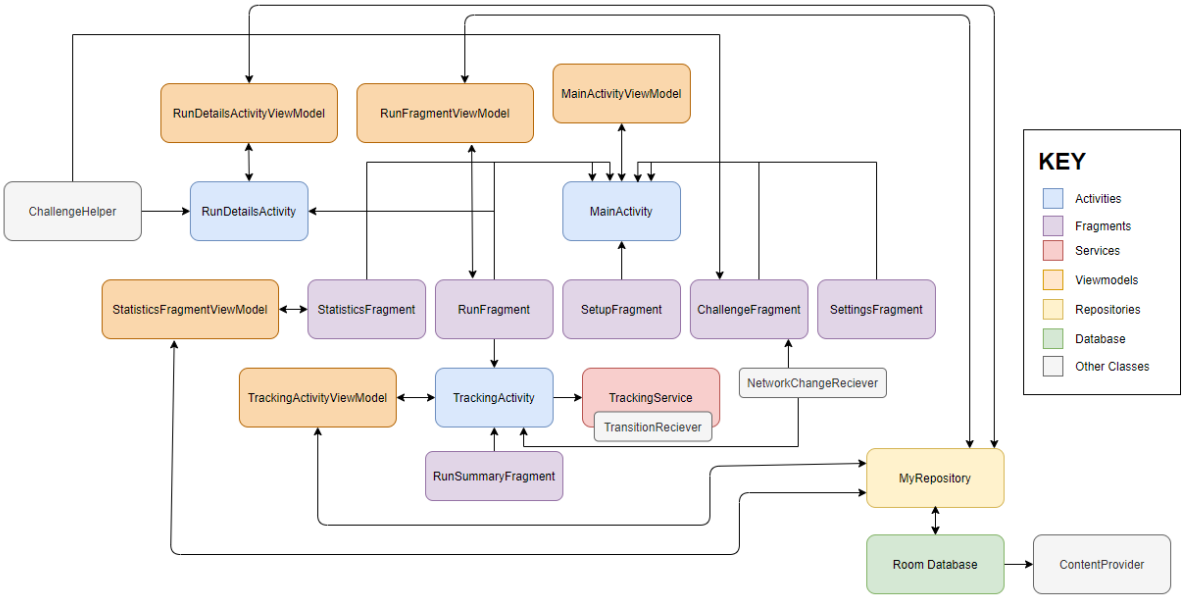
RunApp

MOBILE DEVICE PROGRAMMING

Nadim Rahman
14323693

Architecture



The diagram above displays shows the main classes involved with this app. There are several other classes also included in the app which is not included in the diagram as they do not represent the main components of the system.

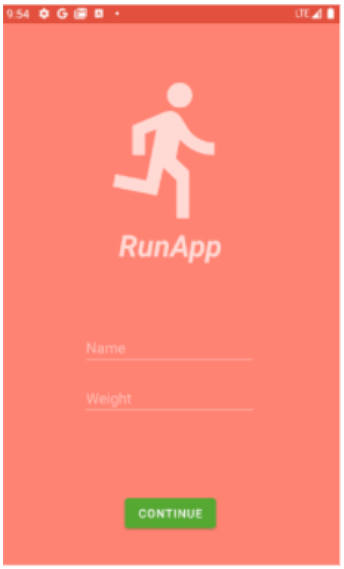
1. **Main Activity, Host Fragments, MainActivityViewModel**: This activity is the most important activity in the application, and it responsible for hosting several different fragments. The reason I have utilised fragments for this application is because communicating between various fragments is much easier than sending data back and forth between activities. The bottom navigation view component in the main activity (which is shared in the view for each fragment) allows me to switch between different fragments easily using backstack manipulation. The MainActivityViewModel class allows data to be retained when the activity is destroyed. This viewmodel class is shared between the different fragments (although in this case, only the setup fragment needs it). One key purpose of the viewmodel is to retain the current fragment that the user is currently on when the screen is rotated.
2. **StatisticsFragment, StatisticsFragmentViewModel**: This fragment displays daily, weekly, and all-time statistics, as well as useful charts using the MPAndroid Library. The reason I have used a separate viewmodel for this fragment is because there is quite a lot of data that this fragment requires which is not needed for the other fragments, so it was more efficient to break the classes into smaller sections. This is the same case for the RunFragment.
3. **Tracking Activity & Tracking Service:** An android service is a component that is used to perform operations in the background, which in this case is to track the user's location. I have utilised a bound service in this case as it allows the activity to interact with the service through the use of a binder object. This allows the activity to pause/resume/end the service using the UI buttons. Throughout the project, I have utilised LiveData to observe changes, and this is the case especially for this service. The Tracking activity will observe changes in these live data variables in the service to update the UI (for example, the duration of the run, and the path that the user is taking). The service is unbound from the activity whenever the user exits the

activity and rebound whenever they enter the tracking activity. The service is a foreground service as this allows to display notifications to your users when running long lived background tasks. The notification acts like any other notification, however it cannot be removed by the user and lives for the duration of the service. Unlike a background service, a foreground service will not be killed when the system is low on memory, which is useful in this case is it won't delete a run randomly. The service keeps track of the duration of the run and also the amount of time the user is idle. An inner class broadcast receiver is within the service class as this allows the receiver to directly make changes and calls to service methods to control the amount of time that the phone is 'idle' whenever the activity recognition API detects a new change in activity.

