CS2043, Project 1, Software System

Health and Fitness monitoring system

Team profile

- Mohamed Mesbahi El Aouame, UNBSJ BscCs student, Third year student at UNBSJ. C++ Programmer and AI enthusiast. Interested in embedded systems development in the domain of IoT, as well as neural networking and deep learning techniques.
- Michael Guo, UNB BscCs student
- Lindsay Mullett, UNBSJ BscCs student. Educated in data management and object orientated programming. Learning proper software development. Motivated to improve implementation and efficiency of systems relating to precision medicine.

For the final project in CS2043 with Dr.Kim, we were tasked with creatively identifying and implementing a solution to a complex problem that exists within the domain of ICT. It’s clear to anyone paying attention that our healthcare system is not sustainable. The possibility to predict diseases or to warn individuals of their progressions before it is a severe problem would help reduce strain on the health care system. To personalize a menu, measure the amount of activity with better accuracy, and finally, monitoring and advice on current health status would greatly impact the quality of life for those individuals with elevated care requirements.

Creating a healthcare monitoring system allowed us to work as an effective team to implement a software based solution that would deliver measurable value to the industry.

Overview of the System

Client – Upon purchasing a wearable device (eg.smart watch) user will upload personal data to be monitored and measured for self-improvement, health monitoring and medical analysis. User will be identified through a unique identification number and password to create account and assess own data. The wearable device will measure vitals and record physical data. There will be different options for users to subscribe to. High risk individuals such as diabetics or those with genetic disorders will be able to have their data closely monitored by their health care provider and have automatic alerts if vitals reports are concerning. In the event of a medical emergency, a health care professional will be able to access data off of the user’s device such as vitals, medical history, medications, allergies and blood type. Those wishing to have guidance for self-improvement will subscribe to coaches for fitness and nutrition. Through that they will submit reports to have analyzed then receive feedback and modifications from their coaches.

System – Website and app connected to user's wearable device. Device will measure physical data and store it with the client’s profile accessible through a website. User friendly interface for client to upload and monitor data. Interface for healthcare professionals to connect to in the event of urgent medical care. Available connection for fitness and nutrition coaches to check progress and submit changes to recommendations. System will notify health care professionals of changes that may be concerning. Alerts will be issued to the user if blood sugar or heart rates are at dangerous levels. It is required to keep medical information and personal information separate. The system will include a specialized database that allows the storage of dynamic information for clients.
This type of system is targeted at many stakeholders. Users ca be split into various categories including:

- Health care professionals
- Fitness motivated individuals
- Diabetics
- Individuals with genetic disorders
- Personal Trainers
- High risk individuals
- Nutritionists

Due to time constraints, we decided to use the Agile approach for this project. The first phase consisted of finding the right functional and non-functional system requirements along with a large set of target actors. The functional system requirements we outlined as team (refer to table 1 below). Understanding the requirements allowed us to focus on the effective implementation of our system. Although with the time available we were only able to implement a small portion of what the final system would be, we have an outline to guide future development while expecting it to be subject to more changes. We did spend a large chunk of time coping with the requirements specification (we specified some requirements that we did not assess later) as well as the system architecture, leaving us with little time for the implementation phase. However, we managed to build a small prototype for both the website and the Android Application.

Project Organization

A Gantt chart was created so we could organize our schedule and progress (refer to figure 1 below). Then, an array of use-case scenarios was defined following an assessment of the viewpoints of all stakeholders. Through our system, various users will be able to communicate about all their health and wellness concerns. We were able to demonstrate these connections through our use cases (refer to table 2 below). These scenarios were curated and merged to provide us with some grounding for the implementation and testing of our first prototypes. A use care diagram was created for a visual representation of users interacting with the database (refer to figure 2 below). Creating a class diagram was chosen to illustrate how different entities would work together (refer to figure 3). After creating diagrams to visualize the design, we were able to begin the construction of our system.

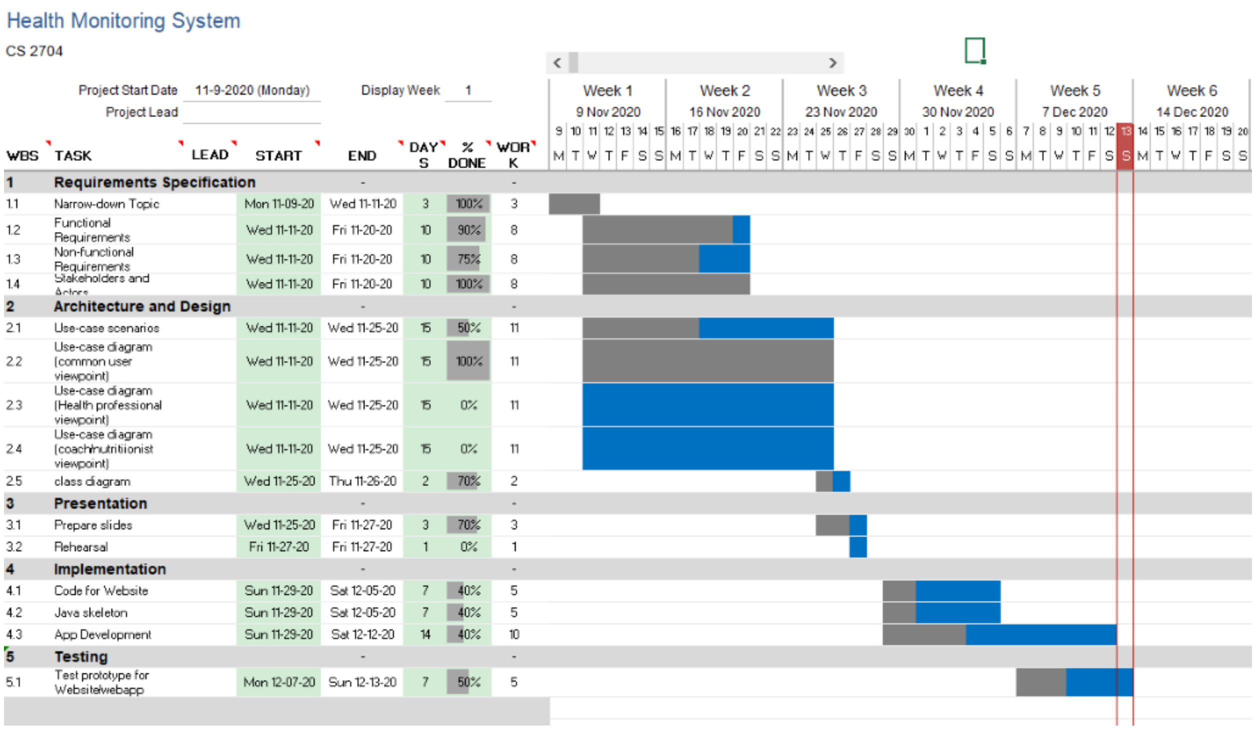
Table 1: Functional System Requirements of health monitoring system listed with description of requirement and the priority rating on a scale of 1-5.

