEAR));
        System.out.println("DateFormat2 " + today.get(Calendar.DAY_OF_YEAR) + "---" + dob.get(Calendar.DAY_OF_YEAR));

        if (today.get(Calendar.DAY_OF_YEAR) < dob.get(Calendar.DAY_OF_YEAR)) {
            age--;
        }
```

```
        Integer ageInt = new Integer(age);
        String ageS = ageInt.toString();
        return ageS;
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {

        //RESULT FROM SELECTED IMAGE
        super.onActivityResult(requestCode, resultCode, data);


        if (resultCode == Activity.RESULT_OK && requestCode == ImagePicker.REQUEST_CODE && data != null) {
            //Image Uri will not be null for RESULT_OK
            Uri uri = data.getData();

            resultUri = uri;

            try {
                bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), uri);
                binding.userImage.setImageBitmap(bitmap);
            } catch (IOException e) {
                e.printStackTrace();
            }
            // Use Uri object instead of File to avoid storage permissions

        }
    }
```

```java
    private void getUserData() {
        binding.loading.setVisibility(View.VISIBLE);
        userRef.child(Objects.requireNonNull(mAuth.getUid()))
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot snapshot) {
                        if (snapshot.exists()) {
                            binding.loading.setVisibility(View.GONE);
                            binding.emailEt.setEnabled(false);
                            binding.name.setEnabled(false);
                            binding.dobEt.setEnabled(false);
                            binding.phoneNoEt.setEnabled(false);
                            binding.userImage.setEnabled(false);
                            binding.qualificationEt.setEnabled(false);
                            binding.specializationSpinner.setEnabled(false);

                            Users users = snapshot.getValue(Users.class);

                            assert users != null;
                            if (users.getType().equals("doctor")) {
                                binding.qualificationLayout.setVisibility(View.VISIBLE);
                                binding.qualificationEt.setText(users.getQualification());
                                binding.specializationSpinner.setVisibility(View.VISIBLE);
                                specializationValue = users.getSpecialization();
                            }

                            binding.emailEt.setText(users.getEmail());
                            binding.name.setText(users.getName());
                            binding.dobEt.setText(users.getDob());
                            binding.phoneNoEt.setText(users.getPhone());
                            dobDate = users.getDob();
```

```java
                            if (!users.getImage().equals("")) {
                                Glide.with( activity: ProfileActivity.this)
                                        .load(users.getImage())
                                        .into(binding.userImage);
                            }

                        }
                    }
                    @Override
                    public void onCancelled(@NonNull DatabaseError error) {
                        Snackbar snackbar = Snackbar
                                .make(binding.getRoot(), error.getMessage(), Snackbar.LENGTH_LONG);
                        snackbar.show();
                        binding.loading.setVisibility(View.GONE);
                    }
                });
    }

    public void checkAndroidVersion() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            try {
                requestPermissions(new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE, Manifest.permission.READ_EXTERNAL_STORAGE, Manifest.permission.CAMERA}, requestCode: 555);
            } catch (Exception e) {

            }
        } else {
            ImagePicker.with( activity: ProfileActivity.this)
                    .crop()                    //Crop image(Optional), Check Customization for more option
                    .compress(1024)            //Final image size will be less than 1 MB(Optional)
                    .maxResultSize( width: 1080,  height: 1080)    //Final image resolution will be less than 1080 x 1080(Optional)
                    .start();
        }
```

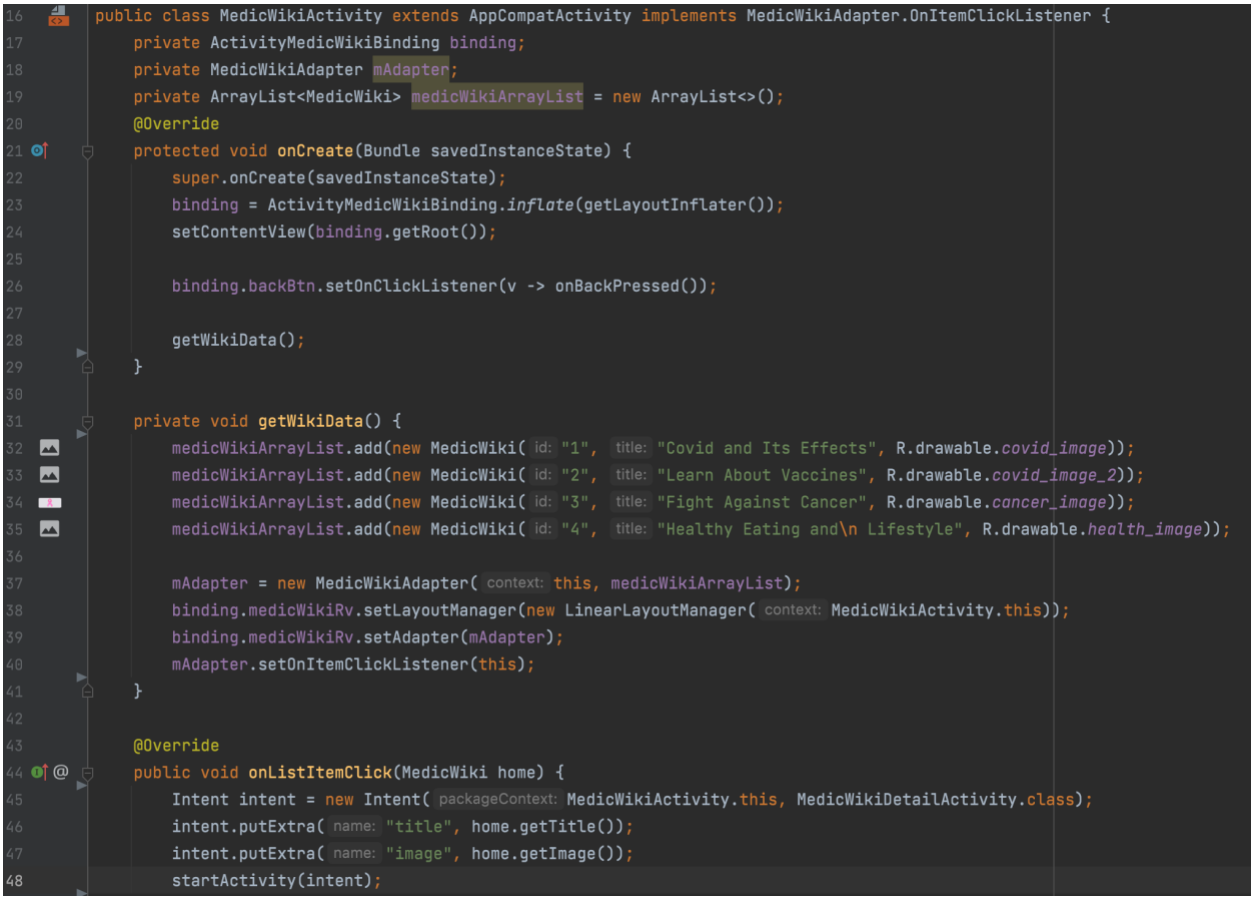
```java
private void updateProfile() {
    if (resultUri != null) {
        binding.loading.setVisibility(View.VISIBLE);

        final StorageReference filePath = FirebaseStorage.getInstance().getReference().child("Users").child(resultUri.getLastPathSegment());
        Bitmap bitmap = null;
        try {
            bitmap = MediaStore.Images.Media.getBitmap(this.getContentResolver(), resultUri);
        } catch (IOException e) {
            e.printStackTrace();
        }

        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        bitmap.compress(Bitmap.CompressFormat.JPEG, quality: 20, baos);
        byte[] data = baos.toByteArray();
        UploadTask uploadTask = filePath.putBytes(data);

        uploadTask.addOnFailureListener(e -> {
            binding.loading.setVisibility(View.GONE);
            Toast.makeText( context: ProfileActivity.this, text: "" + e.getMessage(), Toast.LENGTH_SHORT).show();
            return;
        });
        uploadTask.addOnSuccessListener(taskSnapshot -> filePath.getDownloadUrl().addOnSuccessListener(uri -> {


            Map<String, Object> updateValue = new HashMap<>();
            updateValue.put("name", binding.name.getText().toString());
            updateValue.put("phone", binding.phoneNoEt.getText().toString());
            updateValue.put("dob", dobDate);
            updateValue.put("image", uri.toString());
            updateValue.put("specialization", specializationValue);
            if (!binding.qualificationEt.getText().toString().isEmpty()) {
                updateValue.put("qualification", binding.qualificationEt.getText().toString());
            } else {
                updateValue.put("qualification", qualifications);
            }

            userRef.child(mAuth.getUid()).updateChildren(updateValue).addOnCompleteListener(task -> {
                binding.loading.setVisibility(View.GONE);
                Toast.makeText( context: ProfileActivity.this, text: "Profile Updated Successfully", Toast.LENGTH_SHORT).show();
                onBackPressed();

            }).addOnFailureListener(e -> {
                binding.loading.setVisibility(View.GONE);
                Toast.makeText( context: ProfileActivity.this, text: "" + e.getMessage(), Toast.LENGTH_SHORT).show();
            });
            return;

        }).addOnFailureListener(exception -> {
            binding.loading.setVisibility(View.GONE);
            Toast.makeText( context: ProfileActivity.this, text: "" + exception.getMessage(), Toast.LENGTH_SHORT).show();

        }));
    } else {
        binding.loading.setVisibility(View.VISIBLE);

        Map<String, Object> updateValue = new HashMap<>();
        updateValue.put("name", binding.name.getText().toString());
        updateValue.put("phone", binding.phoneNoEt.getText().toString());
        updateValue.put("dob", dobDate);
        updateValue.put("specialization", specializationValue);
        if (!binding.qualificationEt.getText().toString().isEmpty()) {
            updateValue.put("qualification", binding.qualificationEt.getText().toString());
        } else {
```

Figure 6.5

Figure above shows the code for the profile page of admin, doctors, and users. ProfileActivity.java is implemented to show and update the information of admin, doctors, and users. logoutUser() function is used to sign users out when the button is clicked.

4.2.4 Medic Wiki

MedicWikiActivity.java



```
public class MedicWikiActivity extends AppCompatActivity implements MedicWikiAdapter.OnItemClickListener {
    private ActivityMedicWikiBinding binding;
    private MedicWikiAdapter mAdapter;
    private ArrayList<MedicWiki> medicWikiArrayList = new ArrayList<>();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMedicWikiBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        binding.backBtn.setOnClickListener(v -> onBackPressed());

        getWikiData();
    }

    private void getWikiData() {
        medicWikiArrayList.add(new MedicWiki( id: "1",  title: "Covid and Its Effects", R.drawable.covid_image));
        medicWikiArrayList.add(new MedicWiki( id: "2",  title: "Learn About Vaccines", R.drawable.covid_image_2));
        medicWikiArrayList.add(new MedicWiki( id: "3",  title: "Fight Against Cancer", R.drawable.cancer_image));
        medicWikiArrayList.add(new MedicWiki( id: "4",  title: "Healthy Eating and\n Lifestyle", R.drawable.health_image));

        mAdapter = new MedicWikiAdapter( context: this, medicWikiArrayList);
        binding.medicWikiRv.setLayoutManager(new LinearLayoutManager( context: MedicWikiActivity.this));
        binding.medicWikiRv.setAdapter(mAdapter);
        mAdapter.setOnItemClickListener(this);
    }

    @Override
    public void onListItemClick(MedicWiki home) {
        Intent intent = new Intent( packageContext: MedicWikiActivity.this, MedicWikiDetailActivity.class);
        intent.putExtra( name: "title", home.getTitle());
        intent.putExtra( name: "image", home.getImage());
        startActivity(intent);
```

Figure 6.6

The figure above shows the codes for the Medic Wiki feature. getWikiData() is implemented to get information to store into medicWikiArrayList.

MedicWikiAdapter.java

```
public class MedicWikiAdapter extends RecyclerView.Adapter<MedicWikiAdapter.ViewHolder> {
    private Context context;
    ArrayList<MedicWiki> homeArrayList;
    OnItemClickListener mListener;

    public MedicWikiAdapter(Context context, ArrayList<MedicWiki> homeArrayList) {
        this.context = context;
        this.homeArrayList = homeArrayList;
    }

    public interface OnItemClickListener {
        void onListItemClick(MedicWiki home);
    }

    public void setOnItemClickListener(OnItemClickListener listener) { mListener = listener; }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        CustomMedicWikiLayoutBinding binding = CustomMedicWikiLayoutBinding
                .inflate(
                        LayoutInflater.from(parent.getContext()),
                        parent,
                        attachToParent: false);

        return new ViewHolder(binding);
    }

    @Override
    public void onBindViewHolder(@NonNull final ViewHolder holder, int position) {
        MedicWiki home = homeArrayList.get(position);
        holder.bind(home);
    }

