Fitlife

Android Application

Prepared by:

Mohammed Al-Shaibani

Anuhya Chowdary Paturi

Supreyo Atonu

Elias Tokola

Table of Content

1. Acknowledgment

The first acknowledgement we would like to make is to the professor of the course Dr. Shaon Bhatta Shuvo, for teaching the material needed to build this project.

Lastly, we would like to acknowledge the developer jjoe64 who created the GraphView library. Having functional and dynamic graphs in this project would not have been possible without this library. The GitHub link for the library is here along with the wiki on how to use it: https://github.com/jjoe64/GraphView.

2. Introduction

Fitness and health continues to be an important aspect of people's lifestyles, and with the growth of technology comes new and different paths to accomplish one's goals. As a result, it’s becoming more difficult to know where and how to get reliable information and resources. From diet hacks to workout routines, there are so many supposed solutions for everyone, but most of them come with a downside. Companies have been capitalizing on this trend by selling fitness app subscriptions, and these apps often offer the bare minimum with a free tier while also collecting user data. With this in mind, our fitness app aims to provide a comprehensive and reliable solution for people who want to reach their fitness goals without additional costs or privacy concerns.

3. Objectives

FitLife's idea for a comprehensive fitness app is to create a platform that houses all the fitness essentials in one place. The main objective is to make it easier for users to access different types of fitness content such as workout routines, nutrition plans, and tracking tools all in one app. To incentivize users, FitLife plans to offer FREE workout routines that are curated by our team, ensuring that the content is of high quality and effective. Additionally, we want to create a sense of community by allowing users to create and share their workout routines within the app, providing users with a wider variety of routines to choose from and making the app more engaging and interactive.

FitLife also plans to incorporate real-time charts that track weight gain or weight loss, as this would be a valuable feature for users looking to achieve specific fitness goals. This could also include progress tracking for different types of exercises or metrics such as

body fat percentage or muscle mass. Finally, we want to add a personalization feature that would allow users to customize their workout routines and nutrition plans based on their fitness goals, preferences, and fitness levels. This would make the app more appealing to a wider range of users and differentiate it from other fitness apps on the market.

By combining expert-curated content with community-driven content, real-time tracking, and personalization, FitLife believes that our fitness app would offer users a comprehensive and engaging fitness experience that would set it apart from the many other fitness apps on the Play Store.

4. Project Requirements

User Interface: The app should have an easy-to-use, intuitive interface with attractive graphics and typography.

Content Management System: Should develop a content management system that lets users create and share their workout routines, while also allowing the management of fitness content such as workout routines and nutrition plans.

Real-time Tracking: The app should include real-time tracking features that allow users to track their weight gain or loss, as well as progress tracking for different types of exercises or metrics such as body fat percentage or muscle mass.

Personalization: The app should offer a feature that lets users customize their workout routines and nutrition plans based on their fitness goals, preferences, and fitness levels.

5. Tools and Components

For this project, we used multiple tools and components to build the app with all its different functions. The base of the app uses a main activity to manage different pages that are navigable by a navigation bar created by a BottomNavigationView. The different pages are created in fragments and contained within a frame layout within the main activity layout. These fragment pages are managed by the navigation bar and are interchangeable through the navigation buttons. Within these fragment pages, there are accessible option pages that temporarily open to new activities.

The storage of user and routine data is handled with an sqlite database, that stores all the data within the systems storage. When a user is logged into the app, their user information is stored within the device through a SharedPreferences function. Once it's stored, it allows the user's information to be accessible from all the pages. The different pages are able to pull that info from the device and make queries to modify the users related data. Finally, the external library GraphView was used to display graphs in the home page.

