3.2.1 Mobile Application

As we made the mobile application, we felt that we did not need to use any software to plan how we wanted our android application to look. We knew that mockups are one of the most effective ways of communicating visual requirements clearly but as we were making decisions while working closely together, we felt as though we had a good idea of what we wanted on our mobile application to look like when we created it.

When we started to create the mobile application, we used the Android Studio program to make our Smart Watch mobile application. Android Studio is the official integrated development environment for Googles Android operating system. This software is specifically designed for Android development and is available for download on Windows, macOS and Linux based operating systems. It is written using Java and XML. We used Java for the Android functions and XML for the User Interface design.

We created the app in a way that the user should be able to successfully register an account and login using firebase. When the user is logged in successfully, they are directed to the application's homepage, where they are given a variety of options that lead to different functionality within the application. From here, they can choose to check their heart rate and heart rate history, their current body temperature, or the number of steps they have walked. The user also has access to a timer and stopwatch within the application to use for a variety of different fitness-related tasks. The user should have access to all the product functions when they are connected on a Wi-Fi network with access to the internet.

I made the homepage using XML. Any photographs used in the application were added to the application's *drawables* folder, and included in the application using XML's Image

View utility. All the buttons available on the home page are links to different activities in the application, with each button being handled by an intent within the application. We also added the functionality to close the application when the *back* button is pressed twice in quick succession.

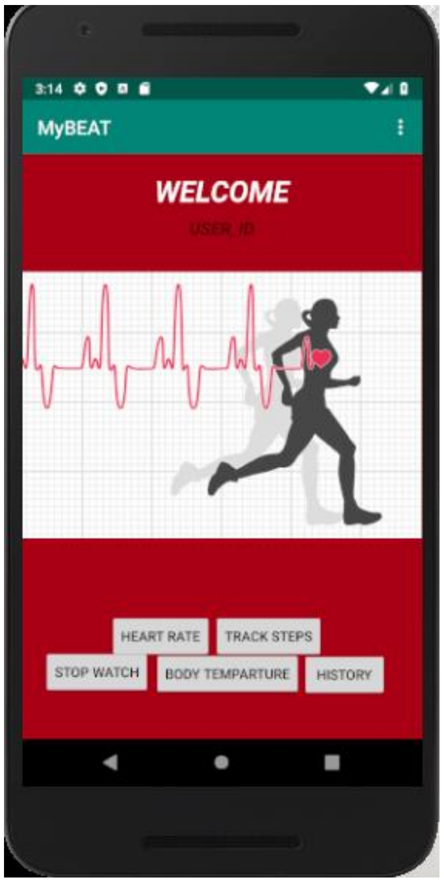
After the homepage, we have the pages for the three main sensors; the heart rate sensor, temperature sensor and the accelerometer. The pages are just made to show the current data output of each sensor individually and the history activity was made to view all three of the sensor's data simultaneously. The pages are made using the XML Text View, where the input number is saved as a string and shown on the screen. The data is pulled from the database in real-time, where firmware from the development platform writes to.

There is also the timer activity and stop-watch activity. These pages are made using the XML and a Thread/Handler Combination. There are also pages such as the About page which talks about the group project and the members also there is the settings page with user functions to change the language preferences.

For the login activity, we used Firebase SDK Authentication for logging users into the application. This is used to authenticate users with their email address and passwords. The Firebase Authentication SDK provides methods to create and manage users that use their emails and passwords to sign in. It also handles sending password reset emails. After the user signs in for the first time, a new user account is created and linked to the credentials the user signed in with. This new account is stored as part of the Firebase project, and can be used to identify a user across every app in our project. In the Firebase Realtime Database and Cloud Storage Security rules, we can get the signed in users unique id from the Authentication variable and use it to control what data the user can access.

For the Data visualization activity (**Error! Reference source not found.**), we have our file home.xml to show the 5 different activities that can be accessed and used. This is the main page for our android application, where the users can switch between the different sensors incorporated in the project. So far, we can click to go into the activity but there is no data as the database has not been set up. When the database would be set up the user can use the application to its full potential.

For the Action control activity, we have Java functions for the timer and stop-watch activities. The app will allow the user to access these functions for multiple reason may those reasons be for workout or just a simple timer to count your beats per minute. The Data visualization and Action control activity both work together we have designed our mobile application to work as a tool for users to view their vitals and to have access to the resources that they need to help with active living.



Link to code: https://github.com/Breezydust/SmartWatch/tree/master/Software/app

7.0 Appendix

7.2 Application code

bodyTemp.java – Activity to show current body temperature and history

```
public class bodyTemp extends AppCompatActivity implements View.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_bodytemp);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

        switch (item.getItemId()) {
            case R.id.about:
                Intent intent1 = new Intent(this, About.class);
                this.startActivity(intent1);
                return true;
            case R.id.Setting:
                Intent intent2 = new Intent(this, settings.class);
                this.startActivity(intent2);
                return true;
            case R.id.SignOut:
                Intent intent3 = new Intent(this, MainActivity.class);
                this.startActivity(intent3);
                return true;
            case R.id.Quit:
                finish();
                System.exit(0);
                return true;
            default:
                return super.onOptionsItemSelected(item);

        }



    }
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(bodyTemp.this, home.class);
        startActivity(intent);
    }
}
```

heartRateActivity.java – This activity shows the users current heart rate and history

```
public class heartRateActivity extends AppCompatActivity implements
View.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_heartrate);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.about:
                Intent intent1 = new Intent(this, About.class);
                this.startActivity(intent1);
                return true;
            case R.id.Setting:
                Intent intent2 = new Intent(this, settings.class);
                this.startActivity(intent2);
                return true;
            case R.id.SignOut:
                Intent intent3 = new Intent(this, MainActivity.class);
                this.startActivity(intent3);
                return true;
            case R.id.Quit:
                finish();
                System.exit(0);
                return true;
            default:
                return super.onOptionsItemSelected(item);

        }



    }
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(heartRateActivity.this, home.class);
        startActivity(intent);
    }
}
```

<u>trackSteps.java –</u> This activity shows users steps

```
public class trackSteps extends AppCompatActivity implements View.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tracksteps);
    }
```

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {

    switch (item.getItemId()) {
        case R.id.about:
            Intent intent1 = new Intent(this, About.class);
            this.startActivity(intent1);
            return true;
        case R.id.Setting:
            Intent intent2 = new Intent(this, settings.class);
            this.startActivity(intent2);
            return true;
        case R.id.SignOut:
            Intent intent3 = new Intent(this, MainActivity.class);
            this.startActivity(intent3);
            return true;
        case R.id.Quit:
            finish();
            System.exit(0);
            return true;
            default:
                return super.onOptionsItemSelected(item);

        }



    }

    @Override
    public void onClick(View v) {
        Intent intent = new Intent(trackSteps.this, home.class);
        startActivity(intent);
    }
    public void setV(View v) {
        Toast toast = Toast.makeText(getApplicationContext(),
                "Step Goal set",
                Toast.LENGTH_SHORT);

        toast.show();
    }
}
```