    @Override
    public int getItemCount() { return homeArrayList.size(); }

    public class ViewHolder extends RecyclerView.ViewHolder {
        private final CustomMedicWikiLayoutBinding binding;

        public ViewHolder(CustomMedicWikiLayoutBinding binding) {
            super(binding.getRoot());
            this.binding = binding;

            binding.cardView.setOnClickListener(view -> {
                int position = getAdapterPosition();
                if (position != RecyclerView.NO_POSITION) {
                    MedicWiki home = homeArrayList.get(position);
                    mListener.onListItemClick(home);
                }
            });
        }

        public void bind(MedicWiki home) {
            binding.title.setText(home.getTitle());
            Glide.with(context)
                    .load(home.getImage())
                    .into(binding.image);
        }
    }
```

Figure 6.7

The figure above shows the codes for the MedicWikiAdapter. The adapter is used to display the information into a cardview using a viewholder.

MedicWikiDetailActivity.java

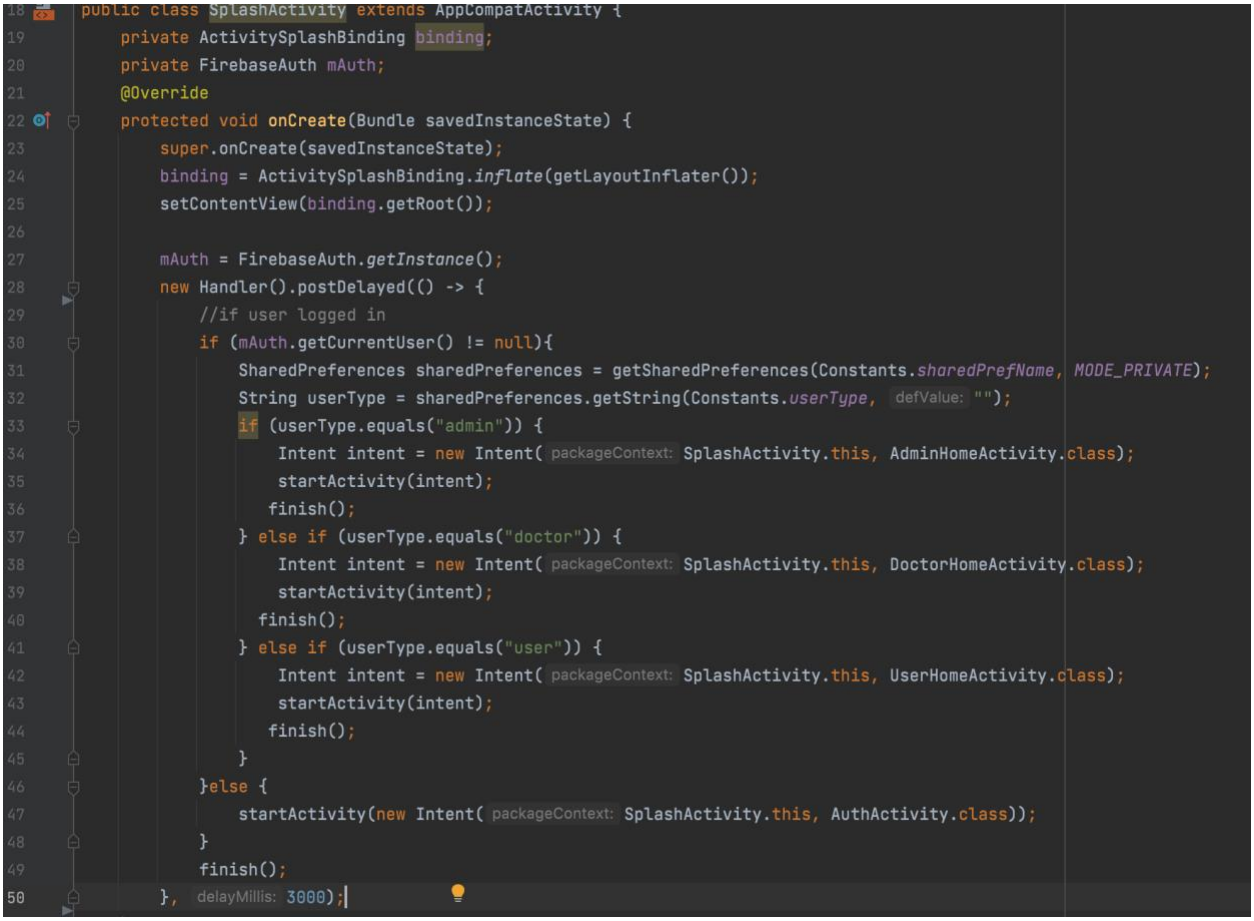
```
public class MedicWikiDetailActivity extends AppCompatActivity {
    private ActivityMedicWikiDetailBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMedicWikiDetailBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        binding.backBtn.setOnClickListener(v -> onBackPressed());

        Bundle bundle = getIntent().getExtras();
        if (bundle != null) {
            Glide.with( activity: this)
                    .load(bundle.getInt( key: "image"))
                    .into(binding.wikiImage);

            binding.title.setText(bundle.getString( key: "title"));
        }

    }
}
```

Figure 6.8

The figure above shows the codes for the MedicWikiDetailActivity. It is the main activity where the information and articles are published.

4.2.5 Loading Screen

SplashActivity.java



```
public class SplashActivity extends AppCompatActivity {
    private ActivitySplashBinding binding;
    private FirebaseAuth mAuth;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivitySplashBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        mAuth = FirebaseAuth.getInstance();
        new Handler().postDelayed(() -> {
            //if user logged in
            if (mAuth.getCurrentUser() != null){
                SharedPreferences sharedPreferences = getSharedPreferences(Constants.sharedPrefName, MODE_PRIVATE);
                String userType = sharedPreferences.getString(Constants.userType, defValue: "");
                if (userType.equals("admin")) {
                    Intent intent = new Intent( packageContext: SplashActivity.this, AdminHomeActivity.class);
                    startActivity(intent);
                   finish();
                } else if (userType.equals("doctor")) {
                    Intent intent = new Intent( packageContext: SplashActivity.this, DoctorHomeActivity.class);
                    startActivity(intent);
                  finish();
                } else if (userType.equals("user")) {
                    Intent intent = new Intent( packageContext: SplashActivity.this, UserHomeActivity.class);
                    startActivity(intent);
                   finish();
                }
            }else {
                startActivity(new Intent( packageContext: SplashActivity.this, AuthActivity.class));
            }
            finish();
        }, delayMillis: 3000);
```

Figure 6.9

The figure above shows the codes for the SplashActivity. It is the loading screen when the application is launched, and the duration is set to 3 seconds.

4.2.6 Add Doctors

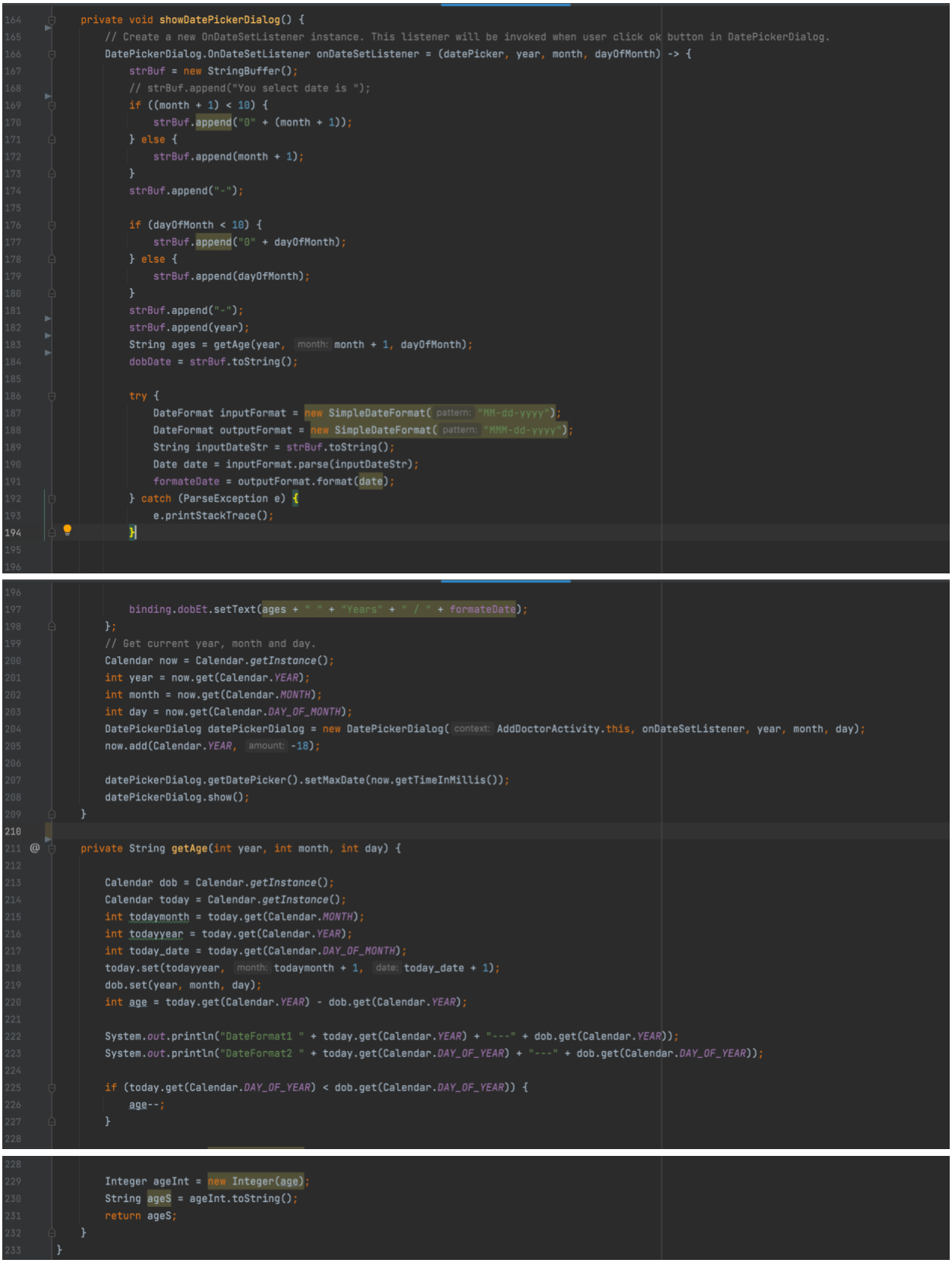
AddDoctorActivity.java

```java
public class AddDoctorActivity extends AppCompatActivity {
    private ActivityAddDoctorBinding binding;
    private String[] specialization;
    private String specializationValue = "";
    private StringBuffer strBuf;
    private String dobDate, formateDate;
    private DatabaseReference userRef;
    private FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityAddDoctorBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        //initialize
        userRef = FirebaseDatabase.getInstance().getReference( path: "users");
        mAuth = FirebaseAuth.getInstance();

        specialization = getResources().getStringArray(R.array.specialization_array);
        //click listeners
        binding.backArrow.setOnClickListener(view -> onBackPressed());

        binding.dobEt.setOnClickListener(v -> showDatePickerDialog());
        ArrayAdapter arrayAdapter = new ArrayAdapter(
                context: this,
                android.R.layout.simple_dropdown_item_1line,
                specialization
        );
        //set spinner adapter
        binding.specializationSpinner.setAdapter(arrayAdapter);


```

```
        //getting profile type
        binding.specializationSpinner.setOnItemSelectedListener(new MaterialSpinner.OnItemSelectedListener() {
            @Override
            public void onItemSelected(@NonNull MaterialSpinner materialSpinner, @Nullable View view, int i, long l) {
                specializationValue = Objects.requireNonNull(materialSpinner.getSelectedItem()).toString();
            }

            @Override
            public void onNothingSelected(@NonNull MaterialSpinner materialSpinner) {

            }
        });

        binding.registerDoctorBtn.setOnClickListener(view -> {

            //check validations
            if (Objects.requireNonNull(binding.name.getText()).toString().isEmpty()) {
                binding.name.setError("Please enter user name");
                return;
            }
            if (Objects.requireNonNull(binding.phoneNoEt.getText()).toString().isEmpty()) {
                binding.phoneNoEt.setError("Please enter phone number");
                return;
            }
            if (Objects.requireNonNull(binding.emailEt.getText()).toString().isEmpty()) {
                binding.emailEt.setError("Please enter valid email");
                return;
            }
            if (Objects.requireNonNull(binding.passwordEt.getText()).toString().isEmpty()) {
                binding.passwordEt.setError("Please enter password");
                return;
            }

```