6. Project Development

6.1: Bottom Navigation Bar

The first major step we took was to first build and design the front end of the project. As mentioned in the Tools and Components section, the use of BottomNavigationView with a MainActivity to handle all the Fragments was essential for our frontend to be clean and easy to navigate.

```
public class MainActivity extends AppCompatActivity {
    13 usages
    FragmentManager fragmentManager = getSupportFragmentManager();
    2 usages
    BottomNavigationView bottomNavigationView;
    6 usages
    HomeFragment homeFragment = new HomeFragment();
    6 usages
    DiscoverFragment discoverFragment = new DiscoverFragment();
    6 usages
    SavedWorkoutsFragment savedWorkoutsFragment = new SavedWorkoutsFragment();
    4 usages
    ProfileFragment profileFragment = new ProfileFragment();
    9 usages
    Fragment currentActive;
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    sqLiteManager = new SQLiteManager( context: MainActivity.this);

    bottomNavigationView = findViewById(R.id.navigation_bar);
    fragmentManager.beginTransaction().add(R.id.page_container, homeFragment,  tag: "home_page").hide(homeFragment).commit();
    fragmentManager.beginTransaction().add(R.id.page_container, discoverFragment,  tag: "discover_page").hide(discoverFragment).commit();
    fragmentManager.beginTransaction().add(R.id.page_container, savedWorkoutsFragment,  tag: "saved_page").hide(savedWorkoutsFragment).commit();
    fragmentManager.beginTransaction().add(R.id.page_container, profileFragment,  tag: "profile_page").hide(profileFragment).commit();

    fragmentManager.beginTransaction().show(homeFragment).commit();
    currentActive = homeFragment;

    bottomNavigationView.setOnItemSelectedListener(new NavigationBarView.OnItemSelectedListener() {
```

```
bottomNavigationView.setOnItemSelectedListener(new NavigationBarView.OnItemSelectedListener() {
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        if(item.getItemId() == R.id.home){
            fragmentManager.beginTransaction().hide(currentActive).show(homeFragment).commit();
            currentActive = homeFragment;
            return true;
        } else if(item.getItemId() == R.id.discover){
            fragmentManager.beginTransaction().detach(discoverFragment).commitNow();
            fragmentManager.beginTransaction().attach(discoverFragment).commitNow();
            fragmentManager.beginTransaction().hide(currentActive).show(discoverFragment).commit();
            currentActive = discoverFragment;
            return true;
        } else if(item.getItemId() == R.id.saved_workouts){
            fragmentManager.beginTransaction().detach(savedWorkoutsFragment).commitNow();
            fragmentManager.beginTransaction().attach(savedWorkoutsFragment).commitNow();
            fragmentManager.beginTransaction().hide(currentActive).show(savedWorkoutsFragment).commit();
            currentActive = savedWorkoutsFragment;
            return true;
        } else if(item.getItemId() == R.id.profile){
            fragmentManager.beginTransaction().hide(currentActive).show(profileFragment).commit();
            currentActive = profileFragment;
            return true;
        }

        return false;
    }
});
```

The 3 screenshots above showcase the structure of the bottom navigation in the MainActivity.java.

6.2: SQLite Database

The next step was to implement the SQLite database after completing the front end. We created a total of 5 tables to connect the frontend to the backend.

DiscoverFragment.java | SQLiteManager.java

```
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {
    //TODO Add any required details for user's weight and height <- possibly for El
    String createUsersTable = "CREATE TABLE USERS " +
            "(USER_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
            "USER_FNAME TEXT NOT NULL, " +
            "USER_LNAME TEXT NOT NULL, " +
            "USER_EMAIL TEXT UNIQUE NOT NULL, " +
            "USER_PASSWORD TEXT NOT NULL, " +
            "USER_WEIGHT DECIMAL(3,2), " +
            "USER_WEIGHT_GOAL DECIMAL(3,2)," +
            "USER_BODY_FAT DECIMAL(2,2)," +
            "USER_BODY_FAT_GOAL DECIMAL(2,2)," +
            "USER_HEIGHT DECIMAL(3,2), " +
            "USER_AGE INTEGER," +
            "USER_REGISTER_DATE TEXT)";

    String createRoutinesTable = "CREATE TABLE ROUTINES " +
            "(ROUTINE_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
            "ROUTINE_NAME TEXT, " +
            "ROUTINE_CREATOR TEXT, " +
            "ROUTINE_LEVEL TEXT, " +
            "ROUTINE_FREQUENCY INTEGER, " +
            "ROUTINE_LENGTH INTEGER)";
```

```
String createWorkoutsTable = "CREATE TABLE WORKOUTS " +
        "(WORKOUT_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
        "WORKOUT_NAME TEXT, " +
        "WORKOUT_DAY TEXT, " +
        "WORKOUT_SETS INTEGER, " +
        "WORKOUT_REPS INTEGER, " +
        "ROUTINE_ID INTEGER, "+
        "FOREIGN KEY(ROUTINE_ID) REFERENCES ROUTINES(ROUTINE_ID))";

String createUserRoutinesTable = "CREATE TABLE USER_ROUTINES " +
        "(USER_ROUTINES_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
        "USER_ID INTEGER NOT NULL, " +
        "ROUTINE_ID INTEGER NOT NULL, " +
        "STATUS INTEGER DEFAULT 0, " + //this is to help determine what routine the user is currently following (0 false, 1 true)
        "FOREIGN KEY(USER_ID) REFERENCES USERS(USER_ID), " +
        "FOREIGN KEY(ROUTINE_ID) REFERENCES ROUTINES(ROUTINE_ID))";

String createUserGoalRecords = "CREATE TABLE USER_RECORDS " +
        "(USER_CURRENT_WEIGHT DECIMAL(3,2), " +
        "USER_CURRENT_BODY_FAT DECIMAL(2,2), " +
        "USER_CURRENT_DATE TEXT," +
        "USER_ID INTEGER, " +
        "FOREIGN KEY(USER_ID) REFERENCES USERS(USER_ID))";

sqLiteDatabase.execSQL(createUsersTable);
sqLiteDatabase.execSQL(createRoutinesTable);
sqLiteDatabase.execSQL(createWorkoutsTable);
sqLiteDatabase.execSQL(createUserRoutinesTable);
sqLiteDatabase.execSQL(createUserGoalRecords);

populateRoutines(sqLiteDatabase);
```

The 2 screenshots above showcase the design and implementation of all the tables.