	Description	Priority
REQ-1	The system should allow the user to create his own profile and modify it	5
REQ-2	The system should be able to store the personal health data uploaded by users from any machine(computer/mobile) into a database	5
REQ-3	The system should provide 4 types of	3

	accounts: one for health professionals, one for personal coaches and nutritionists, one for regular users and one for users at risk (people with diabetes, genetic conditions, chronic diseases...)	
REQ-4	For the regular users account, the system should have these features: Fitness monitoring Dietary guidance Health monitoring	2
REQ-5	Connecting actors to the user. Allowing the diverse types of professionals to modify plans and provide recommendations to users.	2
REQ-6	System will allow for user's vital information to be accessed by health care professional in the event of a medical emergency	5
REQ-7	System will monitor vitals of user and send alert when there is a dramatic change in blood sugars or heart rate	3
REQ-8	Access personal data about the client health presented as formalized	2

	electronic medical records that enable displaying the current integral evaluation of the health state	
REQ-9	The system will include a specialized database that allows storage of dynamic information of clients.	1
REQ-10	System will connect to other companies' technologies that monitor heart rate, blood pressure and blood sugar	2
REQ-11	Encapsulation to ensure only authorized individuals access sensitive information	1
REQ-12	User interface for webapp/website should be user-friendly and intuitive	3
REQ-13	User interface for Android app should be user-friendly and intuitive	3
REQ-14	System should not give access to the main database to regular users.	4

REQ-15	System should not allow the user to access the account for another user	4
REQ-16	A maximum of 2 superusers are allowed.	2



Health Monitoring System

CS 2704

Project Start Date 11-9-2020 (Monday) Display Week 1

Project Lead

WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK
1	**Requirements Specification**			-			-
1.1	Narrow-down Topic		Mon 11-09-20	Wed 11-11-20	3	100%	3
1.2	Functional Requirements		Wed 11-11-20	Fri 11-20-20	10	90%	8
1.3	Non-functional Requirements		Wed 11-11-20	Fri 11-20-20	10	75%	8
1.4	Stakeholders and Actors		Wed 11-11-20	Fri 11-20-20	10	100%	8
2	**Architecture and Design**			-			-
2.1	Use-case scenarios		Wed 11-11-20	Wed 11-25-20	15	50%	11
2.2	Use-case diagram (common user viewpoint)		Wed 11-11-20	Wed 11-25-20	15	100%	11
2.3	Use-case diagram (Health professional viewpoint)		Wed 11-11-20	Wed 11-25-20	15	0%	11
2.4	Use-case diagram (coach/nutritiionist viewpoint)		Wed 11-11-20	Wed 11-25-20	15	0%	11
2.5	class diagram		Wed 11-25-20	Thu 11-26-20	2	70%	2
3	**Presentation**			-			-
3.1	Prepare slides		Wed 11-25-20	Fri 11-27-20	3	70%	3
3.2	Rehearsal		Fri 11-27-20	Fri 11-27-20	1	0%	1
4	**Implementation**			-			-
4.1	Code for Website		Sun 11-29-20	Sat 12-05-20	7	40%	5
4.2	Java skeleton		Sun 11-29-20	Sat 12-05-20	7	40%	5
4.3	App Development		Sun 11-29-20	Sat 12-12-20	14	40%	10
5	**Testing**			-			-
5.1	Test prototype for Website/webapp		Mon 12-07-20	Sun 12-13-20	7	50%	5

Figure 1: Gantt chart to monitor scheduling and progress

Table 2: Use-case scenarios

Name	Description	Satisfied Requirement
Client creating profile	Allows user to create his/her personal profile and modify its access/information	REQ-1
Health care professional	Allow health care professionals to monitor high risk patients, or patients with different disorders to make	REQ-5

	necessary changes to optimize health of patient.	
Healthcare professional (In Emergency situation)	Access vitals of patient for emergency treatment	REQ-6
Fitness/ nutrition coach	Access client's reports then provide feedback	REQ-8
User	Connecting to database to review and submit changes	REQ-14

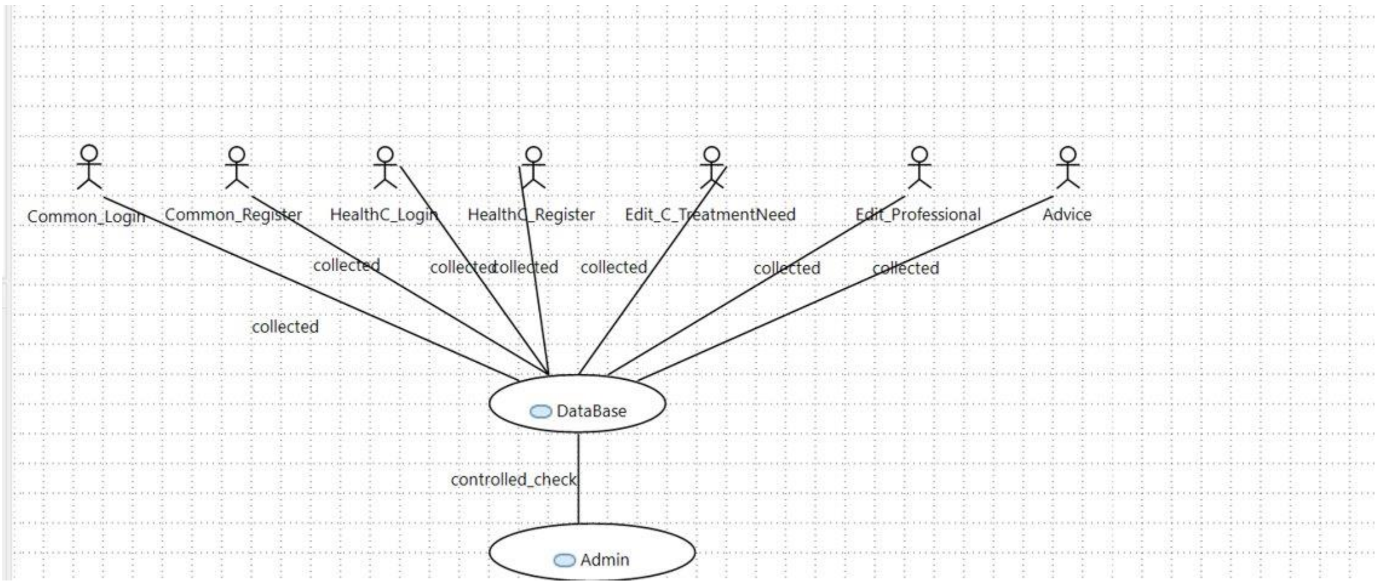


Figure 2: Use case diagram to represent users interacting with database.