```
        if (Objects.requireNonNull(binding.dobEt.getText()).toString().isEmpty()) {
            binding.dobEt.setError("Please select date of birth");
            return;
        }
        if (Objects.requireNonNull(binding.qualificationEt.getText()).toString().isEmpty()) {
            binding.dobEt.setError("Please select qualification");
            return;
        }
        if (specializationValue.equals("")) {
            Toast.makeText( context: this,  text: "Please select specialization", Toast.LENGTH_SHORT).show();
            return;
        }
        //register doctor
        registerDoctor();
    });

}

private void registerDoctor() {
    binding.loading.setVisibility(View.VISIBLE);
    binding.registerDoctorBtn.setEnabled(false);
    mAuth.createUserWithEmailAndPassword(Objects.requireNonNull(binding.emailEt.getText()).toString(),
                    Objects.requireNonNull(binding.passwordEt.getText()).toString())
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {

                    //store user detail in firebase
                    Users users;
                    users = new Users(
                            mAuth.getUid(),
                            Objects.requireNonNull(binding.name.getText()).toString(),
                            Objects.requireNonNull(binding.phoneNoEt.getText()).toString(),
                            dobDate,
```

```
                            type: "doctor",
                            binding.emailEt.getText().toString(),
                            image: "",
                            specializationValue,
                            binding.qualificationEt.getText().toString()
                    );

                    userRef.child(Objects.requireNonNull(mAuth.getUid())).setValue(users).addOnCompleteListener(task1 -> {
                        if (task1.isSuccessful()) {
                            binding.loading.setVisibility(View.GONE);

                            mAuth.signOut();

                            SharedPreferences sharedPreferences = getSharedPreferences(Constants.sharedPrefName, MODE_PRIVATE);

                            mAuth.signInWithEmailAndPassword(sharedPreferences.getString(Constants.adminEmail,  defValue: ""),

                                    sharedPreferences.getString(Constants.adminPassword,  defValue: "")).addOnCompleteListener(task2 -> {
                                        Snackbar snackbar = Snackbar
                                                .make(binding.getRoot(),  text: "Doctor Added Successfully", Snackbar.LENGTH_LONG);
                                        snackbar.show();
                                        onBackPressed();
                                    });
                        }
                    });

                }
            }).addOnFailureListener(e -> {
                binding.loading.setVisibility(View.GONE);
                binding.registerDoctorBtn.setEnabled(true);
                Snackbar snackbar = Snackbar
                        .make(binding.getRoot(), Objects.requireNonNull(e.getMessage()), Snackbar.LENGTH_LONG);
                snackbar.show();
```



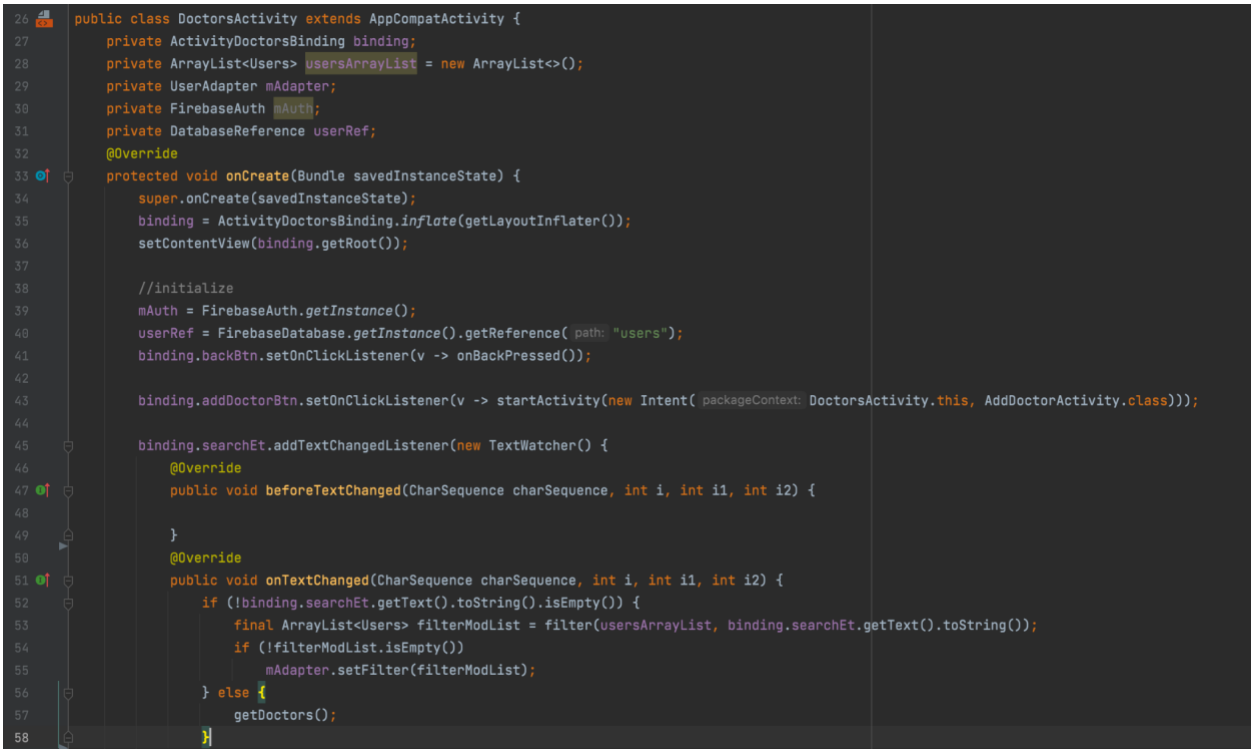
```
    private void showDatePickerDialog() {
        // Create a new OnDateSetListener instance. This listener will be invoked when user click ok button in DatePickerDialog.
        DatePickerDialog.OnDateSetListener onDateSetListener = (datePicker, year, month, dayOfMonth) -> {
            strBuf = new StringBuffer();
            // strBuf.append("You select date is ");
            if ((month + 1) < 10) {
                strBuf.append("0" + (month + 1));
            } else {
                strBuf.append(month + 1);
            }
            strBuf.append("-");

            if (dayOfMonth < 10) {
                strBuf.append("0" + dayOfMonth);
            } else {
                strBuf.append(dayOfMonth);
            }
            strBuf.append("-");
            strBuf.append(year);
            String ages = getAge(year,  month: month + 1, dayOfMonth);
            dobDate = strBuf.toString();

            try {
                DateFormat inputFormat = new SimpleDateFormat( pattern: "MM-dd-yyyy");
                DateFormat outputFormat = new SimpleDateFormat( pattern: "MMM-dd-yyyy");
                String inputDateStr = strBuf.toString();
                Date date = inputFormat.parse(inputDateStr);
                formateDate = outputFormat.format(date);
            } catch (ParseException e) {
                e.printStackTrace();
            }


            binding.dobEt.setText(ages + " " + "Years" + " / " + formateDate);
        };
        // Get current year, month and day.
        Calendar now = Calendar.getInstance();
        int year = now.get(Calendar.YEAR);
        int month = now.get(Calendar.MONTH);
        int day = now.get(Calendar.DAY_OF_MONTH);
        DatePickerDialog datePickerDialog = new DatePickerDialog( context: AddDoctorActivity.this, onDateSetListener, year, month, day);
        now.add(Calendar.YEAR,  amount: -18);

        datePickerDialog.getDatePicker().setMaxDate(now.getTimeInMillis());
        datePickerDialog.show();
    }

    private String getAge(int year, int month, int day) {

        Calendar dob = Calendar.getInstance();
        Calendar today = Calendar.getInstance();
        int todaymonth = today.get(Calendar.MONTH);
        int todayyear = today.get(Calendar.YEAR);
        int today_date = today.get(Calendar.DAY_OF_MONTH);
        today.set(todayyear,  month: todaymonth + 1,  date: today_date + 1);
        dob.set(year, month, day);
        int age = today.get(Calendar.YEAR) - dob.get(Calendar.YEAR);

        System.out.println("DateFormat1 " + today.get(Calendar.YEAR) + "---" + dob.get(Calendar.YEAR));
        System.out.println("DateFormat2 " + today.get(Calendar.DAY_OF_YEAR) + "---" + dob.get(Calendar.DAY_OF_YEAR));

        if (today.get(Calendar.DAY_OF_YEAR) < dob.get(Calendar.DAY_OF_YEAR)) {
            age--;
        }


        Integer ageInt = new Integer(age);
        String ageS = ageInt.toString();
        return ageS;
    }
}
```

Figure 6.10

The figure above shows the codes for the AddDoctorActivity. It is where the admin adds doctors into the database using registerDoctor() function. It uses similar logic as the register account for users which is createUserWithEmailAndPassoword. However, the user type will be ‘doctor’ and if the doctor is registered successfully, it will display the toast message “Doctor Added Successfully”.

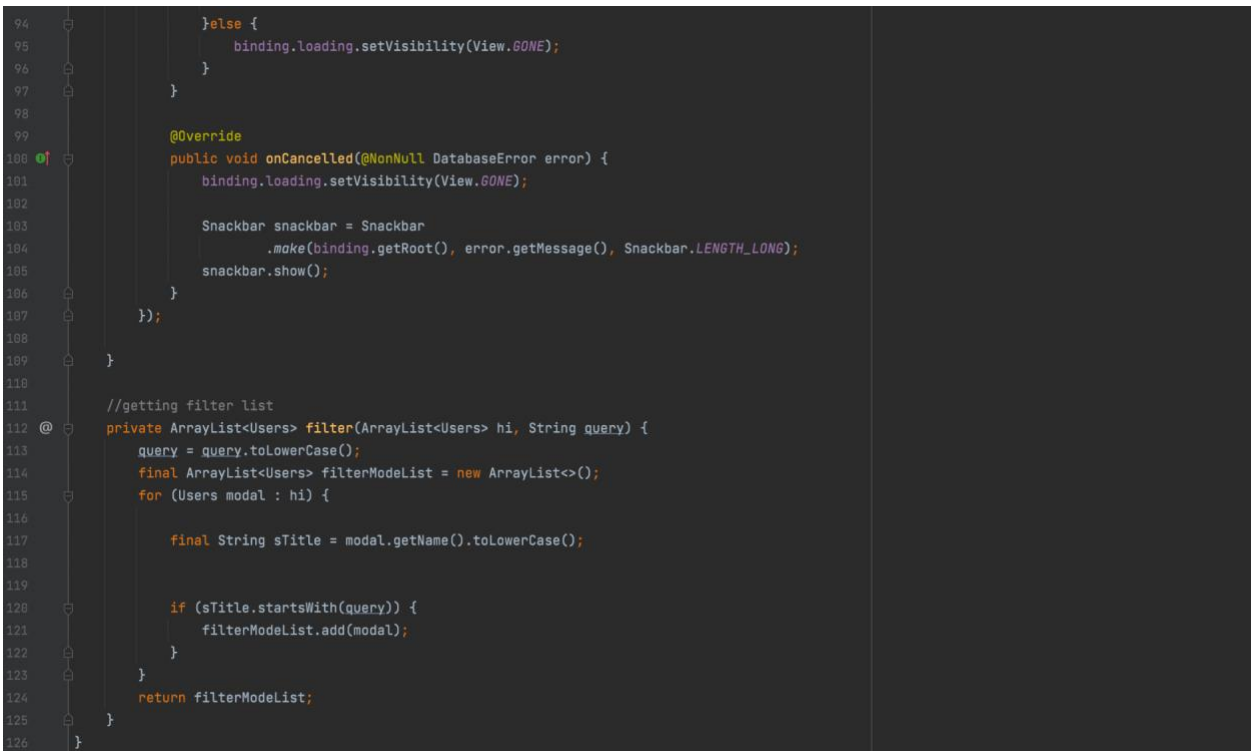
DoctorActivity.java

UserActivity.java



```java
public class DoctorsActivity extends AppCompatActivity {
    private ActivityDoctorsBinding binding;
    private ArrayList<Users> usersArrayList = new ArrayList<>();
    private UserAdapter mAdapter;
    private FirebaseAuth mAuth;
    private DatabaseReference userRef;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityDoctorsBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        //initialize
        mAuth = FirebaseAuth.getInstance();
        userRef = FirebaseDatabase.getInstance().getReference( path: "users");
        binding.backBtn.setOnClickListener(v -> onBackPressed());

        binding.addDoctorBtn.setOnClickListener(v -> startActivity(new Intent( packageContext: DoctorsActivity.this, AddDoctorActivity.class)));

        binding.searchEt.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

            }
            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {
                if (!binding.searchEt.getText().toString().isEmpty()) {
                    final ArrayList<Users> filterModList = filter(usersArrayList, binding.searchEt.getText().toString());
                    if (!filterModList.isEmpty())
                        mAdapter.setFilter(filterModList);
                } else {
                    getDoctors();
                }
```