4. **Challenge Fragment & Network Broadcast Receiver:** The challenge fragment displays the challenges posted by users which is obtained from the firebase database. A firebase database was used instead of Room as Room is only designed for offline use. This fragment is registered to a network receiver in order to let the user know if the network is unavailable, as they will be unable to view challenges in offline mode. Unlike the transition receiver, the network receiver is a separate class to allow other classes to register this broadcast, which is a possible feature for future additions to the app.
5. **Settings / Setup Fragment:** These fragments allow the user to choose their name and weight which are stored as shared preference variables. Shared preferences let you read and write key value pairs in a couple of lines easily, which is why I chose this option rather than storing them in the database.
6. **MyRepository & RoomDatabase:** I have stored all the run data in a Room database. This database layer is on top of the SQLite database that takes care of mundane tasks that I used to handle with a helper class, such as SQLiteOpenHelper. It provides easier local data storage. The Room database uses the DAO (RunDao, PhotoDao) to issue queries to the SQLite database. I have used the repository pattern to abstract access to the database. The repository pulls data from Room as LiveData objects that populate the ViewModel. The Activity binds the ViewModel to appropriate UI components.
7. **Content Provider:** The content provider allows other applications to access the run data from the application. In particular, it allows applications to gather all the runs from the database.
8. **Recycler views:** These views were used to display content from databases. These were used instead of ListViews as the Android recyclerview extends this concept and improves the way this adapter is used to increase the overall performance.

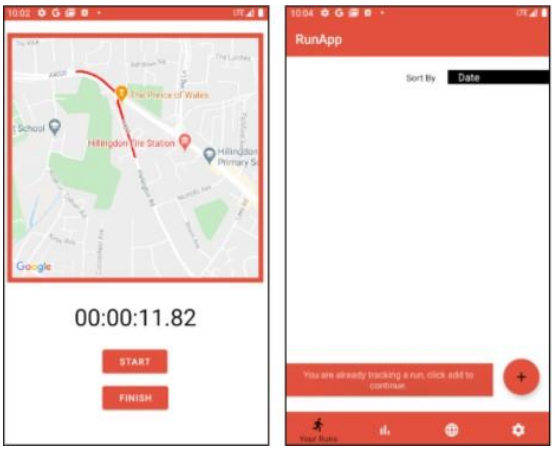
Behaviour of the Application

1) When the user first enters the application, they will be led to the setup fragment. The fragment will appear with several animations and fields which will ask the user for their name and weight. The data is validated and stored in the shared preferences. This fragment will only be loaded on the first time of using the app.



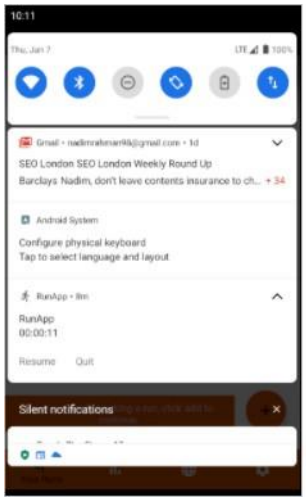
- Professional introduction
- Allows profile to be used to post challenges
- Allows calories burned to be calculated

2) The user is loaded into the run fragment and can easily switch between the fragments using the bottom navigation tool. The run fragment will initially be empty. To create a run, the user can click the floating action button which leads them to the tracking activity. The user can then click 'create run' which starts the service. However, before they can do this, they must accept the permissions required.
3) Once the user has started a run, they can pause/resume the run each time they click the toggle button. The service resumes until the user finishes the run. The service will track the user's location and add path points to the map each time they change their location. The map will animate in the direction of the user's movement. The user can navigate to other activities whilst the service is being run. Each time the user clicks the floating action button in the run fragment whilst a run is occurring, they will be displayed with the details of the currently tracked run.