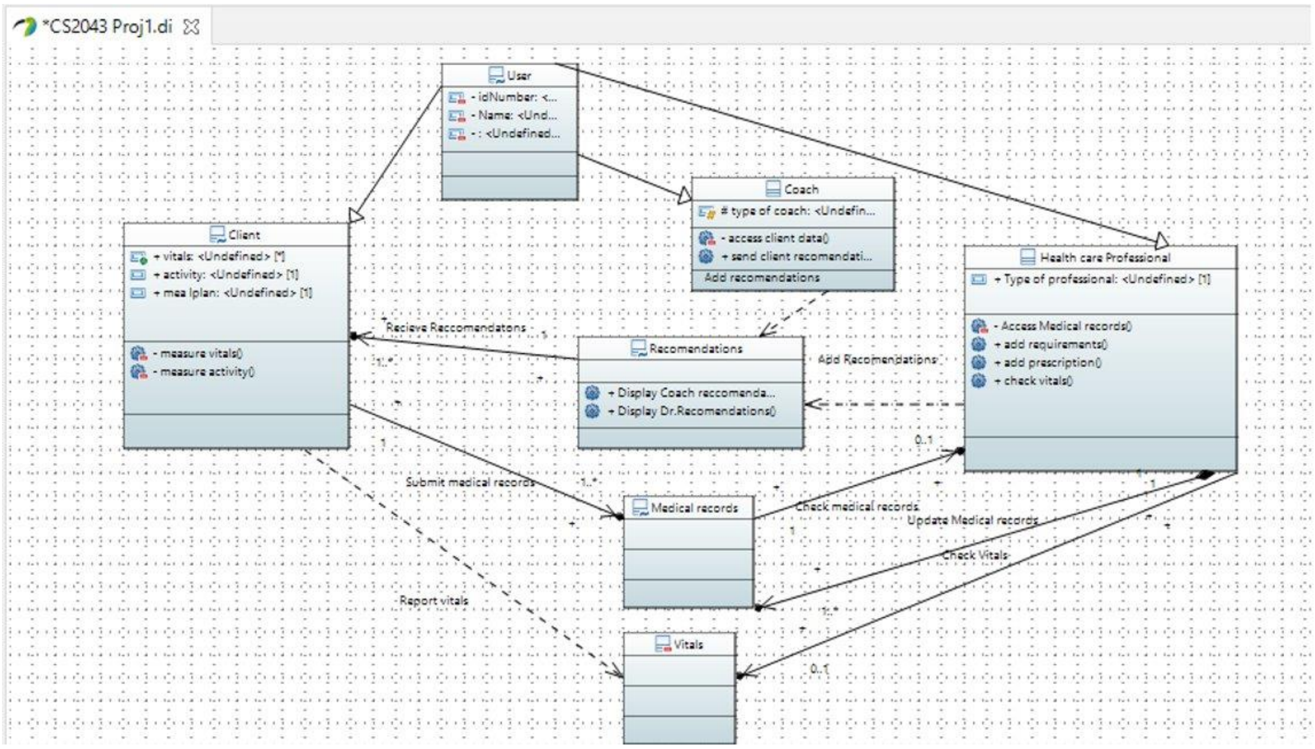


Figure 3: Class diagram to illustrate the desired operation of the system.

Due to a lack of experience in the industry, Java was the chosen language of development. Skeleton classes were coded to begin the development process. Sockets were chosen to implement messaging between users although, with further information, a different decision would have been made. A socket represents a connection for communication between two users (refer to figure 3). Multiple individuals in client/server or distributed systems can simultaneously communicate with a single Web server. Multiple sockets are required to do this. The socket's flow of events starts with a connection-oriented client-to-server model, the socket on the server process waits for requests from a client. To do this, the server first establishes (binds) an address that clients can use to find the server. When the address is established, the server waits for clients to request a service. The client-to-server data exchange takes place when a client connects to the server through a socket. The server performs the client's request and sends the reply back to the client.

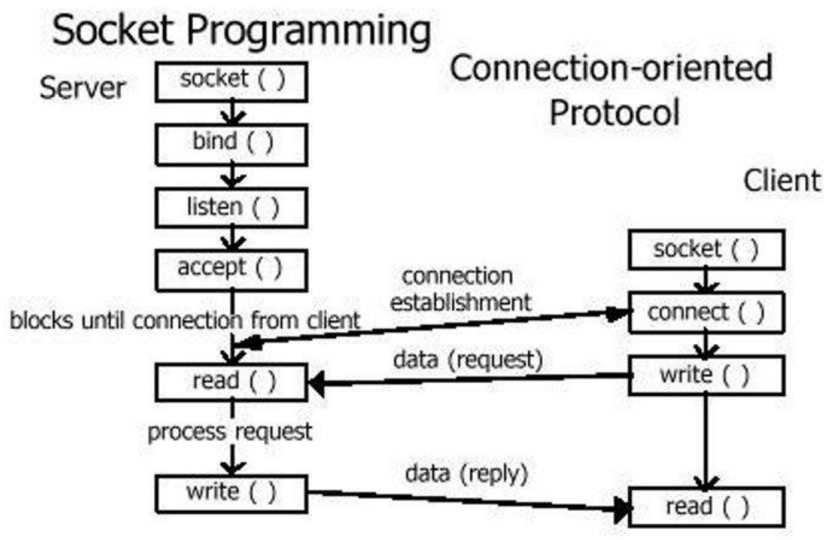


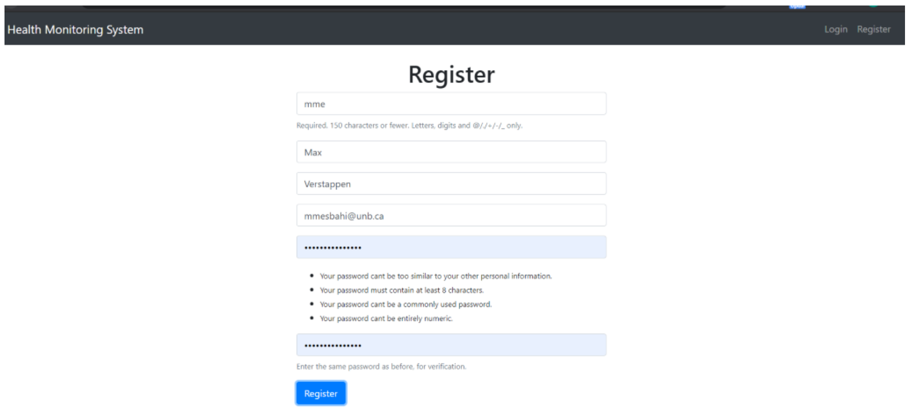
Figure 3: Communication diagram to represent sockets

Health Monitoring System Website

Our website was locally developed using the Django framework, which is web framework for python, along with some HTML, bootstrap, and sqlite3 for the database. Bootstrap was implemented using the hosted bootstrap CDN, and custom forms were included to the Django Project. The user can explore the website through the navigation bar at the top ribbon.