```
        @Override
        public void afterTextChanged(Editable editable) {

        }
    });

}

@Override
protected void onStart() {
    super.onStart();
    getDoctors();
}

private void getDoctors() {
    usersArrayList.clear();
    binding.loading.setVisibility(View.VISIBLE);
    userRef.orderByChild("type").equalTo("doctor").addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            if (snapshot.exists()) {
                binding.loading.setVisibility(View.GONE);

                for (DataSnapshot ds: snapshot.getChildren()){
                    Users users = ds.getValue(Users.class);
                    usersArrayList.add(users);

                }
                mAdapter = new UserAdapter( context: DoctorsActivity.this, usersArrayList);
                binding.doctorRv.setLayoutManager(new LinearLayoutManager( context: DoctorsActivity.this));
                binding.doctorRv.setAdapter(mAdapter);
```



```
            }else {
                binding.loading.setVisibility(View.GONE);
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            binding.loading.setVisibility(View.GONE);

            Snackbar snackbar = Snackbar
                    .make(binding.getRoot(), error.getMessage(), Snackbar.LENGTH_LONG);
            snackbar.show();
        }
    });

}

//getting filter list
private ArrayList<Users> filter(ArrayList<Users> hi, String query) {
    query = query.toLowerCase();
    final ArrayList<Users> filterModeList = new ArrayList<>();
    for (Users modal : hi) {

        final String sTitle = modal.getName().toLowerCase();


        if (sTitle.startsWith(query)) {
            filterModeList.add(modal);
        }
    }
    return filterModeList;
}
}
```

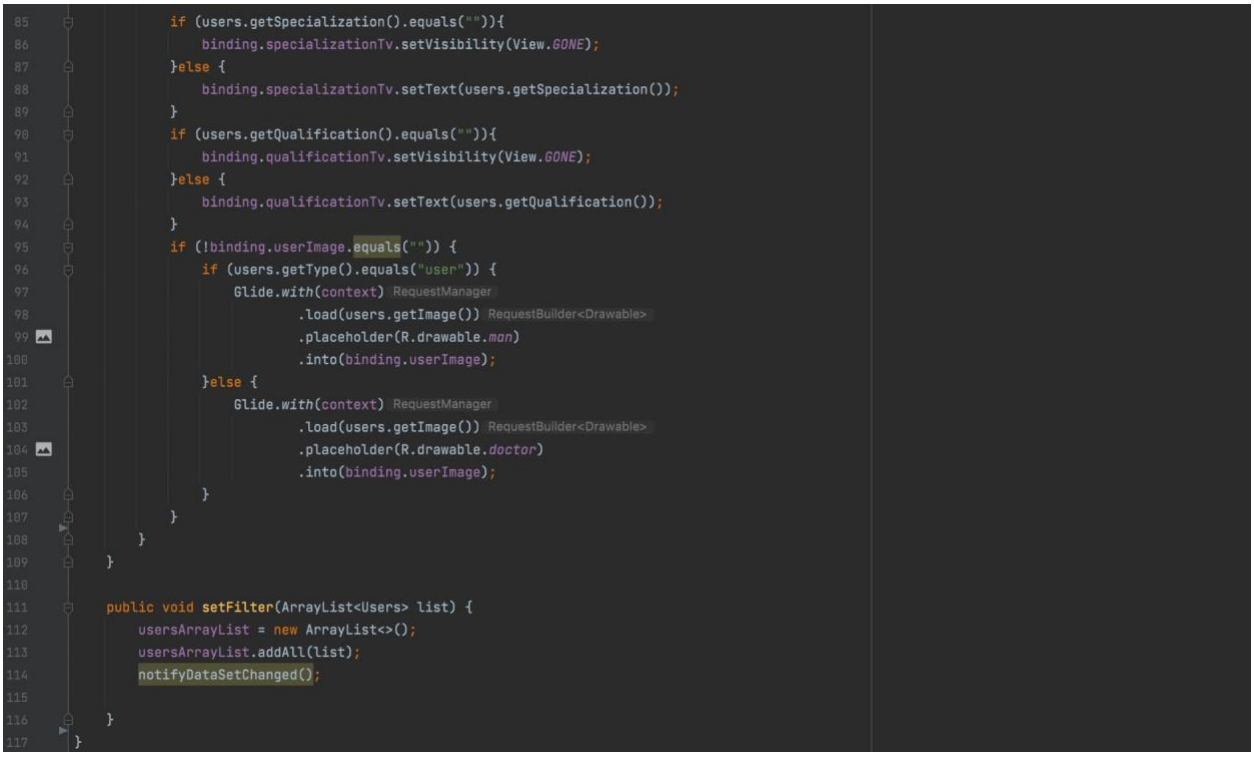
Figure 6.11

The figure above shows the codes for the DoctorActivity but UserActivity is similar. It is the main activity where it handles doctors and users in the admin page. getDoctors() function is used to fetch data from the database and display them into the array. Similarly for UserActivity, getUsers() function is used to fetch data from the database and display them into the array.

UserAdapter.java

```java
public class UserAdapter extends RecyclerView.Adapter<UserAdapter.ViewHolder> {
    private Context context;
    ArrayList<Users> usersArrayList;
    OnItemClickListener mListener;

    public UserAdapter(Context context, ArrayList<Users> usersArrayList) {
        this.context = context;
        this.usersArrayList = usersArrayList;
    }

    public interface OnItemClickListener {
        void onListItemClick(Users users);

    }

    public void setOnItemClickListener(OnItemClickListener listener) { mListener = listener; }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        CustomUsersLayoutBinding binding = CustomUsersLayoutBinding
                .inflate(
                        LayoutInflater.from(parent.getContext()),
                        parent,
                        attachToParent: false);

        return new ViewHolder(binding);
    }
```

```java
    @Override
    public void onBindViewHolder(@NonNull final ViewHolder holder, int position) {
        Users users = usersArrayList.get(position);
        holder.bind(users);
    }

    @Override
    public int getItemCount() { return usersArrayList.size(); }

    public class ViewHolder extends RecyclerView.ViewHolder {
        private final CustomUsersLayoutBinding binding;

        public ViewHolder(CustomUsersLayoutBinding binding) {
            super(binding.getRoot());
            this.binding = binding;

            binding.bookCardView.setOnClickListener(view -> {
                int position = getAdapterPosition();
                if (position != RecyclerView.NO_POSITION) {
                    Users users = usersArrayList.get(position);
                    mListener.onListItemClick(users);
                }
            });
        }

        public void bind(Users users) {
            binding.userName.setText(users.getName());
            binding.userEmail.setText(users.getEmail());
            binding.userPhone.setText(users.getPhone());

            if(users.getType().equals("doctor")){
                binding.bookCardView.setVisibility(View.VISIBLE);
            }
```



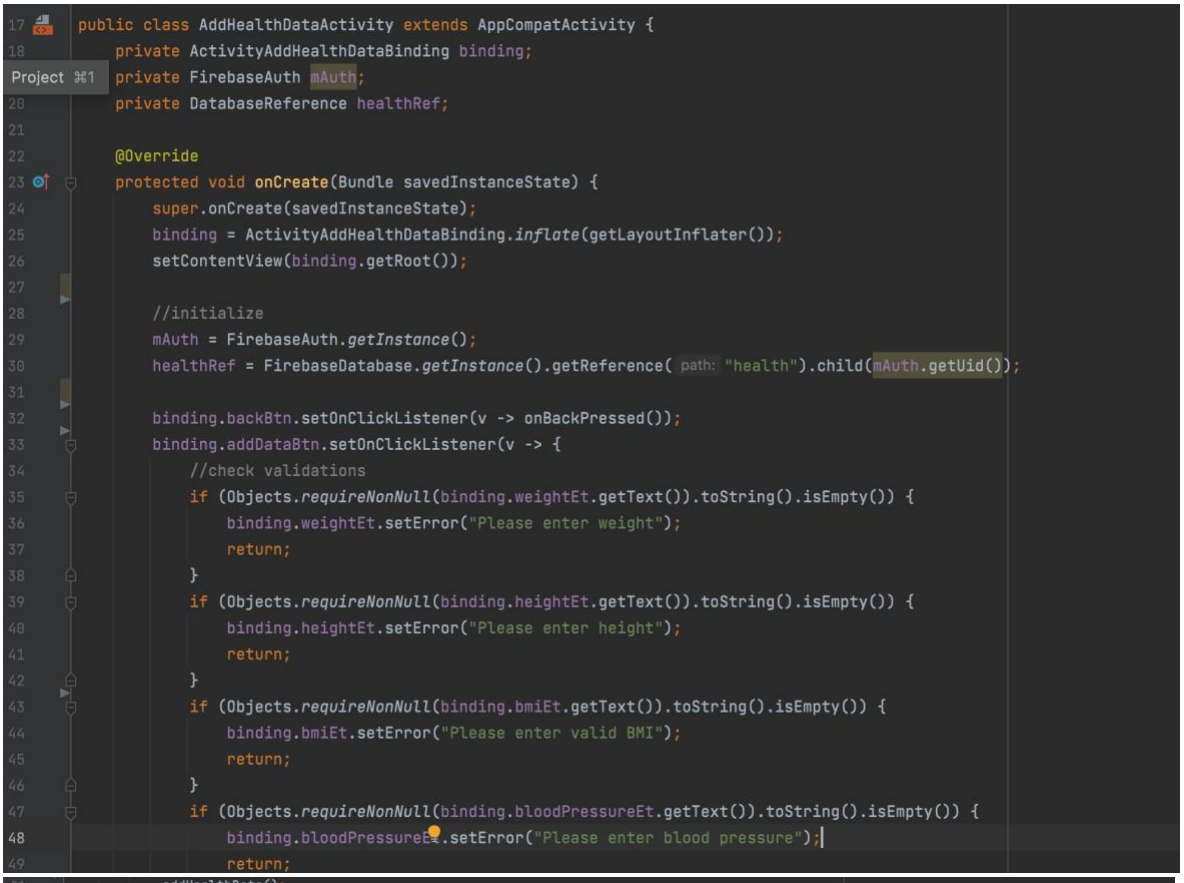
```
            if (users.getSpecialization().equals("")){
                binding.specializationTv.setVisibility(View.GONE);
            }else {
                binding.specializationTv.setText(users.getSpecialization());
            }
            if (users.getQualification().equals("")){
                binding.qualificationTv.setVisibility(View.GONE);
            }else {
                binding.qualificationTv.setText(users.getQualification());
            }
            if (!binding.userImage.equals("")) {
                if (users.getType().equals("user")) {
                    Glide.with(context) RequestManager
                            .load(users.getImage()) RequestBuilder<Drawable>
                            .placeholder(R.drawable.man)
                            .into(binding.userImage);
                }else {
                    Glide.with(context) RequestManager
                            .load(users.getImage()) RequestBuilder<Drawable>
                            .placeholder(R.drawable.doctor)
                            .into(binding.userImage);
                }
            }
        }
    }

    public void setFilter(ArrayList<Users> list) {
        usersArrayList = new ArrayList<>();
        usersArrayList.addAll(list);
        notifyDataSetChanged();