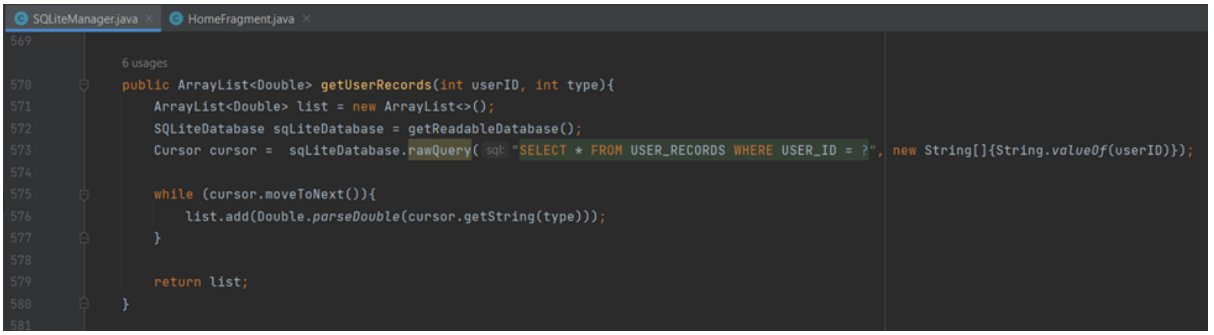
6.3: GraphView

The next step was to implement the GraphView in the home page. We first had to provide the implementation in the dependencies block in the build.gradle.

```
dependencies {

    implementation "androidx.activity:activity:1.6.0-alpha05"
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.8.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
    implementation 'com.jjoe64:graphview:4.2.2'
    implementation 'com.squareup.picasso:picasso:2.71828'
}
```

Next, we create the function in SQLiteManager that gets all user records that hold their weight and bodyfat records. The function takes in parameters userID and type. UsedID is used to search the records of only the specified user in the SQL query.

The type parameter is used for the Cursor to point to either the weight or body fat records. This function will return an ArrayList<Double>.



Next, we must define variables getWeightDP and getBodyFatDP of type DataPoint that will hold all the x and y coordinates. Using a for loop, we call the db.getUserRecords() function that will retrieve all the data and each iteration will create a new DataPoint.

Next, we initialize the graph of type GraphView and the 2 LineGraphSeries for weight and bodyfat. The LineGraphSeries will be initialized to the DataPoints we got earlier and the graph will be set to the weightSeries first.

HomeFragment.java

```
switchGraph.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (clicked){
            graph.removeAllSeries();
            graph.addSeries(weightSeries);
            graph.getViewport().setMinY(weightSeries.getLowestValueY() - 5);
            graph.getViewport().setMaxY(weightSeries.getHighestValueY() + 5);
            graph.setTitle("Weight Trend");
            switchGraph.setText("Weight");
            clicked = false;
        }

        else{
            graph.removeAllSeries();
            graph.addSeries(bodyFatSeries);
            graph.getViewport().setMinY(bodyFatSeries.getLowestValueY() - 2);
            graph.getViewport().setMaxY(bodyFatSeries.getHighestValueY() + 2);
            graph.setTitle("Body Fat Trend");
            switchGraph.setText("Body Fat");
            clicked = true;
        }
    }
});
```

Lastly, the function to switch between the weight and bodyfat graph is what the screenshot above showcases. We use a button with an onClick listener to switch between graphs.

6.4: Discovering Routines

The implementation of the discovery page was the next step in our development process. The discover page displays all the routines saved within the database in a list view, and gets the routine data from a call to the getRoutines function in the database manager.

```java
import ...

public class DiscoverFragment extends Fragment {
    View view;
    TextView routinesQuantity;
    EditText searchInput;
    Button filterButton, searchButton, clearButton;
    RecyclerView routinesView;
    BottomSheetDialog dialogFragment;
    String[] selectedLevels = new String[3];
    ArrayList<RoutineData> routines = new ArrayList<>();

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        view = inflater.inflate(R.layout.fragment_discover, container,  attachToRoot: false);
        routinesView = view.findViewById(R.id.routine_list);
        LinearLayoutManager layoutManager = new LinearLayoutManager(getActivity());
        routinesView.setLayoutManager(layoutManager);
        searchInput = view.findViewById(R.id.search_input);
        filterButton = view.findViewById(R.id.filter_button);
        searchButton = view.findViewById(R.id.search_button);
        clearButton = view.findViewById(R.id.clear_button);
        routinesQuantity = view.findViewById(R.id.result_number);
        dialogFragment = new BottomSheetDialog(requireActivity());

        SQLiteManager db = new SQLiteManager(getActivity());

        routines = db.getRoutines();

        String resultsNum = routines.size()+" results";
        routinesQuantity.setText(resultsNum);
        RoutineAdapter adapter = new RoutineAdapter(routines,  context: DiscoverFragment.this);
        routinesView.setAdapter(adapter);