The website allows the user to perform the following set of functionalities:

- A potential user can create a unique profile using his/her first and last name, email, username, and user-defined password. The password should follow the requirements presented on the register page and the user can create his account only if the security requirements are followed



Register

mme

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Max

Verstappen

mmesbahi@unb.ca

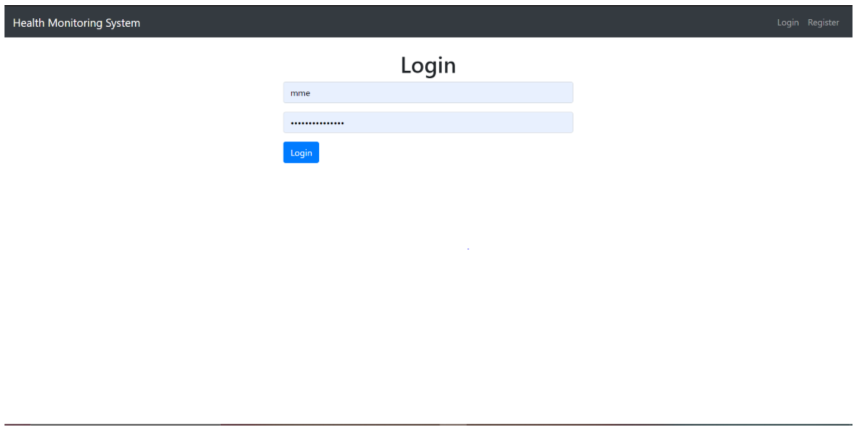
•••••••••••••••

- Your password cant be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password cant be a commonly used password.
- Your password cant be entirely numeric.

•••••••••••••••

Enter the same password as before, for verification.

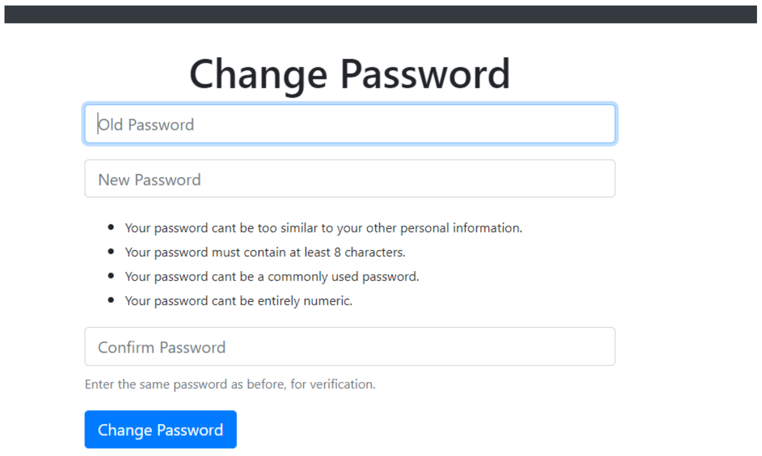
- A user can login to his/her created account without any limitations or constraints, but the same user cannot access another user's profile



mme

•••••••••••••••

- When the user is logged in, he/she can change the password associated with the account (following the same security requirements as in the registration phase) if the old password is provided.


Change Password

Old Password

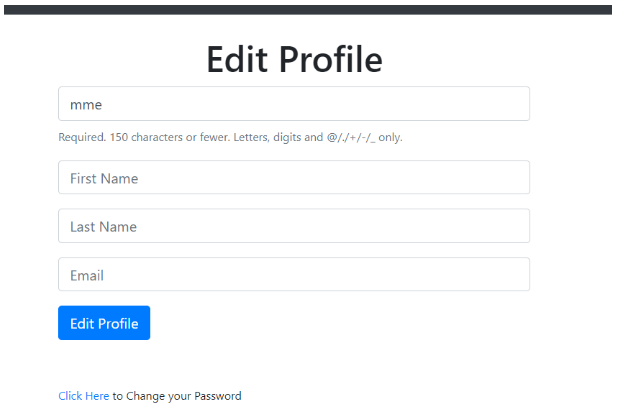
New Password

- Your password cant be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password cant be a commonly used password.
- Your password cant be entirely numeric.

Confirm Password

Enter the same password as before, for verification.

- In the edit profile section, a logged-in user can alter his/her username, first and last name, as well as the email address. At the bottom of the page, a user can click on the link which will redirect to the change password page.


Edit Profile

mme

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

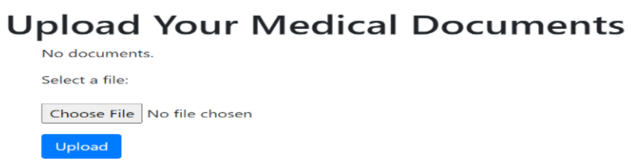
First Name

Last Name

Email

Click Here to Change your Password

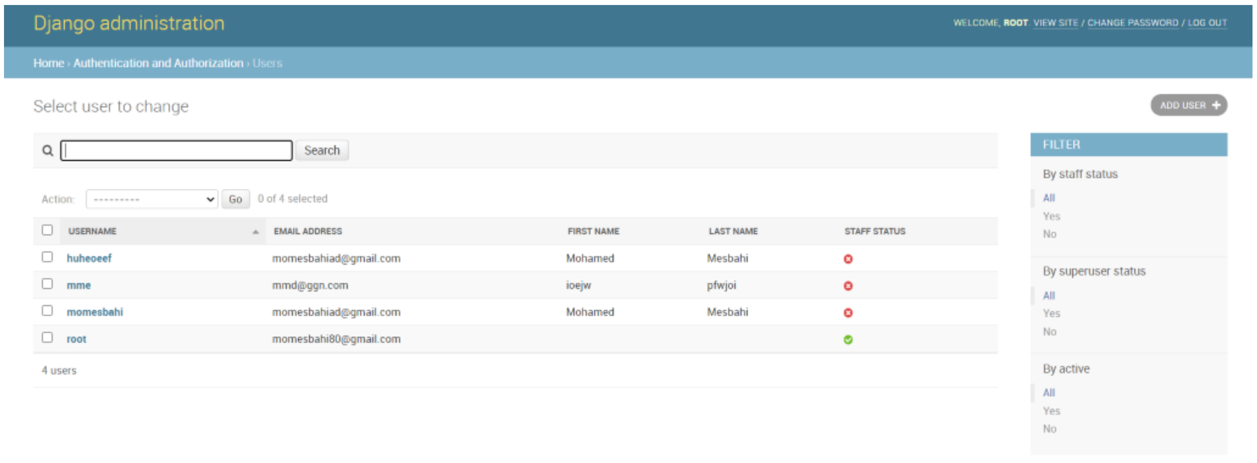
- An upload feature is added to the registered user's page. The user can upload his/her medical data using this feature (files are hosted locally).


Upload Your Medical Documents

No documents.

Select a file:

- The Django framework provides an admin space where a superuser can monitor the number of users and either add a new user or delete an existing user. The admin can also track the uploaded files and choose to keep or delete these files.