    }
}
```

Figure 6.12

The figure above shows the codes for the UserAdapter. It is to display the users information from the array into their respective cardview using a viewholder.

4.2.5 Health Data

AddHealthDataActivity.java

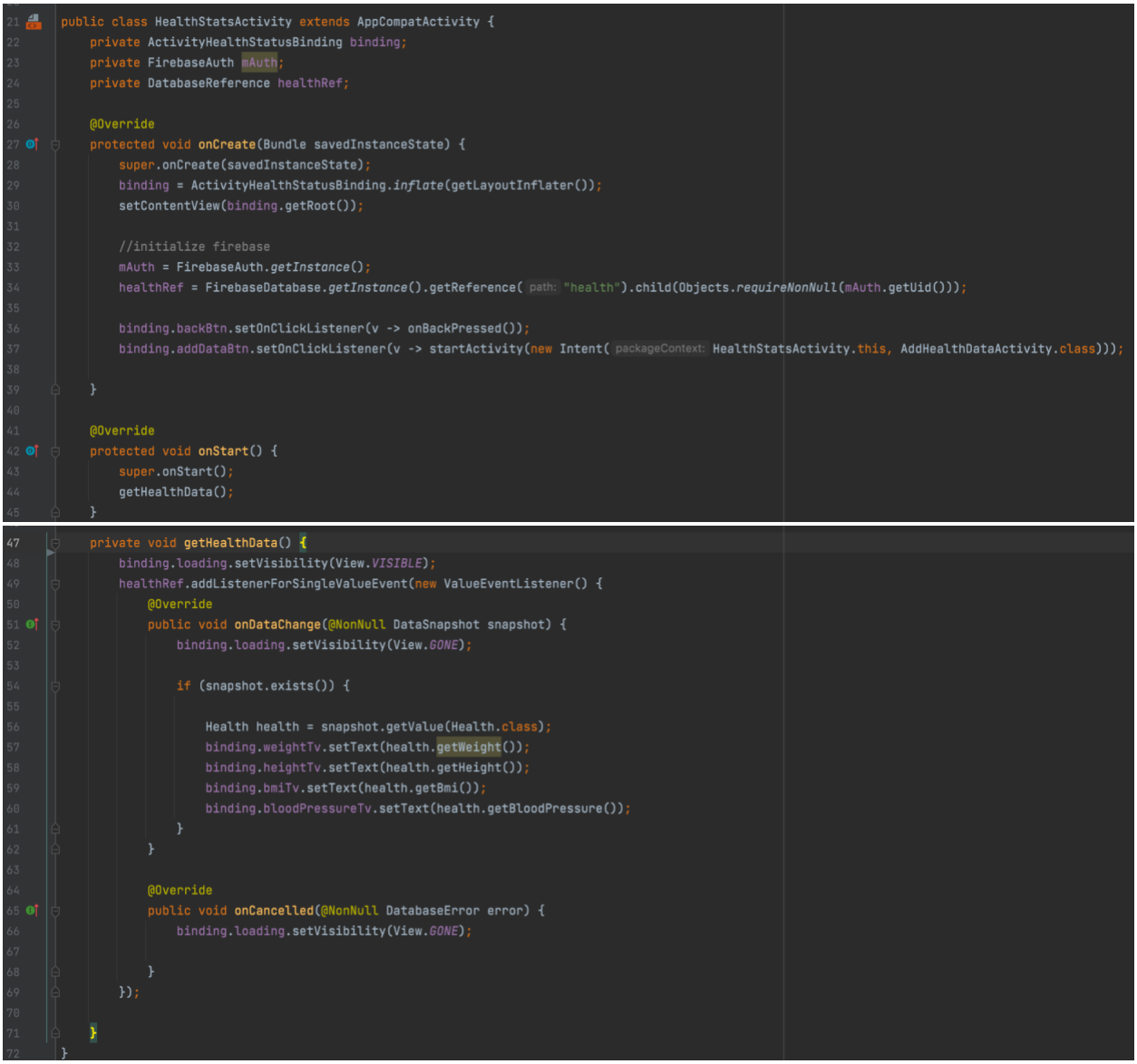


```java
public class AddHealthDataActivity extends AppCompatActivity {
    private ActivityAddHealthDataBinding binding;
    private FirebaseAuth mAuth;
    private DatabaseReference healthRef;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityAddHealthDataBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        //initialize
        mAuth = FirebaseAuth.getInstance();
        healthRef = FirebaseDatabase.getInstance().getReference( path: "health").child(mAuth.getUid());

        binding.backBtn.setOnClickListener(v -> onBackPressed());
        binding.addDataBtn.setOnClickListener(v -> {
            //check validations
            if (Objects.requireNonNull(binding.weightEt.getText()).toString().isEmpty()) {
                binding.weightEt.setError("Please enter weight");
                return;
            }
            if (Objects.requireNonNull(binding.heightEt.getText()).toString().isEmpty()) {
                binding.heightEt.setError("Please enter height");
                return;
            }
            if (Objects.requireNonNull(binding.bmiEt.getText()).toString().isEmpty()) {
                binding.bmiEt.setError("Please enter valid BMI");
                return;
            }
            if (Objects.requireNonNull(binding.bloodPressureEt.getText()).toString().isEmpty()) {
                binding.bloodPressureEt.setError("Please enter blood pressure");
                return;
            }
            addHealthData();
        });
    }

    private void addHealthData() {
        binding.loading.setVisibility(View.VISIBLE);

        Health health = new Health(
                binding.weightEt.getText().toString(),
                binding.heightEt.getText().toString(),
                binding.bmiEt.getText().toString(),
                binding.bloodPressureEt.getText().toString()
        );

        healthRef.setValue(health)
                .addOnCompleteListener(task -> {
                    if (task.isSuccessful()) {
                        binding.loading.setVisibility(View.GONE);

                        Toast.makeText( context: AddHealthDataActivity.this, text: "Health data added successfully", Toast.LENGTH_SHORT).show();
                        onBackPressed();

                    }
                }).addOnFailureListener(e -> {
                    binding.loading.setVisibility(View.GONE);
                    Toast.makeText( context: AddHealthDataActivity.this, e.getMessage(), Toast.LENGTH_SHORT);
                });

    }
}
```

Figure 6.13

The figure above shows the codes for the AddHealthDataActivity. It is the activity that allows the users to add their health data in the application using addHealthData() function.

HealthStatsActivity.java



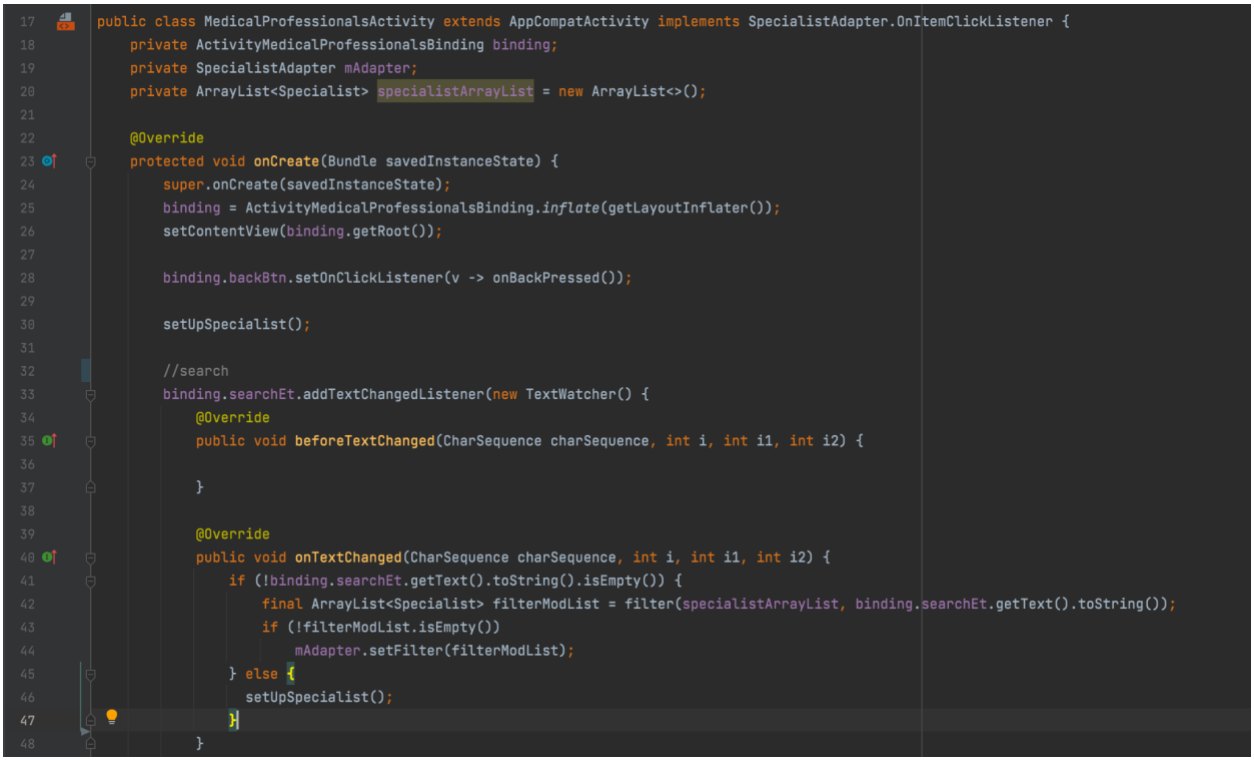
```java
public class HealthStatsActivity extends AppCompatActivity {
    private ActivityHealthStatusBinding binding;
    private FirebaseAuth mAuth;
    private DatabaseReference healthRef;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityHealthStatusBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        //initialize firebase
        mAuth = FirebaseAuth.getInstance();
        healthRef = FirebaseDatabase.getInstance().getReference( path: "health").child(Objects.requireNonNull(mAuth.getUid()));

        binding.backBtn.setOnClickListener(v -> onBackPressed());
        binding.addDataBtn.setOnClickListener(v -> startActivity(new Intent( packageContext: HealthStatsActivity.this, AddHealthDataActivity.class)));

    }

    @Override
    protected void onStart() {
        super.onStart();
        getHealthData();
    }

    private void getHealthData() {
        binding.loading.setVisibility(View.VISIBLE);
        healthRef.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                binding.loading.setVisibility(View.GONE);

                if (snapshot.exists()) {

                    Health health = snapshot.getValue(Health.class);
                    binding.weightTv.setText(health.getWeight());
                    binding.heightTv.setText(health.getHeight());
                    binding.bmiTv.setText(health.getBmi());
                    binding.bloodPressureTv.setText(health.getBloodPressure());
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                binding.loading.setVisibility(View.GONE);

            }
        });

    }
}
```

Figure 6.14

The figure above shows the codes for the HealthStatsActivity. It is the main activity that handles the health data feature in the application. If the current user id is validated, it will display the health data accordingly. To simplify the fetching of data, Health Class is created where we can getWeight(), getHeight, getBMI() and getBloodPressure() just by accessing the Class.

4.2.7 Appointment

MedicalProfessionalsActivity.java



```java
public class MedicalProfessionalsActivity extends AppCompatActivity implements SpecialistAdapter.OnItemClickListener {
    private ActivityMedicalProfessionalsBinding binding;
    private SpecialistAdapter mAdapter;
    private ArrayList<Specialist> specialistArrayList = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMedicalProfessionalsBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        binding.backBtn.setOnClickListener(v -> onBackPressed());

        setUpSpecialist();

        //search
        binding.searchEt.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

            }

            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {
                if (!binding.searchEt.getText().toString().isEmpty()) {
                    final ArrayList<Specialist> filterModList = filter(specialistArrayList, binding.searchEt.getText().toString());
                    if (!filterModList.isEmpty())
                        mAdapter.setFilter(filterModList);
                } else {
                  setUpSpecialist();
                }
            }
```

```
        @Override
        public void afterTextChanged(Editable editable) {
        }
    });
}

private void setUpSpecialist() {
    specialistArrayList.clear();
    specialistArrayList.add(new Specialist( id: "1",  title: "Anaesthesiology"));
    specialistArrayList.add(new Specialist( id: "2",  title: "Cardiology"));
    specialistArrayList.add(new Specialist( id: "3",  title: "Cardiovascular/Thoracic Surgery"));
    specialistArrayList.add(new Specialist( id: "4",  title: "Clinical Immunology/Allergy"));
    specialistArrayList.add(new Specialist( id: "5",  title: "Dentistry"));
    specialistArrayList.add(new Specialist( id: "6",  title: "Dermatology"));
    specialistArrayList.add(new Specialist( id: "7",  title: "Gastroenterology"));
    specialistArrayList.add(new Specialist( id: "7",  title: "General Surgery"));
    specialistArrayList.add(new Specialist( id: "8",  title: "General/Clinical Pathology"));
    specialistArrayList.add(new Specialist( id: "9",  title: "Gynecology"));
    specialistArrayList.add(new Specialist( id: "10",  title: "Otolaryngology"));
    specialistArrayList.add(new Specialist( id: "11",  title: "Physician"));
    specialistArrayList.add(new Specialist( id: "12",  title: "Plastic Surgery"));
    specialistArrayList.add(new Specialist( id: "13",  title: "Psychiatry"));
    specialistArrayList.add(new Specialist( id: "14",  title: "Neurology"));
    specialistArrayList.add(new Specialist( id: "15",  title: "Urology"));

    mAdapter = new SpecialistAdapter( context: this, specialistArrayList);
    binding.specialistRv.setLayoutManager(new GridLayoutManager( context: this,  spanCount: 2));
    binding.specialistRv.setAdapter(mAdapter);
    mAdapter.setOnItemClickListener(this);
}


@Override
public void onListItemClick(Specialist specialist) {
    Intent intent = new Intent( packageContext: MedicalProfessionalsActivity.this, ProfessionalsActivity.class);
    intent.putExtra( name: "title", specialist.getTitle());
    startActivity(intent);
}