- Logging the movement of a user when they go running or walking (specification point)
- Path points allow user to track run
- Start and stop buttons allow users to pause if obstructions occur during a run
- Stopwatch allows user to keep track of time
- Allowing user to navigate to different activities makes it more user friendly

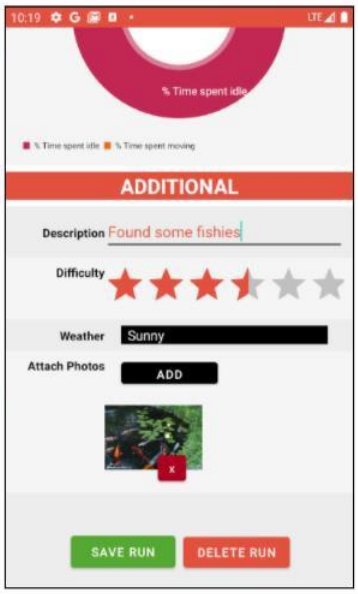
4) The user can control the current run through a notification. The notification provides several different features, such as displaying the current stopwatch time in seconds, allowing the user to pause/resume the run through the notification buttons, whilst also allowing the user to kill the service. The UI is updated whenever the user performs an action on the notification.



Notification allows user to keep track of run without needing the app open in the foreground

Resume/Pause/Quit buttons allows user to control run without app being open. Allowing a more user friendly interface.

5) When the user completes a run, they are led to the run summary fragment, which displays the key details of the run, such as the final map path, the distance, speed, calories burned and duration. The user is able to view a pie chart which displays the proportion of time they were idle during the run (Note: Mock locations have no effect on the activity recognition algorithm used by Location Services. Use a physical android phone to test this functionality). There are several inputs available that allows the user to annotate their run, such as setting a description, rating, weather, and uploading several photos. Finally, the user can save the run to the database or delete it.



Saving the movement data in an appropriate manner (specification point)

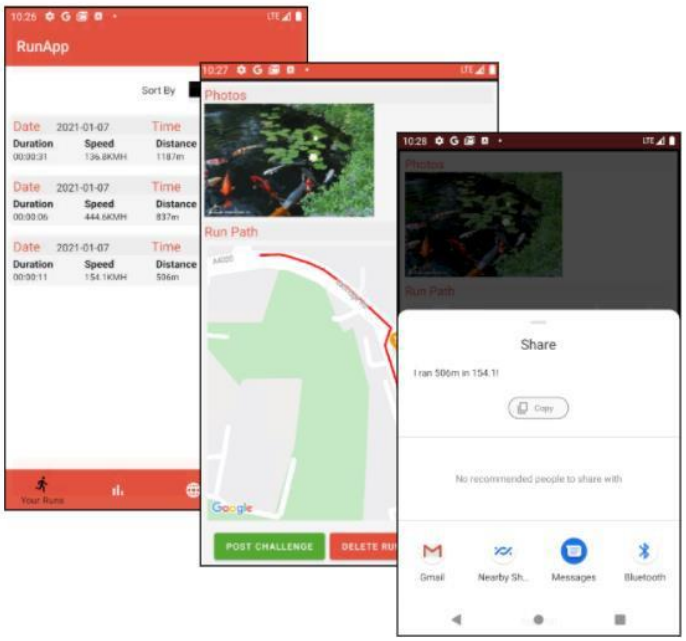
Allowing the user to annotate their data in a useful manner (specification point)

Summary fragment allows user to see how well they did

Pie chart allows user to see how long they take breaks for, allowing for future improvement

6) After a run is added, the recycler view in the run fragment is updated. This fragment can display the runs in different sorting orders, such as by date, distance travelled, calories burned, etc. When the user clicks a specific run from the list, they can inspect more details, such as the map path and photos associated with the run. You can additionally 'post a challenge' which will append the run to the online firebase database and be displayed on the

feed in the challenges fragment. Additionally, you can delete the run / share the run on different social media platforms.



- Allowing the user to inspect their data in an appropriate manner (specification point)
- Sharing information gives the user the opportunity to share their achievements. Also allows other applications to make use of data (specification point)

7) The statistics fragment allows the user to inspect their data in a useful format. They can find information such as their daily statistics, weekly, and all-time statistics. Using the graphs, they can inspect their best runs, as well as inspecting the amount of distance they have travelled for each day of the week.

- Allowing the user to inspect their data in an appropriate manner (specification point)
- "How much have I run today?" (Daily statistics)
- "Have much have I improved?" (All-time statistics, Line graphs)
- "Have I ran faster than my best time?" (All-time statistics, Line graphs)
- "How much have I run every-day this week?" (Weekly statistics, Bar chart)

8) The challenge fragment displays the runs posted by other users worldwide (limited to the most recent 10 challenges). This fragment requires internet connection to access the firebase database. A broadcast receiver will display a message if an internet connection is not available.

- Network unavailability provides useful information to the user (user-friendly)
- Challenges allow users to compare how they do against users in the World (Gamifies app)

9) The settings fragment allows the user to update their user profile.