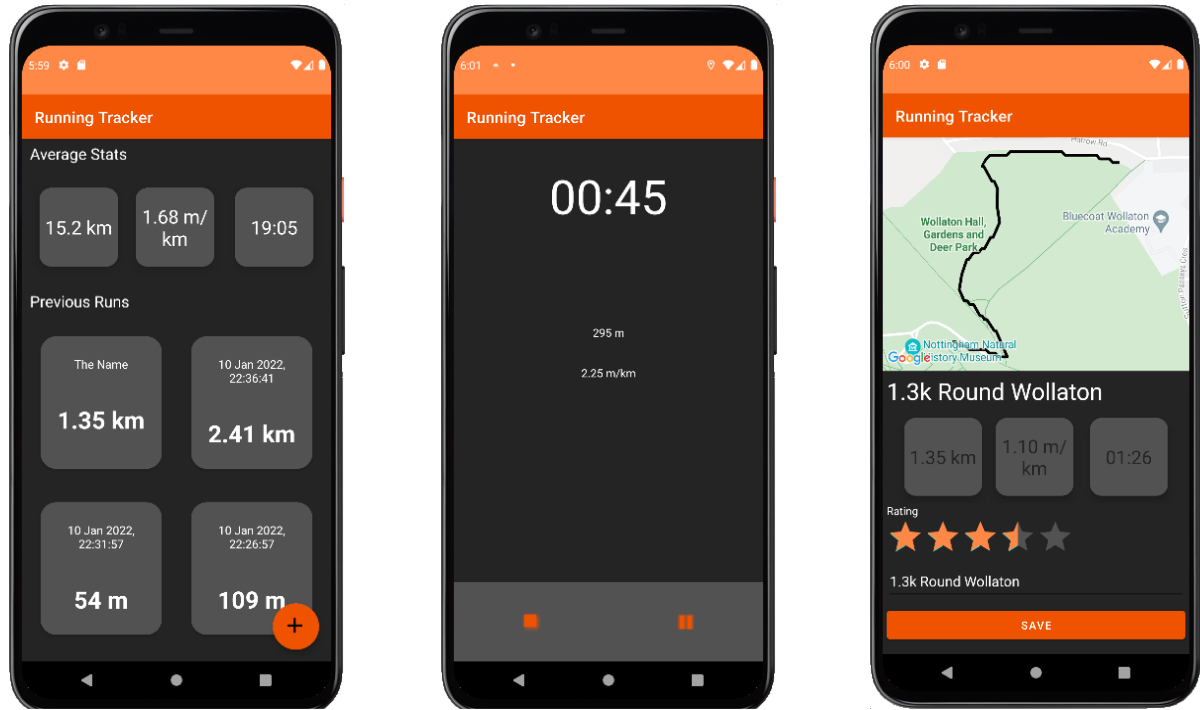


Overview




The application records any activity using the GPS location, although it is primarily targeted towards runners. The user can start tracking a new run by tapping the plus button, which takes the user to the second screen. From here, the tracking begins immediately; the user can see their current time, distance, and average pace. They can pause the run when they need to take a break. Once finished with their run, the user can press the stop button to finish tracking their workout.

The user's previous workouts are saved on their device. From the home screen the user can scroll through a list of their previous runs, which show the name of the workout and the total distance covered. This list loads more runs in as the user scrolls down. At the top of this screen, the user can see 3 top statistics over their running lifetime. The total distance covered, total time spent running, and average pace can all be seen at a glance. This allows the user to track their running progress over time and evaluate their performance.

In the list of previous runs, the user can tap on a workout card and see details about that workout. The app plots their route on a map - this allows the user to see exactly where they ran. Below this the user can see statistics about this run - the distance covered in kilometres, the duration, and the average pace over each section of the run. The user can also rate this run out of 5, so they can filter their runs by their own metric - whether that is performance, speed, or anything else. Finally, there is a text input where the user can give the run a name. This helps them sort their workouts by memorable routes or performance, for example. They can add any relevant notes they want here.