//getting filter list
private ArrayList<Specialist> filter(ArrayList<Specialist> hi, String query) {
    query = query.toLowerCase();
    final ArrayList<Specialist> filterModeList = new ArrayList<>();
    for (Specialist modal : hi) {

        final String sTitle = modal.getTitle().toLowerCase();

        if (sTitle.startsWith(query)) {
            filterModeList.add(modal);
        }
    }
    return filterModeList;
}

}
```

Figure 6.15

The figure above shows the codes for the MedicalProfessionalsActivity. It is the main activity that handles the viewing of medical professionals featured in the application. To set up the department or specialisation, setUpSpeacialist() is implemented where they are hard-coded and stored into an array due to time constraints.

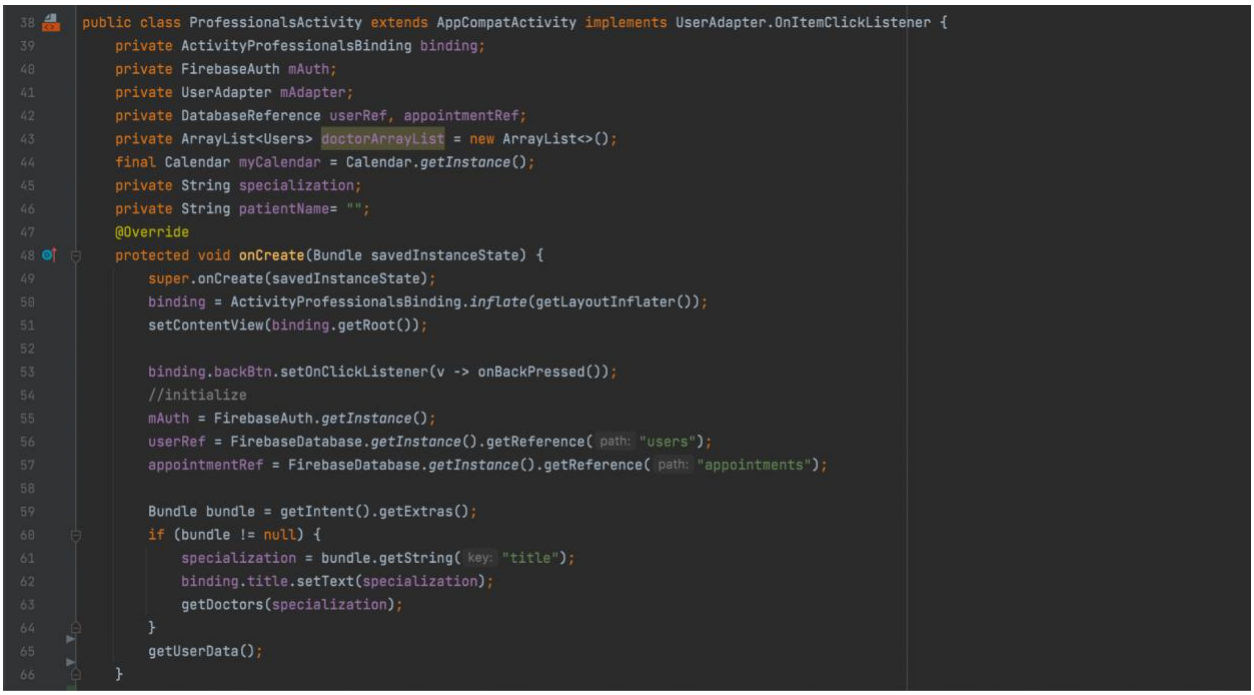
SpecialistAdapter.java

```
public class SpecialistAdapter extends RecyclerView.Adapter<SpecialistAdapter.ViewHolder> {
    private Context context;
    ArrayList<Specialist> specialistArrayList;
    OnItemClickListener mListener;

    public SpecialistAdapter(Context context, ArrayList<Specialist> specialistArrayList) {
        this.context = context;
        this.specialistArrayList = specialistArrayList;
    }

    public interface OnItemClickListener {
        void onListItemClick(Specialist home);
    }

    public void setOnItemClickListener(OnItemClickListener listener) { mListener = listener; }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        CustomSpecialistLayoutBinding binding = CustomSpecialistLayoutBinding
                .inflate(
                        LayoutInflater.from(parent.getContext()),
                        parent,
                        attachToParent: false);

        return new ViewHolder(binding);
    }

    @Override
    public void onBindViewHolder(@NonNull final ViewHolder holder, int position) {
        Specialist home = specialistArrayList.get(position);
        holder.bind(home);
    @Override
    public int getItemCount() { return specialistArrayList.size(); }

    public class ViewHolder extends RecyclerView.ViewHolder {
        private final CustomSpecialistLayoutBinding binding;

        public ViewHolder(CustomSpecialistLayoutBinding binding) {
            super(binding.getRoot());
            this.binding = binding;

            binding.linear.setOnClickListener(view -> {
                int position = getAdapterPosition();
                if (position != RecyclerView.NO_POSITION) {
                    Specialist home = specialistArrayList.get(position);
                    mListener.onListItemClick(home);
                }
            });
        }

        public void bind(Specialist home) {
            binding.title.setText(home.getTitle());
        }
    }

    public void setFilter(ArrayList<Specialist> list) {
        specialistArrayList = new ArrayList<>();
        specialistArrayList.addAll(list);
        notifyDataSetChanged();
    }
}
```

Figure 6.16

The figure above shows the codes for the SpecialistAdapter. It is to display the specialists/doctors a recyclerview using a viewholder.

ProfessionalsActivity.java

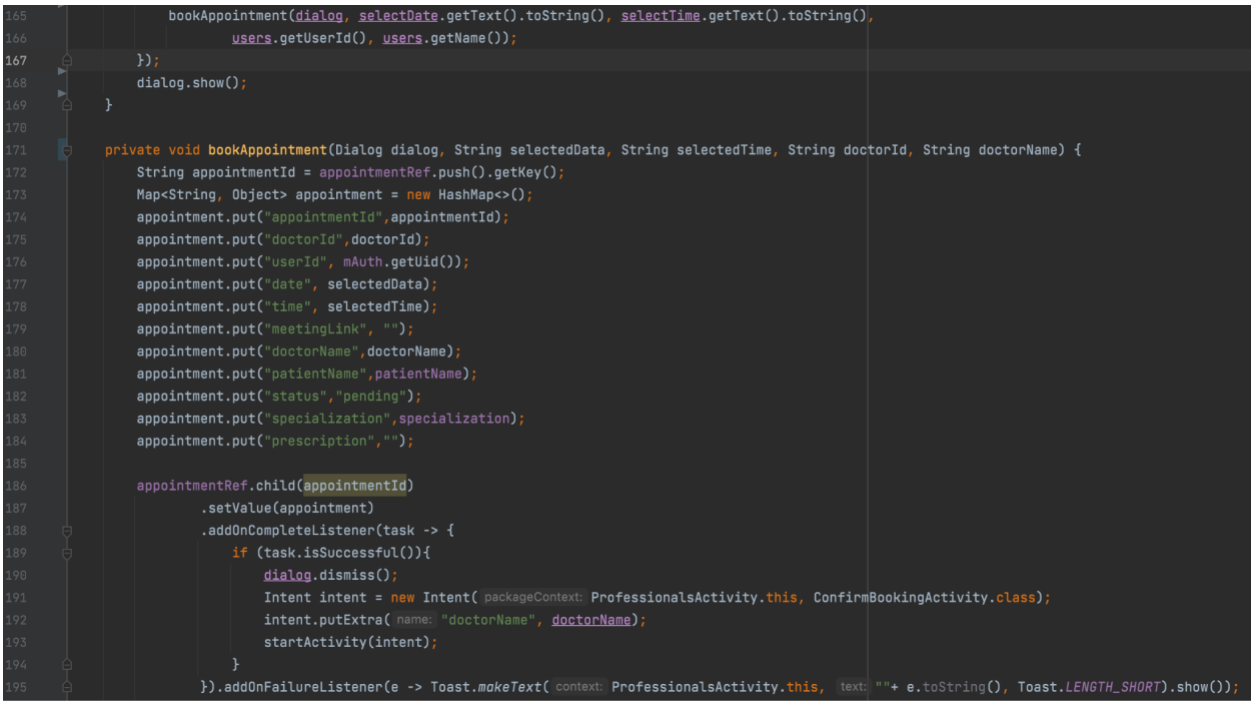


```
public class ProfessionalsActivity extends AppCompatActivity implements UserAdapter.OnItemClickListener {
    private ActivityProfessionalsBinding binding;
    private FirebaseAuth mAuth;
    private UserAdapter mAdapter;
    private DatabaseReference userRef, appointmentRef;
    private ArrayList<Users> doctorArrayList = new ArrayList<>();
    final Calendar myCalendar = Calendar.getInstance();
    private String specialization;
    private String patientName= "";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityProfessionalsBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        binding.backBtn.setOnClickListener(v -> onBackPressed());
        //initialize
        mAuth = FirebaseAuth.getInstance();
        userRef = FirebaseDatabase.getInstance().getReference( path: "users");
        appointmentRef = FirebaseDatabase.getInstance().getReference( path: "appointments");

        Bundle bundle = getIntent().getExtras();
        if (bundle != null) {
            specialization = bundle.getString( key: "title");
            binding.title.setText(specialization);
            getDoctors(specialization);
        }
        getUserData();
    }
```

```
    private void getUserData() {
        userRef.child(Objects.requireNonNull(mAuth.getUid()))
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot snapshot) {
                        if (snapshot.exists()) {
                            Users users = snapshot.getValue(Users.class);
                            patientName = users.getName();
                        }
                    }
                    @Override
                    public void onCancelled(@NonNull DatabaseError error) {
                        Snackbar snackbar = Snackbar
                                .make(binding.getRoot(), error.getMessage(), Snackbar.LENGTH_LONG);
                        snackbar.show();
                    }
                });
    }

    private void getDoctors(String specialization) {
        doctorArrayList.clear();
        binding.loading.setVisibility(View.VISIBLE);
        userRef.orderByChild("specialization").equalTo(specialization).addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists()) {
                    binding.loading.setVisibility(View.GONE);

                    for (DataSnapshot ds : snapshot.getChildren()) {
                        Users users = ds.getValue(Users.class);
                        doctorArrayList.add(users);
                        Log.e( tag: "doctorrr", doctorArrayList.toString());
```

```
                    mAdapter = new UserAdapter( context: ProfessionalsActivity.this, doctorArrayList);
                    binding.doctorsRv.setLayoutManager(new LinearLayoutManager( context: ProfessionalsActivity.this));
                    binding.doctorsRv.setAdapter(mAdapter);
                    mAdapter.setOnItemClickListener(ProfessionalsActivity.this);

                } else {
                    binding.loading.setVisibility(View.GONE);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                binding.loading.setVisibility(View.GONE);

                Snackbar snackbar = Snackbar
                        .make(binding.getRoot(), error.getMessage(), Snackbar.LENGTH_LONG);
                snackbar.show();
            }
        });
    }

    @Override
    public void onListItemClick(Users users) {
        final Dialog dialog = new Dialog( context: this);
        dialog.setContentView(R.layout.custom_dialog_layout);

        TextInputEditText selectDate = dialog.findViewById(R.id.booking_date_et);
        TextInputEditText selectTime = dialog.findViewById(R.id.booking_time_et);
        CardView bookingBtn = dialog.findViewById(R.id.book_card_view);
        CardView cancelBtn = dialog.findViewById(R.id.cancel_card_view);
        cancelBtn.setOnClickListener(v -> dialog.dismiss());
```