Django administration

WELCOME, ROOT. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Authentication and Authorization › Users

Select user to change

ADD USER

Search

Action: --------- Go 0 of 4 selected

USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
huheoeef	momesbahiad@gmail.com	Mohamed	Mesbahi	✗
mme	mmd@ggn.com	ioejw	pfwjoi	✗
momesbahi	momesbahiad@gmail.com	Mohamed	Mesbahi	✗
root	momesbahi80@gmail.com			✓

4 users

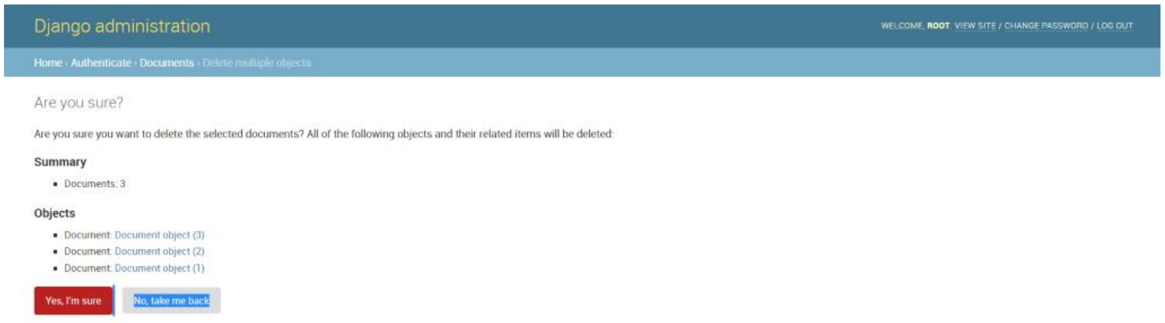
FILTER

By staff status: All, Yes, No

By superuser status: All, Yes, No

By active: All, Yes, No

Figure 4: Django Admin Users Dashboard



Django administration

WELCOME, ROOT. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Authenticate › Documents › Delete multiple objects

Are you sure?

Are you sure you want to delete the selected documents? All of the following objects and their related items will be deleted:

Summary

- Documents: 3

Objects

- Document: Document object (3)
- Document: Document object (2)
- Document: Document object (1)

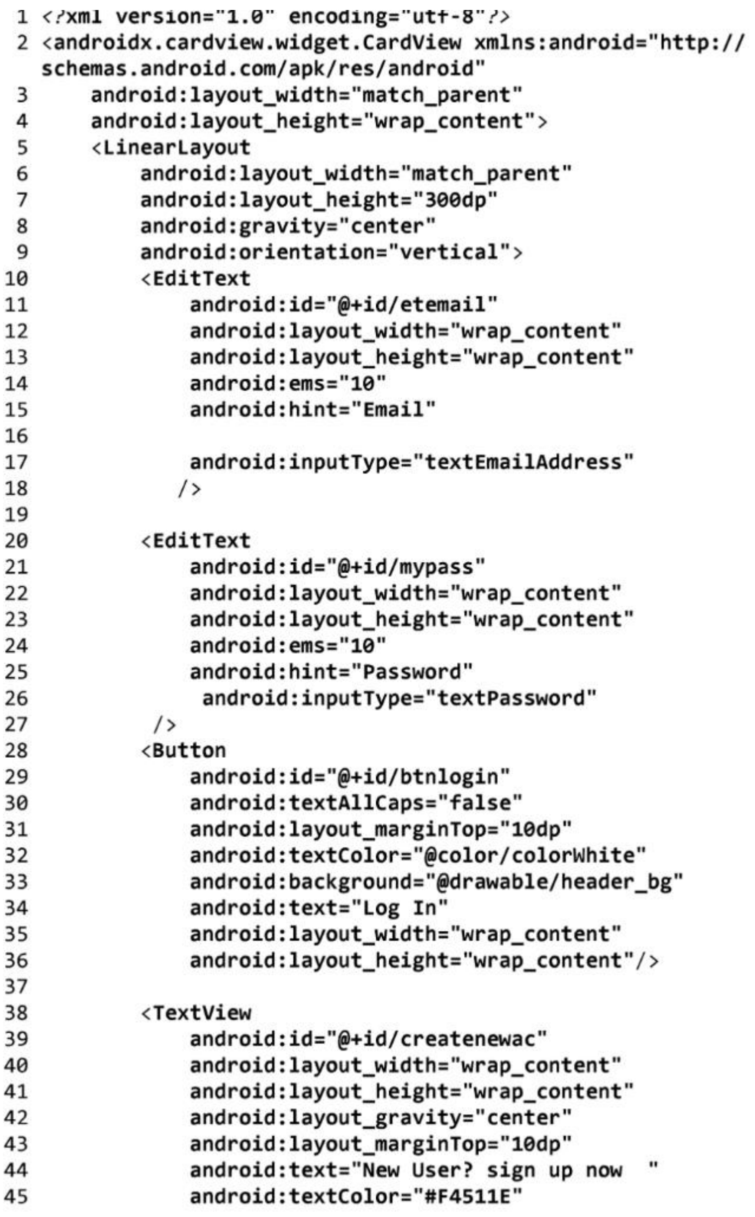
Yes, I'm sure | No, take me back

Figure 5: Deleting Documents using The Admin Functionality

Design

In the design part, we decided to use two forms of web and Android softwares to develop our program. The following is the Android software design process. We used Android Studio to develop. First, we designed the application login interface. We used some simple codes to create the input account, password and login and registration buttons for the login interface.

Here is the code for our login interface



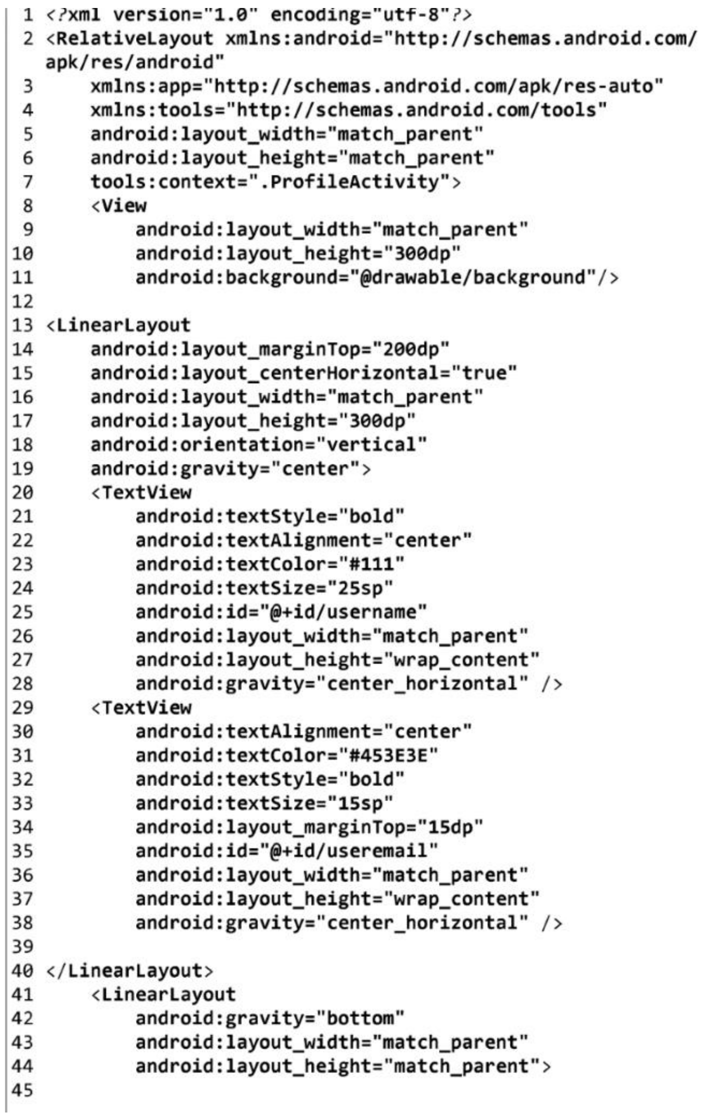
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:gravity="center"
        android:orientation="vertical">
        <EditText
            android:id="@+id/etemail"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:ems="10"
            android:hint="Email"