Notifications

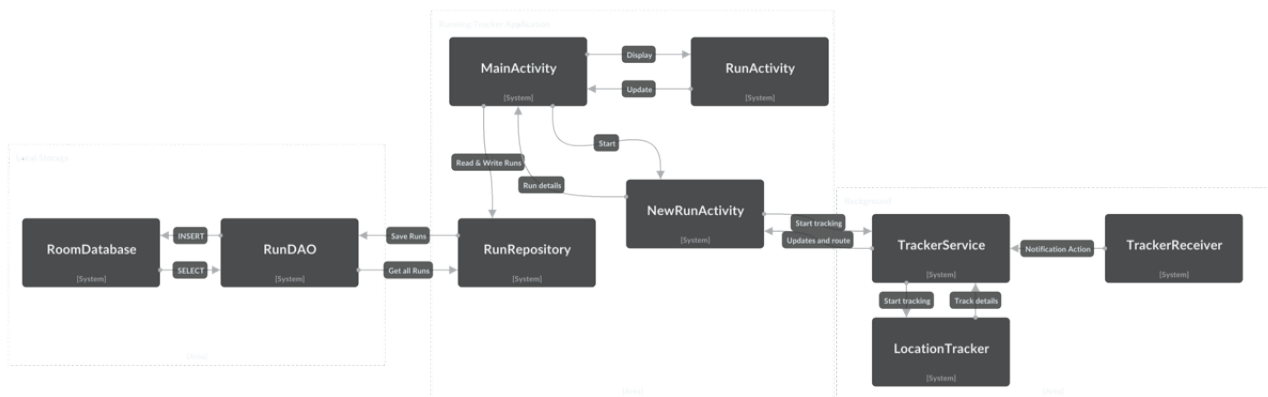
 Running Tracker 

Tracking run
49 m, 00:11

PAUSE

The application continuously tracks the user in the background, once given permission, so that the user can close the app whilst they workout. The app will also show a notification to the user, showing their current stats such as distance and time, and allows them to pause or continue the activity tracking without having to open the app.

Architecture



The Application is built using the MVVM architecture (Model-View-ViewModel). This is a software design pattern that aims to cleanly separate the presentation and business logic from the user interface. This helps decouple components, so that the same business logic could be used with a different user interface, or the application could be refactored to save workouts to a networked backend rather than local storage without having to update the user interface, for example.

This separation of concerns also has other benefits, such as making it easier to unit test logic, or test the UI without connecting to a real backend. For a project of this size, the benefits of MVVM over a simpler architecture such as MVC are marginal; however, were this application to continue in development, with more features added, MVVM would deliver a scalable architecture that allows a team of developers to build a stable, high quality application whilst keeping the code clean.

The View in this case is the three Activities:

- Main Activity
- Run Activity
- New Run Activity

These use a ViewModel, RunViewModel, to connect to the Model, the RunRepository. This allows the application to interact with a local SQLite database saved on the phone's storage. It also means that the current data is persisted even if the UI components are killed by the OS, as a ViewModel's lifecycle is longer than that of an Activity - for example, when rotating a phone, the Activity is restarted, losing any data held by that Activity.

The Activities and ViewModels are part of the *UI layer* of the app, and can be started or killed at any time. The Repository and Model are part of the *data layer*, which persists data permanently for the UI layer to use.

Main Activity

This Activity has the largest responsibility, as it is responsible for interacting with the data storage repository. It queries all runs from the repository to populate the RecyclerView of cards. These cards are built using the CardView component built into Android. This provides a Material-Design-confirming component that clearly shows the separate runs to the user. To handle the on-click action, the RecyclerView Adapter takes a callback function, which is passed from the MainActivity. This allows the MainActivity to navigate to the RunActivity when a card is pressed. The Activity fetches the list of Runs from the RunViewModel. It is also responsible for passing new and updated Runs to the view model for handling.

Run View Model

This class is an interface between the Run Repository and the Activities that utilises the data. It stores the UI data in a lifecycle conscious way, so that data is persisted during UI config changes such as rotating the phone. This class is responsible for converting the Flow list type received from the Repository to LiveData that the Activity can observe and display to the user accordingly. This class also has methods to assist the Activity in inserting and updating Runs - it calls the Repository in a coroutine so as not to block the main thread.

Run Repository

This class is a wrapper around the Run DAO. It centralises any changes to the data from across the app, so that you don't have multiple sources trying to write data in their own way, each overwriting each other or using an incorrect format - this repository acts as a single source of truth for the running data. It allows the data layer to be easily swapped out for another source - for example, refactoring the app to save Runs to a remote, network-accessed, backend is relatively simple, as the UI layer does not need to change thanks to this wrapper class - the interface would be the same no matter the data source. Or, an in-memory cache could be easily added. Again, this separation of concerns helps keep the app more maintainable and testable. For example, in a unit test of the view model, the repository could be mocked so that any test data is not actually saved to the local database.