```

The search function loops through the arraylist of Routines and checks for matches with the user's input. Once all matches are found the RoutinesAdpater updates the listview with the resulting routines.

```java
        searchInput.setOnFocusChangeListener(new View.OnFocusChangeListener() {
            @Override
            public void onFocusChange(View v, boolean hasFocus) {
                if(hasFocus){
                    filterButton.setVisibility(View.GONE);
                    clearButton.setVisibility(View.GONE);
                    searchButton.setVisibility(View.VISIBLE);
                } else if(TextUtils.isEmpty(searchInput.getText())){
                    filterButton.setVisibility(View.VISIBLE);
                    searchButton.setVisibility(View.GONE);
                }
            }
        });

        searchButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(!TextUtils.isEmpty(searchInput.getText())){
                    searchButton.setVisibility(View.GONE);
                    filterButton.setVisibility(View.GONE);
                    clearButton.setVisibility(View.VISIBLE);

                    ArrayList<RoutineData> filteredRoutines = new ArrayList<>();

                    for(RoutineData routine : routines){
                        if(routine.getTitle().toUpperCase().contains(String.valueOf(searchInput.getText()).toUpperCase()) ||
                                routine.getCreator().toUpperCase().contains(String.valueOf(searchInput.getText()).toUpperCase()) ||
                                routine.getLevel().toUpperCase().contains(String.valueOf(searchInput.getText()).toUpperCase()) ||
                                routine.getLength().toString().toUpperCase().contains(String.valueOf(searchInput.getText()).toUpperCase())){
                            filteredRoutines.add(routine);
                        }
                    }

                    String resultsNum = filteredRoutines.size()+" results";
                    routinesQuantity.setText(resultsNum);
                    RoutineAdapter adapter = new RoutineAdapter(filteredRoutines,  context: DiscoverFragment.this);
                    routinesView.setAdapter(adapter);
                } else {
                    clearButton.setVisibility(View.GONE);
                    filterButton.setVisibility(View.VISIBLE);
                    searchButton.setVisibility(View.GONE);
                }
            }
        });
```

The next major step within this page was the function to save routines to specific users. This was done using the 'USER_ROUTINES' table that stored the routine_id and user_id. Now when the user selects a routine the routine details activity is shown displaying the routines details. Within the page is a bookmark button that checks the status of the routine and saves it to the table by calling the saveUserRoutine function in the database manager.

```
103
104         saveButton.setOnClickListener(new View.OnClickListener() {
105             @Override
106             public void onClick(View v) {
107                 if (!saved) {
108                     sqLiteManager.saveUserRoutine(userID, id,  current: false);
109                     saveButton.setBackgroundResource(R.drawable.ic_bookmarked);
110                     saved = true;
111                 } else {
112                     sqLiteManager.unSaveUserRoutine(userID ,id);
113                     saveButton.setBackgroundResource(R.drawable.ic_bookmark);
114                     saved = false;
115                 }
116             }
117         });
118     }
119 }
120
```

6.5: Saved Routines

The next step in the development process is the saved routine page that displays all the users routines. The routines are pulled from the 'USER_ROUTINES' using the user's id and passing that to a function that returns the proper arraylist. This is then stored within a list view, exactly like the discover page.

```
93          set_Workout = view.findViewById(R.id.set_Workout);
94          set_day = view.findViewById(R.id.set_day);
95          set_reps = view.findViewById(R.id.set_reps);
96          set_Sets = view.findViewById(R.id.set_Sets);
97
98          SQLiteManager db = new SQLiteManager(getActivity());
99
100         SharedPreferences sharedPreferences = this.getActivity().getSharedPreferences( name: "user_info", Context.MODE_PRIVATE);
101         int userID = Integer.parseInt(sharedPreferences.getString( key: "user_id",  defValue: null));
102         String user_name = sharedPreferences.getString( key: "fname_key",  defValue: null) + " " + sharedPreferences.getString( key: "lname_key",  defValue: null);
103
104         routines = db.getUserRoutines(userID);
105         RoutineAdapter adapter = new RoutineAdapter(routines,  context: SavedWorkoutsFragment.this);
106         routine_list.setAdapter(adapter);
107
```