            android:inputType="textEmailAddress"
           />

        <EditText
            android:id="@+id/mypass"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:ems="10"
            android:hint="Password"
             android:inputType="textPassword"
         />
        <Button
            android:id="@+id/btnlogin"
            android:textAllCaps="false"
            android:layout_marginTop="10dp"
            android:textColor="@color/colorWhite"
            android:background="@drawable/header_bg"
            android:text="Log In"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>

        <TextView
            android:id="@+id/createnewac"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="10dp"
            android:text="New User? sign up now  "
            android:textColor="#F4511E"
```

```
                android:textSize="20sp"
                android:textStyle="bold" />
    </LinearLayout>

</androidx.cardview.widget.CardView>
```

After that, we built a create registration interface, enter your name, email and password, and sign in up button. Here is the code for registration interface.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SignUpActivity">
    <View
        android:id="@+id/header_view"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:background="@drawable/background"/>
    <LinearLayout
        android:layout_marginTop="10dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/header_view"
        android:orientation="vertical">
        <include layout="@layout/signup_form"/>

    </LinearLayout>

    <LinearLayout
        android:gravity="bottom"
        android:layout_width="match_parent"
        android:layout_height="match_parent">


        <View

            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:background="@drawable/bottom_bg"/>
    </LinearLayout>


</RelativeLayout>
```

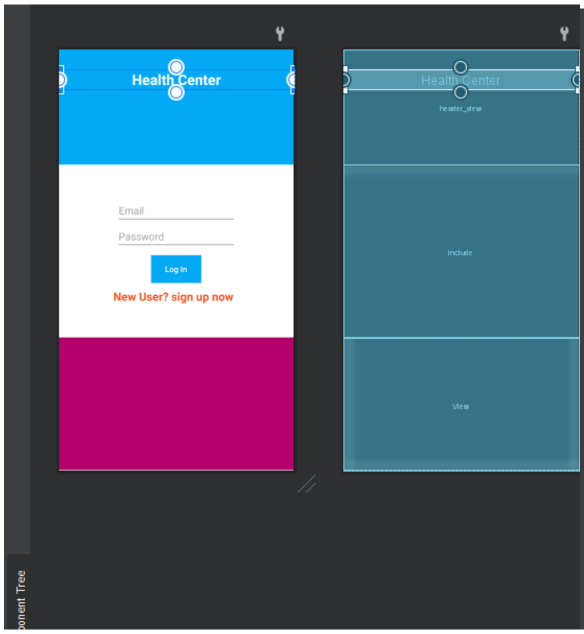
Then, create a profile interface to save user's input information. Here is the code for profile

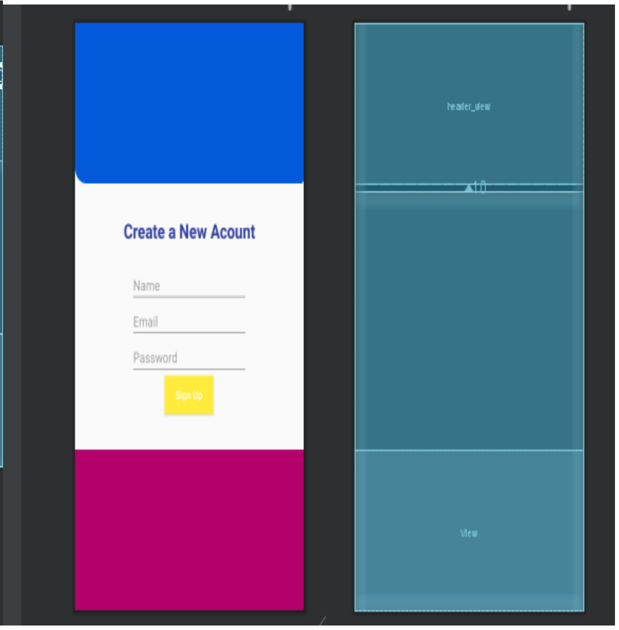


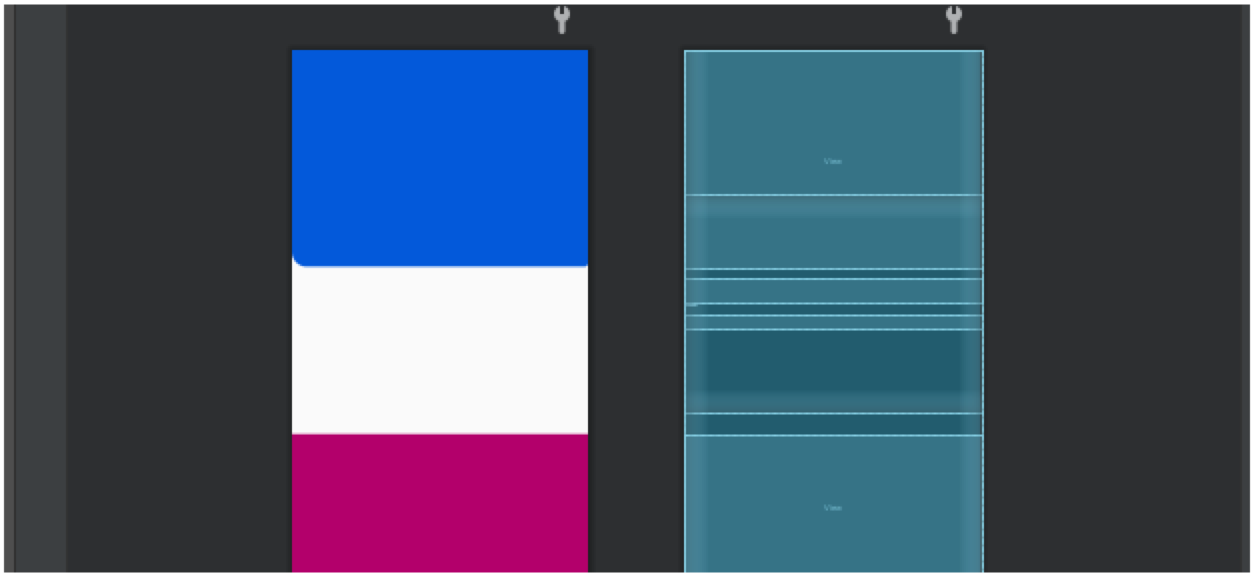
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ProfileActivity">
    <View
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:background="@drawable/background"/>