```
        DatePickerDialog.OnDateSetListener date = (view, year, month, day) -> {
            myCalendar.set(Calendar.YEAR, year);
            myCalendar.set(Calendar.MONTH, month);
            myCalendar.set(Calendar.DAY_OF_MONTH, day);
            String myFormat = "dd/MM/yyyy";
            SimpleDateFormat dateFormat = new SimpleDateFormat(myFormat, Locale.US);
            selectDate.setText(dateFormat.format(myCalendar.getTime()));
        };
        selectDate.setOnClickListener(v -> new DatePickerDialog( context: ProfessionalsActivity.this, date, myCalendar.get(Calendar.YEAR),
                myCalendar.get(Calendar.MONTH), myCalendar.get(Calendar.DAY_OF_MONTH)).show());

        selectTime.setOnClickListener(v -> {
            Calendar mcurrentTime = Calendar.getInstance();
            int hour = mcurrentTime.get(Calendar.HOUR_OF_DAY);
            int minute = mcurrentTime.get(Calendar.MINUTE);
            TimePickerDialog mTimePicker;
            mTimePicker = new TimePickerDialog( context: ProfessionalsActivity.this, (timePicker, selectedHour, selectedMinute) ->
                    selectTime.setText(selectedHour + ":" + selectedMinute), hour, minute,  is24HourView: true);//Yes 24 hour time
            mTimePicker.setTitle("Select Time");
            mTimePicker.show();
        });

        bookingBtn.setOnClickListener(v -> {
            if (selectDate.getText().toString().isEmpty()) {
                selectDate.setError("Please select date");
                return;
            }
            if (selectTime.getText().toString().isEmpty()) {
                selectTime.setError("Please select time");
                return;
            }
```

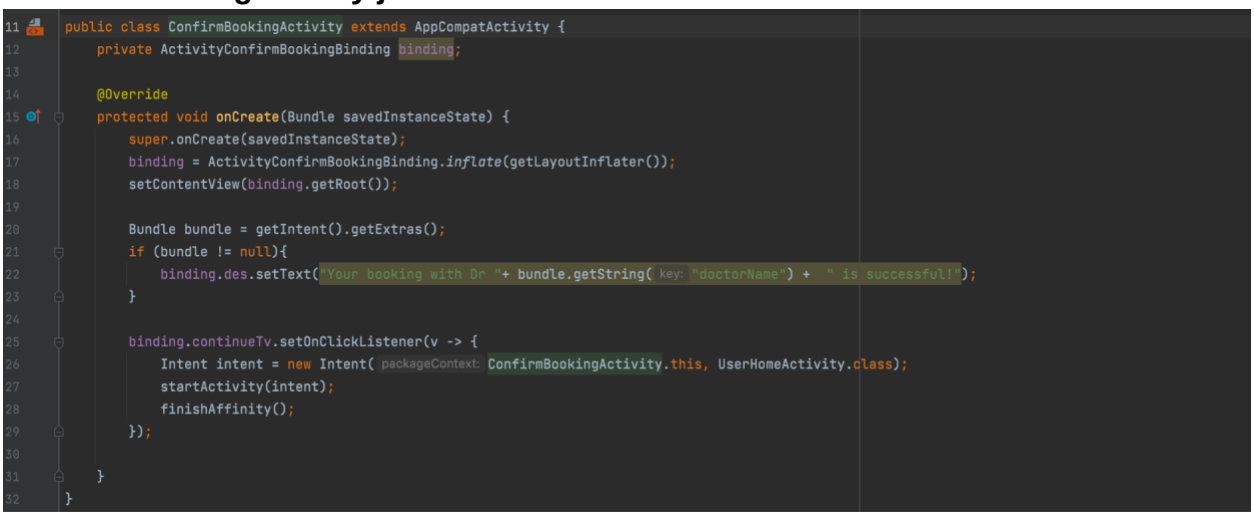


```
            bookAppointment(dialog, selectDate.getText().toString(), selectTime.getText().toString(),
                    users.getUserId(), users.getName());
        });
        dialog.show();
    }

    private void bookAppointment(Dialog dialog, String selectedData, String selectedTime, String doctorId, String doctorName) {
        String appointmentId = appointmentRef.push().getKey();
        Map<String, Object> appointment = new HashMap<>();
        appointment.put("appointmentId",appointmentId);
        appointment.put("doctorId",doctorId);
        appointment.put("userId", mAuth.getUid());
        appointment.put("date", selectedData);
        appointment.put("time", selectedTime);
        appointment.put("meetingLink", "");
        appointment.put("doctorName",doctorName);
        appointment.put("patientName",patientName);
        appointment.put("status","pending");
        appointment.put("specialization",specialization);
        appointment.put("prescription","");

        appointmentRef.child(appointmentId)
                .setValue(appointment)
                .addOnCompleteListener(task -> {
                    if (task.isSuccessful()){
                        dialog.dismiss();
                        Intent intent = new Intent( packageContext: ProfessionalsActivity.this, ConfirmBookingActivity.class);
                        intent.putExtra( name: "doctorName", doctorName);
                        startActivity(intent);
                    }
                }).addOnFailureListener(e -> Toast.makeText( context: ProfessionalsActivity.this,  text: ""+ e.toString(), Toast.LENGTH_SHORT).show());
```

Figure 6.17

The figure above shows the codes for the ProfessionalsActivity. It is the main activity for users to book an appointment with the specialist/medical professional. When the user clicks on the book button in line124, a dialog will show up for the users to enter appointment details such as date and time. bookAppointment() function is implemented to save the data into the database under reference of appointmentRef. Users are then redirected to the confirmation page.

ConfirmBookingActivity.java

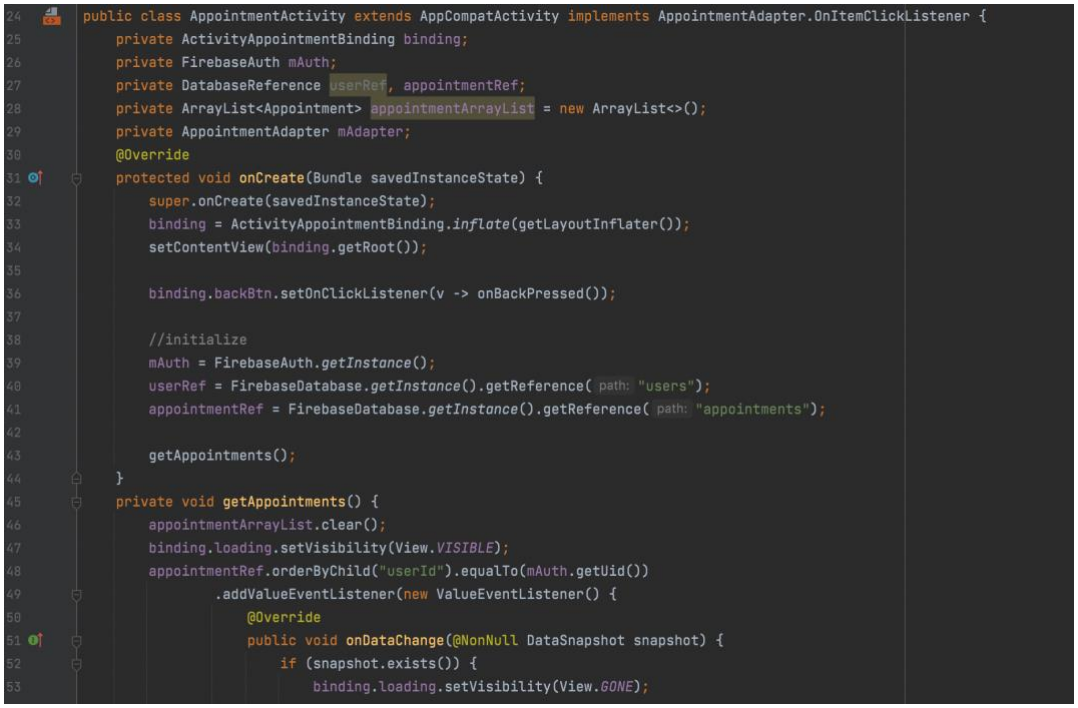


```
public class ConfirmBookingActivity extends AppCompatActivity {
    private ActivityConfirmBookingBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityConfirmBookingBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        Bundle bundle = getIntent().getExtras();
        if (bundle != null){
            binding.des.setText("Your booking with Dr "+ bundle.getString( key: "doctorName") +  " is successful!");
        }

        binding.continueTv.setOnClickListener(v -> {
            Intent intent = new Intent( packageContext: ConfirmBookingActivity.this, UserHomeActivity.class);
            startActivity(intent);
            finishAffinity();
        });

    }
}
```

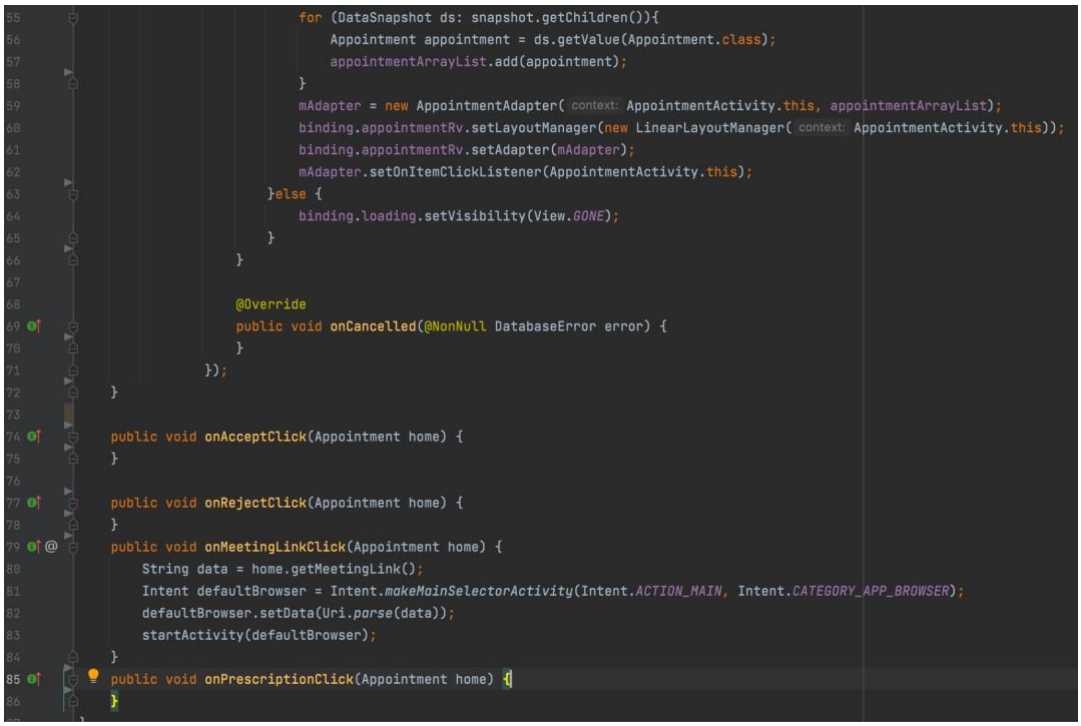
Figure 6.18

The figure above shows the codes for the ConfirmBookingActivity. It is when users are done with the booking and a dialog will show up to indicate the booking is successful and will be redirected to the home page.

AppointmentActivity.java



```
public class AppointmentActivity extends AppCompatActivity implements AppointmentAdapter.OnItemClickListener {
    private ActivityAppointmentBinding binding;
    private FirebaseAuth mAuth;
    private DatabaseReference userRef, appointmentRef;
    private ArrayList<Appointment> appointmentArrayList = new ArrayList<>();
    private AppointmentAdapter mAdapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityAppointmentBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        binding.backBtn.setOnClickListener(v -> onBackPressed());

        //initialize
        mAuth = FirebaseAuth.getInstance();
        userRef = FirebaseDatabase.getInstance().getReference( path: "users");
        appointmentRef = FirebaseDatabase.getInstance().getReference( path: "appointments");

        getAppointments();
    }
    private void getAppointments() {
        appointmentArrayList.clear();
        binding.loading.setVisibility(View.VISIBLE);
        appointmentRef.orderByChild("userId").equalTo(mAuth.getUid())
                .addValueEventListener(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot snapshot) {
                        if (snapshot.exists()) {
                            binding.loading.setVisibility(View.GONE);
```