There are also the functions to unsave routines or create new routines with the buttons within the page.

```
        //OnClick handlers for navigating between views of fragment
        new_routine_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                saved_routine.setVisibility(View.GONE);
                new_routine.setVisibility(View.VISIBLE);
                saved_routine.invalidate();
                new_routine.invalidate();
            }
        });

        bt_add_routine.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                routineData = new RoutineData( id: -1,
                        et_routine_name.getText().toString(),
                        user_name,
                        et_routine_level.getText().toString(),
                        Integer.parseInt(et_routine_freq.getText().toString()),
                        Integer.parseInt(et_routine_length.getText().toString()),  workoutsList: null);

                db.addRoutines(routineData);
                createdRoutineID = db.getRoutineID(et_routine_name.getText().toString());
                db.addUserRoutines(createdRoutineID, userID);
                routines = db.getUserRoutines(userID);
                RoutineAdapter adapter = new RoutineAdapter(routines,  context: SavedWorkoutsFragment.this);
                routine_list.setAdapter(adapter);
                new_routine.setVisibility(View.GONE);
                new_workout.setVisibility(View.VISIBLE);
                new_routine.invalidate();
                new_workout.invalidate();
            }
        });

        bt_add_to_routine.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                created_routine.setVisibility(View.GONE);
                new_workout.setVisibility(View.VISIBLE);
                created_routine.invalidate();
                new_workout.invalidate();
            }
        });
```

```
        create_routine_back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                saved_routine.setVisibility(View.VISIBLE);
                new_routine.setVisibility(View.GONE);
                created_routine.invalidate();
                new_workout.invalidate();
            }
        });

        create_workout_back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                routines = db.getUserRoutines(userID);
                RoutineAdapter adapter = new RoutineAdapter(routines,  context: SavedWorkoutsFragment.this);
                routine_list.setAdapter(adapter);

                saved_routine.setVisibility(View.VISIBLE);
                new_workout.setVisibility(View.GONE);
                saved_routine.invalidate();
                new_workout.invalidate();
            }
        });

        your_routines_back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                created_routine.setVisibility(View.GONE);
                saved_routine.setVisibility(View.VISIBLE);
                created_routine.invalidate();
                saved_routine.invalidate();
            }
        });
        return view;
    }
```

6.6: User Profile

The next major step was to develop the profile page that allows the user to view or edit their personal information. This page first displays the user information by pulling it from its saved location within the system storage using SharedPreferences. There are buttons within the page to open the app setting activities or the edit profile activity.

```
        sharedPreferences = this.getActivity().getSharedPreferences( name: "user_info", Context.MODE_PRIVATE);
        fragmentManager = getActivity().getSupportFragmentManager();

        String fullName = sharedPreferences.getString( key: "fname_key",  defValue: null)+" "+sharedPreferences.getString( key: "lname_key",  defValue: null);
        String weight = "Weight: "+sharedPreferences.getString( key: "weight_key",  defValue: null)+"lbs";
        String height = "Height: "+sharedPreferences.getString( key: "height_key",  defValue: null)+"cm";
        String bodyF = "Body Fat: "+sharedPreferences.getString( key: "bod yFat_key",  defValue: null)+"%";
        String age = "Age: "+sharedPreferences.getString( key: "age_key",  defValue: null)+" years old";

        userName.setText(fullName);
        userEmail.setText(sharedPreferences.getString( key: "email_key",  defValue: null));
        userWeight.setText(weight);
        userBodyFat.setText(bodyF);
        userHeight.setText(height);
        userAge.setText(age);

        editProfileTextView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent details = new Intent(view.getContext(), EditProfileActivity.class);

                details.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
                view.getContext().startActivity(details);
            }
        });

        settingsButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent detailsPage = new Intent(view.getContext(), SettingsActivity.class);

                detailsPage.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
                view.getContext().startActivity(detailsPage);
            }
        });

        return view;
    }
```

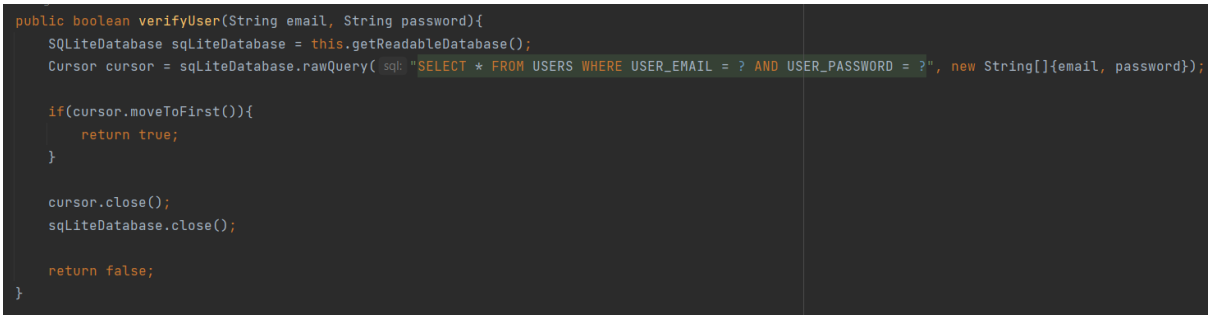
The functionality of the settings page is to change the measurement unit of weight and height. It also allows the user to logout of their account and change to the login activity.

```
        // Set the click listener for the logout button
        Button logoutButton = view.findViewById(R.id.logout_button);
        logoutButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                SharedPreferences.Editor editor = sharedPreferences.edit();

                editor.clear();
                editor.apply();

                Intent intent = new Intent(getActivity(), LoginActivity.class);
                startActivity(intent);
            }
        });

        return view;
    }
```

The edit page was simply designed to allow the user to change their information and update the database with any changes.