<LinearLayout
    android:layout_marginTop="200dp"
    android:layout_centerHorizontal="true"
    android:layout_width="match_parent"
    android:layout_height="300dp"
    android:orientation="vertical"
    android:gravity="center">
    <TextView
        android:textStyle="bold"
        android:textAlignment="center"
        android:textColor="#111"
        android:textSize="25sp"
        android:id="@+id/username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal" />
    <TextView
        android:textAlignment="center"
        android:textColor="#453E3E"
        android:textStyle="bold"
        android:textSize="15sp"
        android:layout_marginTop="15dp"
        android:id="@+id/useremail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal" />

</LinearLayout>
    <LinearLayout
        android:gravity="bottom"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

```

```xml
        <View

            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:background="@drawable/bottom_bg"/>
    </LinearLayout>


</RelativeLayout>
```

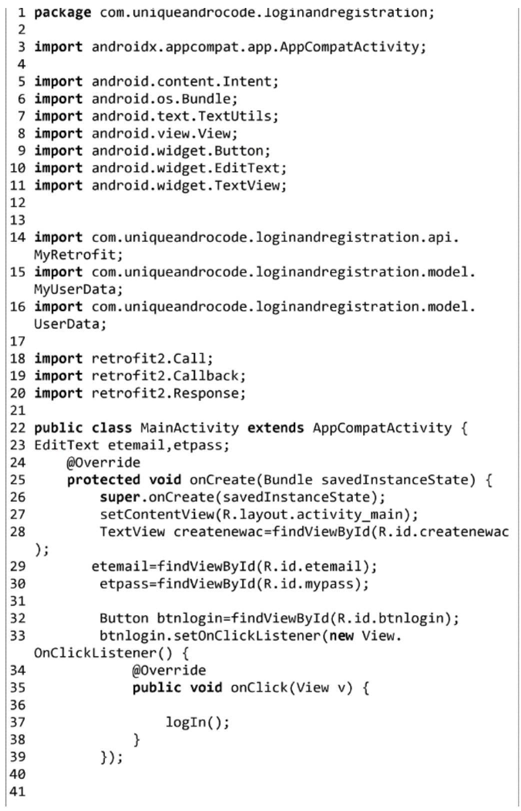
After that, we need to create some elements such as interface color and interface shape and create a main interface slide to include the login interface. Here are three interfaces display:



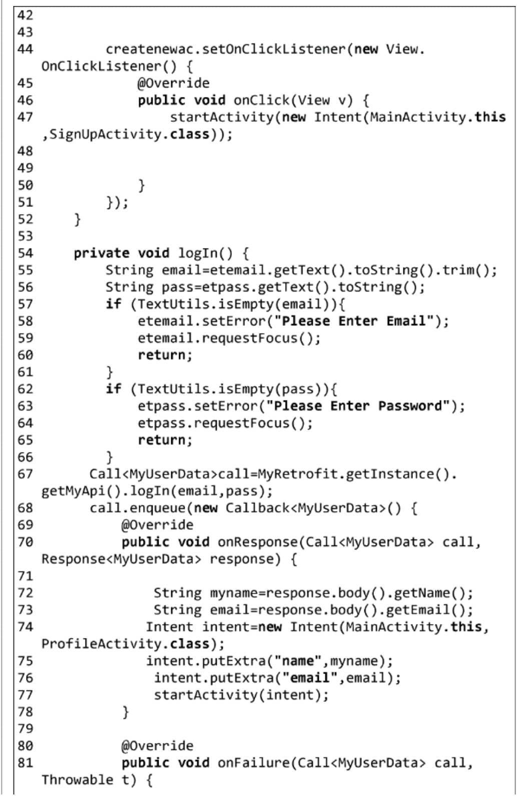




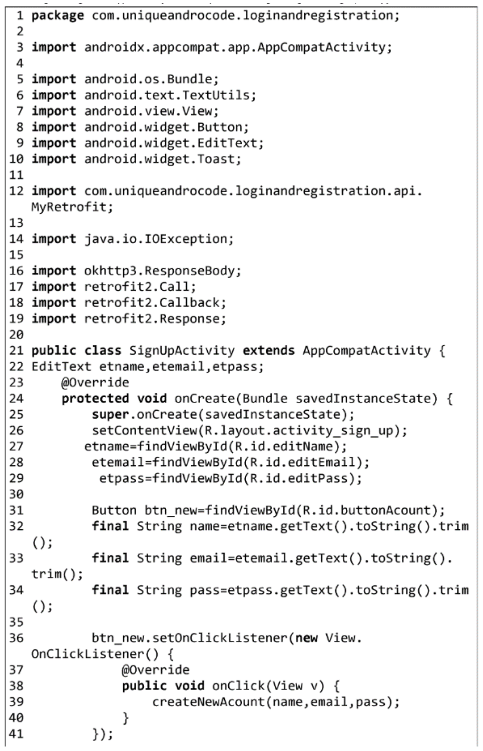
We created three java classes to implement those three interfaces and those three classes are controlled by the main class. Below is the code of the three classes



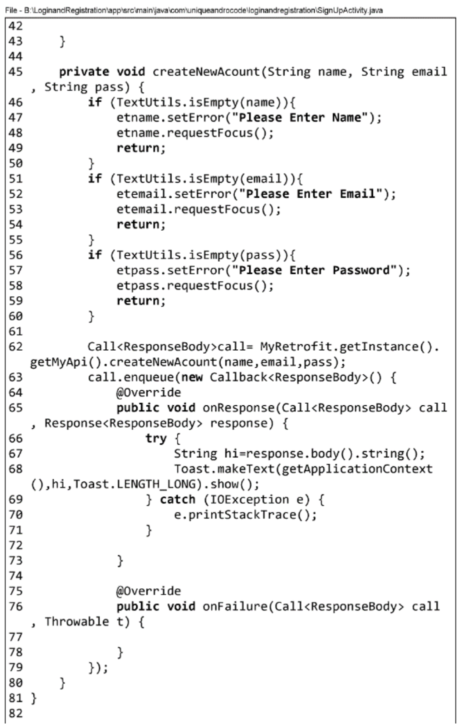
```java
package com.uniqueandrocode.loginandregistration;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;


import com.uniqueandrocode.loginandregistration.api.MyRetrofit;
import com.uniqueandrocode.loginandregistration.model.MyUserData;
import com.uniqueandrocode.loginandregistration.model.UserData;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class MainActivity extends AppCompatActivity {
EditText etemail,etpass;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView createnewac=findViewById(R.id.createnewac);
       etemail=findViewById(R.id.etemail);
        etpass=findViewById(R.id.mypass);

        Button btnlogin=findViewById(R.id.btnlogin);
        btnlogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                logIn();
            }
        });


```



```java


        createnewac.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(MainActivity.this,SignUpActivity.class));