```
                            for (DataSnapshot ds: snapshot.getChildren()){
                                Appointment appointment = ds.getValue(Appointment.class);
                                appointmentArrayList.add(appointment);
                            }
                            mAdapter = new AppointmentAdapter( context: AppointmentActivity.this, appointmentArrayList);
                            binding.appointmentRv.setLayoutManager(new LinearLayoutManager( context: AppointmentActivity.this));
                            binding.appointmentRv.setAdapter(mAdapter);
                            mAdapter.setOnItemClickListener(AppointmentActivity.this);
                        }else {
                            binding.loading.setVisibility(View.GONE);
                        }
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError error) {
                    }
                });
    }

    public void onAcceptClick(Appointment home) {
    }

    public void onRejectClick(Appointment home) {
    }
    public void onMeetingLinkClick(Appointment home) {
        String data = home.getMeetingLink();
        Intent defaultBrowser = Intent.makeMainSelectorActivity(Intent.ACTION_MAIN, Intent.CATEGORY_APP_BROWSER);
        defaultBrowser.setData(Uri.parse(data));
        startActivity(defaultBrowser);
    }
    public void onPrescriptionClick(Appointment home) {
    }
```

Figure 6.19

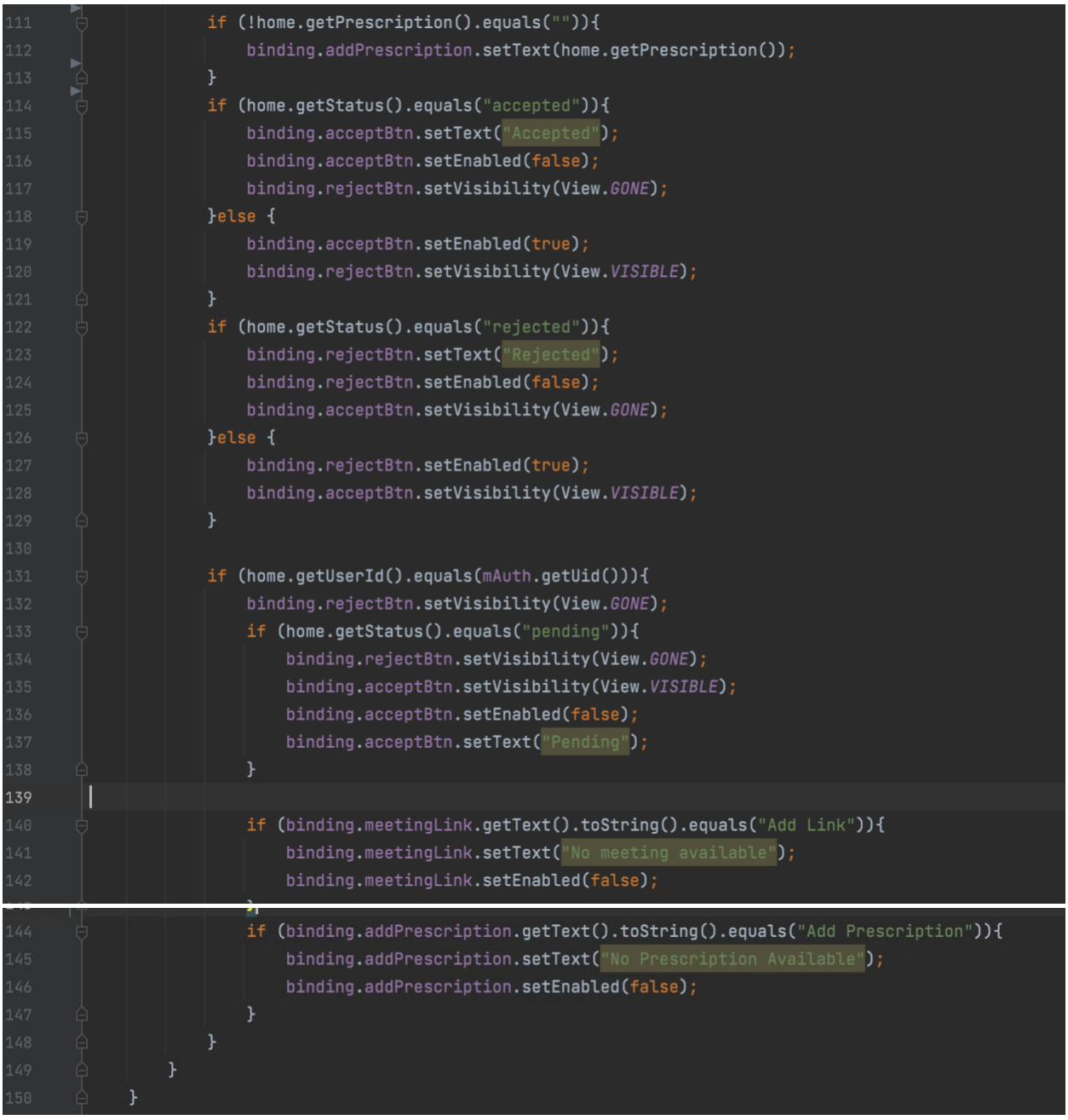
The figure above shows the codes for the AppointmentActivity. It is the main activity where it handles the appointment feature and gets appointments according to the user id.

AppointmentAdapter.java

```java
public class AppointmentAdapter extends RecyclerView.Adapter<AppointmentAdapter.ViewHolder> {
    private Context context;
    ArrayList<Appointment> appointmentArrayList;
    OnItemClickListener mListener;
    private FirebaseAuth mAuth = FirebaseAuth.getInstance();

    public AppointmentAdapter(Context context, ArrayList<Appointment> appointmentArrayList) {

        this.context = context;
        this.appointmentArrayList = appointmentArrayList;
    }

    public interface OnItemClickListener {
        void onAcceptClick(Appointment home);
        void onRejectClick(Appointment home);
        void onMeetingLinkClick(Appointment home);
        void onPrescriptionClick(Appointment home);
    }

    public void setOnItemClickListener(OnItemClickListener listener) { mListener = listener; }


    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        CustomAppointmentLayoutBinding binding = CustomAppointmentLayoutBinding
                .inflate(
                        LayoutInflater.from(parent.getContext()),
                        parent,
                        attachToParent: false);
```

```java
    @Override
    public void onBindViewHolder(@NonNull final ViewHolder holder, int position) {
        Appointment home = appointmentArrayList.get(position);
        holder.bind(home);
    }

    @Override
    public int getItemCount() { return appointmentArrayList.size(); }

    public class ViewHolder extends RecyclerView.ViewHolder {
        private final CustomAppointmentLayoutBinding binding;

        public ViewHolder(CustomAppointmentLayoutBinding binding) {
            super(binding.getRoot());
            this.binding = binding;

            binding.acceptBtn.setOnClickListener(view -> {
                int position = getAdapterPosition();
                if (position != RecyclerView.NO_POSITION) {
                    Appointment home = appointmentArrayList.get(position);
                    mListener.onAcceptClick(home);
                }
            });

            binding.rejectBtn.setOnClickListener(view -> {
                int position = getAdapterPosition();
                if (position != RecyclerView.NO_POSITION) {
                    Appointment home = appointmentArrayList.get(position);
                    mListener.onRejectClick(home);
                }
            });
```

```
78          binding.rejectBtn.setOnClickListener(view -> {
79              int position = getAdapterPosition();
80              if (position != RecyclerView.NO_POSITION) {
81                  Appointment home = appointmentArrayList.get(position);
82                  mListener.onRejectClick(home);
83              }
84          });
85          binding.meetingLink.setOnClickListener(view -> {
86              int position = getAdapterPosition();
87              if (position != RecyclerView.NO_POSITION) {
88                  Appointment home = appointmentArrayList.get(position);
89                  mListener.onMeetingLinkClick(home);
90              }
91          });
92          binding.addPrescription.setOnClickListener(view -> {
93              int position = getAdapterPosition();
94              if (position != RecyclerView.NO_POSITION) {
95                  Appointment home = appointmentArrayList.get(position);
96                  mListener.onPrescriptionClick(home);
97              }
98          });
99      }
100
101 @    public void bind(Appointment home) {
102         binding.patientName.setText(home.getPatientName());
103         binding.doctorName.setText(home.getDoctorName());
104         binding.bookDate.setText(home.getDate());
105         binding.bookTime.setText(home.getTime());
106         binding.bookSpecialization.setText(home.getSpecialization());
107
108         if (!home.getMeetingLink().equals("")){
109             binding.meetingLink.setText(home.getMeetingLink());
110         }
```



```
            if (!home.getPrescription().equals("")){
                binding.addPrescription.setText(home.getPrescription());
            }
            if (home.getStatus().equals("accepted")){
                binding.acceptBtn.setText("Accepted");
                binding.acceptBtn.setEnabled(false);
                binding.rejectBtn.setVisibility(View.GONE);
            }else {
                binding.acceptBtn.setEnabled(true);
                binding.rejectBtn.setVisibility(View.VISIBLE);
            }
            if (home.getStatus().equals("rejected")){
                binding.rejectBtn.setText("Rejected");
                binding.rejectBtn.setEnabled(false);
                binding.acceptBtn.setVisibility(View.GONE);
            }else {
                binding.rejectBtn.setEnabled(true);
                binding.acceptBtn.setVisibility(View.VISIBLE);
            }

            if (home.getUserId().equals(mAuth.getUid())){
                binding.rejectBtn.setVisibility(View.GONE);
                if (home.getStatus().equals("pending")){
                    binding.rejectBtn.setVisibility(View.GONE);
                    binding.acceptBtn.setVisibility(View.VISIBLE);
                    binding.acceptBtn.setEnabled(false);
                    binding.acceptBtn.setText("Pending");
                }

                if (binding.meetingLink.getText().toString().equals("Add Link")){
                    binding.meetingLink.setText("No meeting available");
                    binding.meetingLink.setEnabled(false);
                }
                if (binding.addPrescription.getText().toString().equals("Add Prescription")){
                    binding.addPrescription.setText("No Prescription Available");
                    binding.addPrescription.setEnabled(false);
                }
            }
        }
    }
```

Figure 6.20

The figure above shows the codes for the AppointmentAdapter. The adapter displays the appointments into a recyclerview using a viewholder. It allows doctors to accept or reject the appointments. In the appointment card, details such as patient name, doctor name is fetched from the database by accessing the respective Class.

5.0 Conclusion

To conclude the project and the document, the team has managed to design and implement a mobile application that fulfils the objective of building a system that offers functionalities such as ability to schedule/cancel appointments, the ability to request prescriptions and easy access to medical records. Hopefully with these features, the application would appeal to many people who want to use an application with convenient features like this and vastly improve the field of mobile applications by adhering to these principles of taking note of what people would want in an application. By using the waterfall methodology, we have managed to finish the project in time while making it functional. Some improvements that could be made to the application would be a native video function because currently the video function links to an external site which some users may find cumbersome to use. Besides that, due to time constraints, department/specialisation can be added by the admin instead of hard coding them.

References

References Android Developers. (2018). Get started with the Navigation component | Android Developers. Retrieved August 5, 2022, from Android Developers website: https://developer.android.com/guide/navigation/navigation-getting-started

Android Developers. (2022a). HashMap. Retrieved August 5, 2022, from Android Developers website: https://developer.android.com/reference/java/util/HashMap

Android Developers. (2022b). View Binding. Retrieved August 5, 2022, from Android Developers website: https://developer.android.com/topic/libraries/view-binding

Pandya, S. (2021, December 21). Data Binding in Android: A tutorial with examples. Retrieved August 5, 2022, from LogRocket Blog website: https://blog.logrocket.com/data-binding-android-tutorial-with-examples/

Pew Research Center: Internet, Science & Tech. (2021). *Mobile Fact Sheet*. [online] Available at: https://www.pewresearch.org/internet/fact-sheet/mobile/ [Accessed 9 Aug. 2022].

Aungst, T.D. (2013). Medical Applications for Pharmacists Using Mobile Devices. *Annals of Pharmacotherapy*, [online] 47(7-8), pp.1088–1095. doi:10.1345/aph.1s035.

Mosa, A.S.M., Yoo, I. and Sheets, L. (2012). A Systematic Review of Healthcare Applications for Smartphones. *BMC Medical Informatics and Decision Making*, [online] 12(1). doi:10.1186/1472-6947-12-67.

Batista, M.A. and Gaglani, S.M. (2013). The Future of Smartphones in Health Care. *AMA Journal of Ethics*, [online] 15(11), pp.947–950. doi:10.1001/virtualmentor.2013.15.11.stas1-1311..

georgiou (2022). *Developing a Healthcare App in 2022: What Do Patients Really Want?* [online] Imaginovation | Top Web & Mobile App Development Company Raleigh. Available at: https://imaginovation.net/blog/developing-a-mobile-health-app-what-patients-really-want/#:~:text=What%20Do%20Patients%20Want%20in,easy%20access%20to%20medical%20records. [Accessed 9 Aug. 2022].

TechTarget Contributor (2018). *Android Studio*. [online] SearchMobileComputing. Available at: https://www.techtarget.com/searchmobilecomputing/definition/Android-Studio [Accessed 9 Aug. 2022].

Code Institute Global. (2014). *What is Java and What is it Used For? - Code Institute Global*. [online] Available at: https://codeinstitute.net/global/blog/what-is-java/ [Accessed 9 Aug. 2022].

Stevenson, D. (2018). *What is Firebase? The complete story, abridged. - Firebase Developers - Medium*. [online] Medium. Available at: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0 [Accessed 9 Aug. 2022].

Kinsta®. (2022). *What Is GitHub? A Beginner's Introduction to GitHub*. [online] Available at: https://kinsta.com/knowledgebase/what-is-github/ [Accessed 9 Aug. 2022].

Cousins, C. (2019). *What Is Figma? a 101 Intro*. [online] Designshack.net. Available at: https://designshack.net/articles/software/what-is-figma-intro/ [Accessed 9 Aug. 2022].