Run & RunDAO

The Run class defines a data model to be stored in the database.. This class is decorated with Room decorators to create a table to store Runs in the database.

RunDAO provides an interface for other components of the app to interact with this model. The interface defines a set of methods, each with a SQL query attached, which is then used by Room to generate the database interaction implementation.

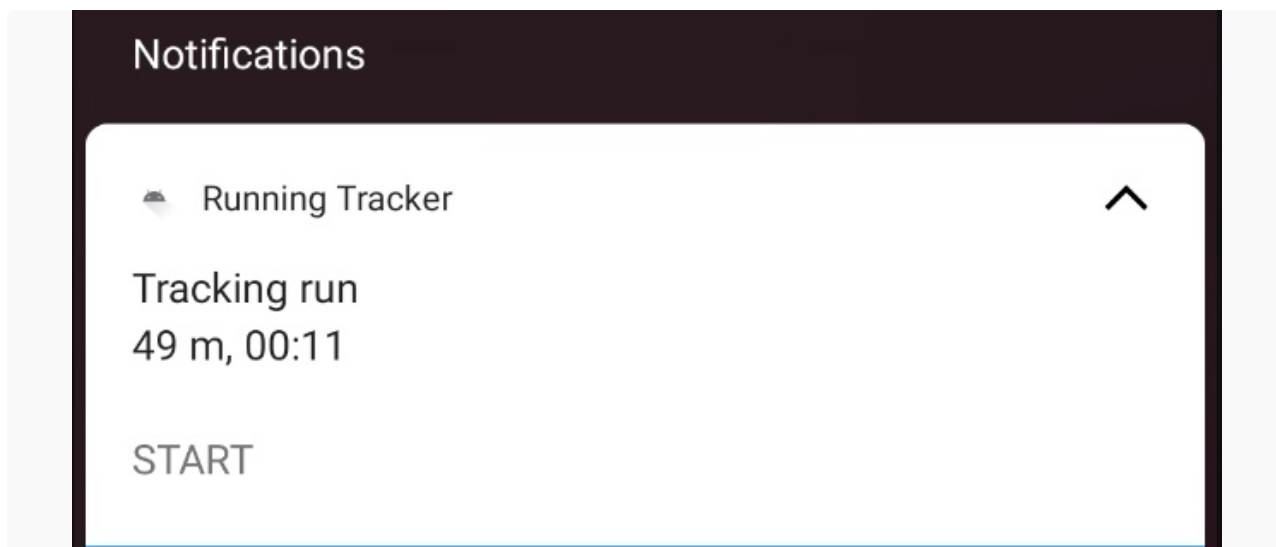
The MainActivity is responsible for interacting with the data sources, via the view model. It also handles redirecting the user to the other Activities in the app: NewRunActivity and RunActivity.

New Run Activity

This activity handles the workout tracking components. It does not do the tracking itself; instead, this is delegated to dedicated components. This Activity starts the tracking immediately on start. The user's current time, distance covered, and average pace are displayed. There are also buttons that allow the user to pause or stop the run. When the run is stopped, the data is collated, packaged into a Run object, and passed back to the MainActivity. When the run is paused, the TrackerService's pause method is called, of which this activity is bound to in order to receive status updates.

Tracker Service

This service contains the main tracking logic in the app. When this service starts, it immediately starts listening to location updates using the LocationTracker class. As this is a Bound Service, it exposes various details about the current workout, such as state, distance, and pace, so that the NewRunActivity can update the statistics in realtime. This service also shows a notification to the user, showing the current distance covered and allowing them to pause or continue the current workout. This notification is updated regularly with the current state and distance - sometimes, Android will rate limit this notification update, so changes may not always show.



The service registers a Broadcast Receiver to listen for user action on the Start/Pause buttons; this is an inner class, therefore can access the play and pause methods of the service, rather than an external class that would not be able to access these service methods.

As this is a Bound Service, it exposes various details about the current workout, such as state, distance, and pace, so that the NewRunActivity can update the statistics in realtime. These are all calculated by the Track class, and its sub classes Segment and Point. The workouts are stored in the Track data structure, which uses composition to simplify the storage and abstract logic such as distance and pace calculations.

Run Content Provider

This provides external applications with access to running data. Other applications can query for a list of runs, or get details about a specific run.