6.7: Login/Signup

The last major step was to implement the register and login since we now have the backend to process simple authentication of checking whether a password matches the username.



```
public boolean verifyUser(String email, String password){
    SQLiteDatabase sqLiteDatabase = this.getReadableDatabase();
    Cursor cursor = sqLiteDatabase.rawQuery( sql: "SELECT * FROM USERS WHERE USER_EMAIL = ? AND USER_PASSWORD = ?", new String[]{email, password});

    if(cursor.moveToFirst()){
        return true;
    }

    cursor.close();
    sqLiteDatabase.close();

    return false;
}
```

The screenshot above showcases the SQL query used to verify the user.

```
public boolean addUser(UserData user) {
    SQLiteDatabase sqLiteDatabase = this.getWritableDatabase();
    ContentValues values = new ContentValues();

    values.put("USER_FNAME", user.getFname());
    values.put("USER_LNAME", user.getLname());
    values.put("USER_EMAIL", user.getEmail());
    values.put("USER_PASSWORD", user.getPassword());
    values.put("USER_WEIGHT", user.getWeight());
    values.put("USER_WEIGHT_GOAL", user.getWeightG());
    values.put("USER_BODY_FAT", user.getBodyFat());
    values.put("USER_BODY_FAT_GOAL", user.getBodyFatG());
    values.put("USER_HEIGHT", user.getHeight());
    values.put("USER_AGE", user.getAge());
    values.put("USER_REGISTER_DATE", user.getRegisteredDate());

    long status = sqLiteDatabase.insert( table: "USERS",  nullColumnHack: null, values);

    sqLiteDatabase.close();
    return status != -1;
}
```

Lastly, this is the SQL queries to add a user once they finish registering their account.

7. User Manual

7.1 Creating an Account

On the first launch of FitLife, you are greeted with this screen (fig. 1).You will have the option to login, or press the register button below. Since this is the first app launch, press Register to create your account.

Once you press the register button, you are greeted with the following screen (fig. 2). Enter your details as asked, including a valid email address and password (min 4 characters).

Once that is completed, you are greeted with the last prompt for Registering an Account (fig. 3). On the left you will input your current weight and body fat percentage, as prompted, and on the right you will enter your goals that you will accomplish with the use of FitLife. Next you input your height and age. Once completed, press the Register button.

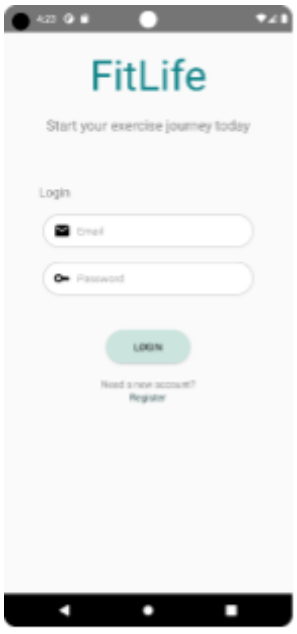


Figure 1

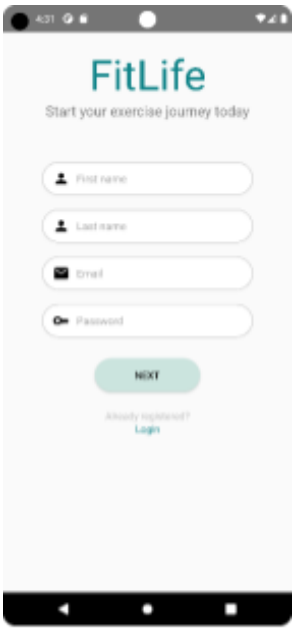


Figure 2

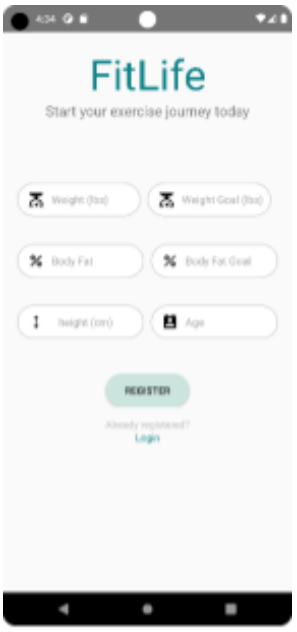


Figure 3

Once you press the register button, you will be greeted with the first screen (fig 1.). Enter your user details and press the Login button.

Congratulations! You have successfully created a FitLife account! Welcome to the FitLife Community!

7.2 The Home Page

When you successfully create your account and login, you will be greeted with the home page (fig. 4). You will see two progress bars, determining how close you are to your goal. Pressing the white + floating action button on the right will take you to this page (fig. 5). This will allow you to keep track and see your consistent progress. Once you enter your new weight and body fat information, make sure to put in the correct date the information was recorded, and press the Add button. As shown in the next screen (fig. 6), you will see a visible change in the weight graph. Pressing the Weight button below the chart will switch between 2 charts, as seen below (fig. 7).

Congratulations, you now know how to navigate through the home page!