            }
        });
    }

    private void logIn() {
        String email=etemail.getText().toString().trim();
        String pass=etpass.getText().toString();
        if (TextUtils.isEmpty(email)){
            etemail.setError("Please Enter Email");
            etemail.requestFocus();
            return;
        }
        if (TextUtils.isEmpty(pass)){
            etpass.setError("Please Enter Password");
            etpass.requestFocus();
            return;
        }
      Call<MyUserData>call=MyRetrofit.getInstance().getMyApi().logIn(email,pass);
      call.enqueue(new Callback<MyUserData>() {
          @Override
          public void onResponse(Call<MyUserData> call, Response<MyUserData> response) {

              String myname=response.body().getName();
              String email=response.body().getEmail();
             Intent intent=new Intent(MainActivity.this, ProfileActivity.class);
             intent.putExtra("name",myname);
              intent.putExtra("email",email);
              startActivity(intent);
          }

          @Override
          public void onFailure(Call<MyUserData> call, Throwable t) {
```

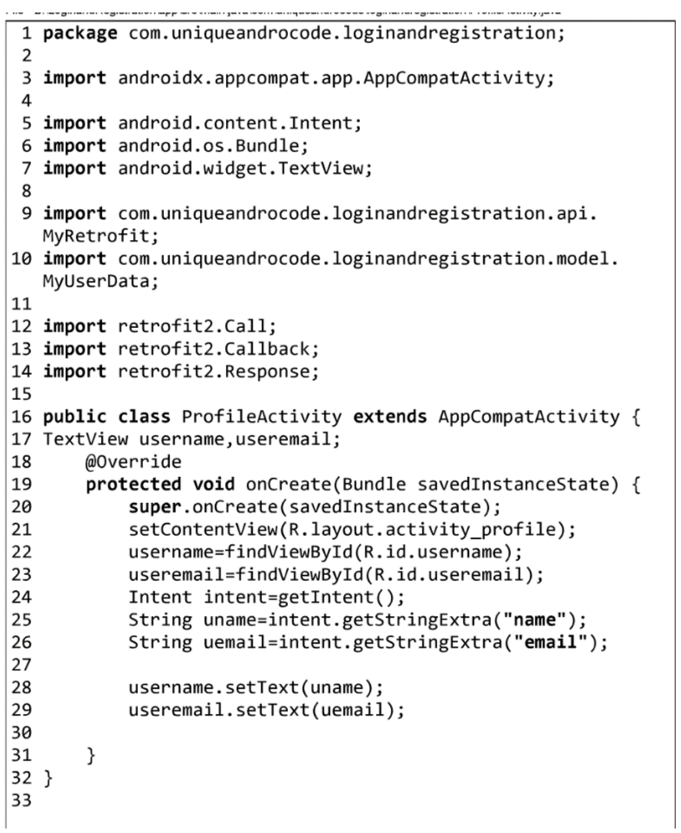


```java
package com.uniqueandrocode.loginandregistration;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.uniqueandrocode.loginandregistration.api.MyRetrofit;

import java.io.IOException;

import okhttp3.ResponseBody;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class SignUpActivity extends AppCompatActivity {
EditText etname,etemail,etpass;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_up);
       etname=findViewById(R.id.editName);
        etemail=findViewById(R.id.editEmail);
         etpass=findViewById(R.id.editPass);

        Button btn_new=findViewById(R.id.buttonAcount);
        final String name=etname.getText().toString().trim();
        final String email=etemail.getText().toString().trim();
        final String pass=etpass.getText().toString().trim();

        btn_new.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                createNewAcount(name,email,pass);
            }
        });
```



File - B:\LoginandRegistration\app\src\main\java\com\uniqueandrocode\loginandregistration\SignUpActivity.java

```java

    }

    private void createNewAcount(String name, String email, String pass) {
        if (TextUtils.isEmpty(name)){
            etname.setError("Please Enter Name");
            etname.requestFocus();
            return;
        }
        if (TextUtils.isEmpty(email)){
            etemail.setError("Please Enter Email");
            etemail.requestFocus();
            return;
        }
        if (TextUtils.isEmpty(pass)){
            etpass.setError("Please Enter Password");
            etpass.requestFocus();
            return;
        }

        Call<ResponseBody>call= MyRetrofit.getInstance().getMyApi().createNewAcount(name,email,pass);
        call.enqueue(new Callback<ResponseBody>() {
            @Override
            public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {
                try {
                    String hi=response.body().string();
                    Toast.makeText(getApplicationContext(),hi,Toast.LENGTH_LONG).show();
                } catch (IOException e) {
                    e.printStackTrace();
                }

            }

            @Override
            public void onFailure(Call<ResponseBody> call, Throwable t) {

            }
        });
    }
}

```



```java
package com.uniqueandrocode.loginandregistration;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;

import com.uniqueandrocode.loginandregistration.api.
MyRetrofit;
import com.uniqueandrocode.loginandregistration.model.
MyUserData;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class ProfileActivity extends AppCompatActivity {
TextView username,useremail;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profile);
        username=findViewById(R.id.username);
        useremail=findViewById(R.id.useremail);
        Intent intent=getIntent();
        String uname=intent.getStringExtra("name");
        String uemail=intent.getStringExtra("email");

        username.setText(uname);
        useremail.setText(uemail);

    }
}
```

Testing

The next step is to test these three interfaces (after 3 tests and modify some bugs, the compiler has no errors and can run normally). When running the simulator, we can open the application successfully and try to log in and register to test. The result of the operation is: the program cannot detect the input and cannot jump to the profile interface. This problem cannot be fixed for technical reasons. However, we believe that we can solve this problem by further studying Android applications.

With a system associated with human health, the risks are severe. Our system would be subject to rigorous testing before being released. Some testing could include:

- Test connectivity and functionality of each use case
- Check encapsulation by attempting to access medical information from coach perspective
- Ensure client is able to access multiple recommendations and requirements at the same time

- Input altered data from external monitors to verify the system displays changes in real time
- Monitor healthy client with a duplicate account to ensure there are no discrepancies between accounts

Risks involved in this type of system include:

- Bugs providing inaccurate results
- Security breach affecting client privacy
- Trainability of health care professionals required to operate the system
- Relying on other technology (wearable device, Dexcom) to connect properly to our software

Future evolution of our system would be a lengthy process and the following would need to be considered:

- Enlisting guidance from professionals to determine estimated development costs
- Expanding classes to include various types of health care professionals with different requirements
- Meeting with representatives who have developed physical monitoring devices
- Consulting with health care professionals and coaches to determine the most affective user interface for their applications
- Implementing a functioning system is subject to rigorous testing

This project allowed us to participate in all the development phases of a software system. We were better able to understand what is involved in requirements, analysis, design, construction, testing and documentation. Our goal was to take the specific information we learned in CS2043 and implement it in numerous ways. Initially we developed our software management plan. Deciding what we wanted to create and who would be responsible for what was the beginning of making and defending sound engineering decisions. Although, as we progressed some of those decisions were altered to better suit our finished product. Project management was the key to stay connected as a team and make decisions on what we wanted to do and how. We achieved most goals through planning, preparation, results and evaluation, contributing to achieving our strategic goals.