Figure 4

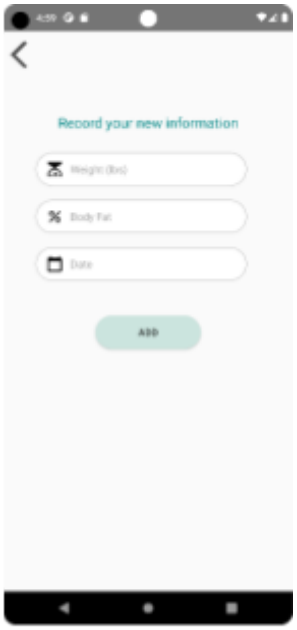


Figure 5

Figure 6

Figure 7

7.3 The Discover Page

You may have noticed a little navigation bar at the bottom with different icons, this is the primary way to navigate through the app, to access the discover page, tap on the small search icon next to the home icon. Once you press that icon, you are greeted with the following page (fig. 8). Here you can search for or scroll through and

discover different workout routines. You have many options on this page, pressing on a workout card will lead you to more details about that workout (fig. 9). Pressing the bookmark icon on the top right will save the workout for you to refer to.

The search bar works as all search bars do, once you enter a word, it will filter out any non matching results, next to the search bar is a filter button that allows you to further narrow down your searches. When you press on the filter button this is the screen you are greeted with (fig. 10). The filters sort out the different program difficulties, allowing you to select from 1 to all 3 options to filter through. Once you select the appropriate filter, press the Apply Button and you will be greeted with a list of workouts that match your requirements (fig. 11).

Once you have looked through all the discovery page workout routines, you can press the bookmark inside a book icon and will be led to the saved workouts page.

Congratulations! You now know how to navigate through the discover page and save workouts!

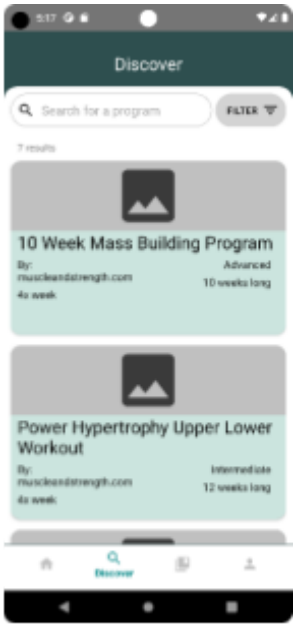


Figure 8

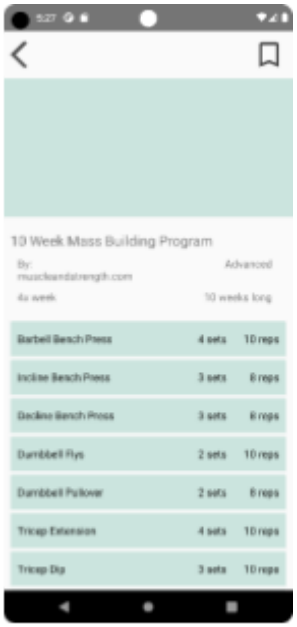


Figure 9

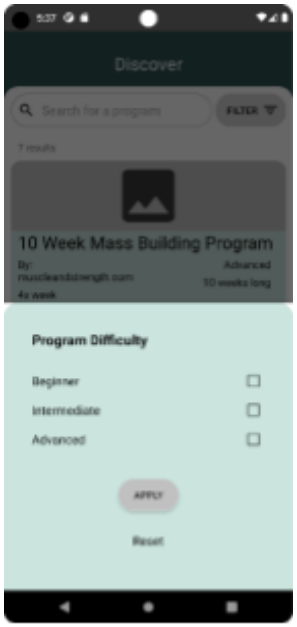


Figure 10

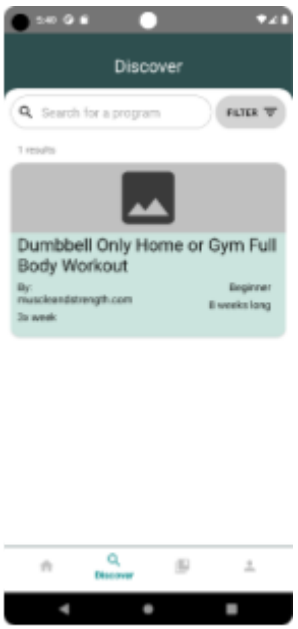


Figure 11

7.4 The Saved Workouts Page

This is the Saved Workout Routines page (fig. 12). Here you will be able to see workout routines you save from the discovery page and routines you created

yourself. Pressing the white “+” floating action button on the top right will take you through the creation process for a new workout routine. You will first be greeted with a form asking you the details of the workout routine you wish to create (fig. 13). Once you fill out the information and press “Create Routine”, you will be taken to the next step, adding workouts to your routine (fig. 14). You can fill in the information for each workout, once you press “Add to Routine”, you can add as many workouts as you want before pressing the back button in the top left.

Once you create your routine and add your workouts, you press the back button and go back to the saved workouts screen (fig. 12). You will find your created routine under it, if you press on the routine, you will find it similar to the workout details page on the discover page (fig. 9). With one big difference, instead of a button to save or unsave, you will see a trash icon to fully delete your workout routine.

Congratulations! You now know how to navigate through and create and delete saved workouts!

Figure 12

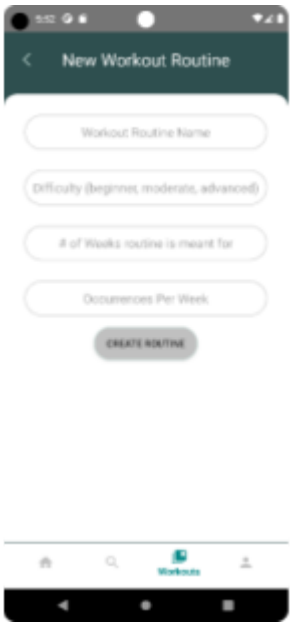


Figure 13

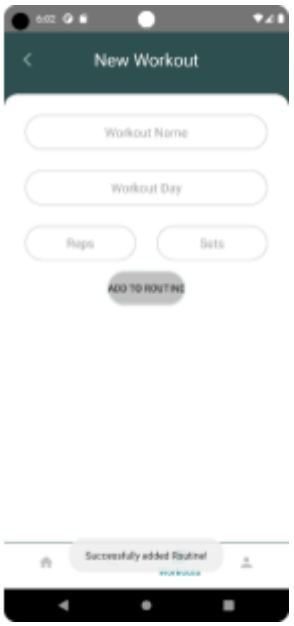


Figure 14

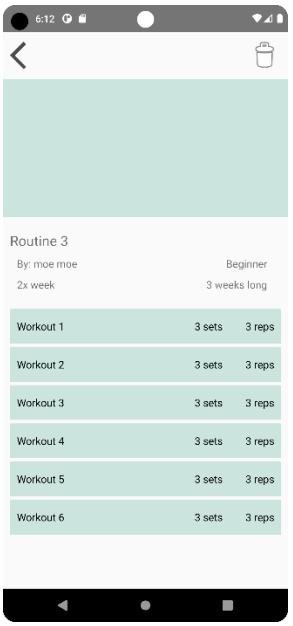


Figure 15

7.5 The Profile Page

The last icon on the bottom right, is your profile page (fig. 16). Here is where you can change app settings or edit your personal details. Pressing the settings icon in the top right of the page will take you to the app settings screen (fig. 17). Here you

can change the measurement units used for Weight and Height from imperial to metric or vice versa. You can also enable App Lock. Below all these options, you can choose to logout and create a new account or log into another existing account, or you can choose to delete your account entirely once you have reached your fitness goals! When you press the back button, you will again be greeted with the profile page (fig. 16). You can press the pencil icon on the right side of your name, and you will be taken to a screen to edit all your user details (fig. 18). Here you can edit any information about yourself including your starting weight, your height, along with your name or age. There is also an option to take a profile picture with your camera or upload a profile picture from your camera roll.

Figure 16

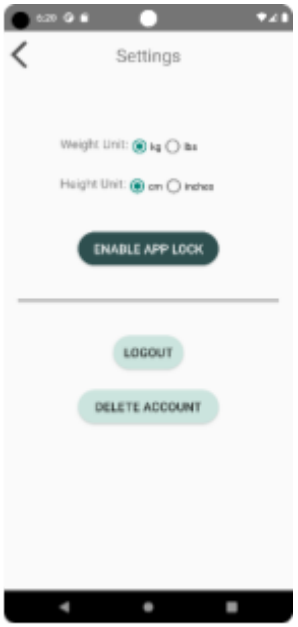


Figure 17

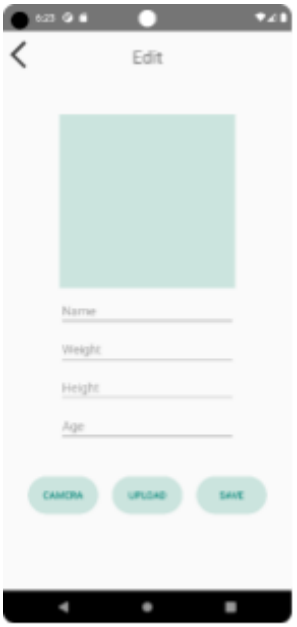


Figure 18

Congratulations! You now know how to navigate through and make changes to your profile and user details, along with being able to customize your app experience!

This concludes the User Manual.

We hope you enjoy your FitLife experience and fulfill all your fitness goals!

8. Conclusion

FitLife was created to help people of all fitness levels to reach their goals without worrying of companies taking advantage of them. We provide a free, secure, and reliable app in order to combat these types of companies. However there are still a lot of limitations with the app that hinder the experience of our users. One of the larger limitations is that our database is not server hosted and rather per-device hosted. This means there is no communication among users and we believe having a community is very important in fitness. This is something we want to implement in the near future. Other features we want to include to provide a better service is to include more of the diet aspect of fitness. Currently our app has no calorie tracker nor information for food. In conclusion, this is a very early stage of FitLife and we believe in future iterations, this app will be essential for all users in their fitness journey.

8.1 Github Link

https://github.com/EliasTkla/Mobile-Applications-Project