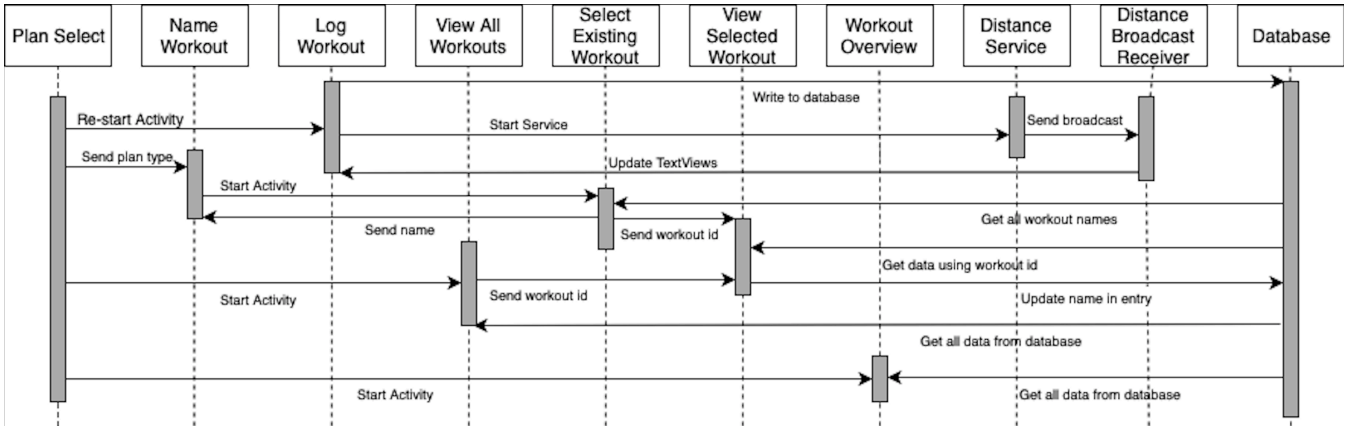
Mobile device programming CW2 Report

Introduction:

My solution to the running tracker application consists of seven activities, a broadcast receiver and a service. Each activity is designed to streamline the user's experience allowing them to easily access every function the app has to offer. The app has three workout 'modes' available; walking, jogging and running. The user is capable of selecting one of these modes to track their workout, and the app tells them their distance travelled, time taken so far and current speed. The user has a few different methods of progress tracking available to them, allowing them to see their progress from different angles.

Interaction diagram:

The diagram below shows how all the activities, services, broadcast receivers and database interact with each other. You can see the flow of data and the navigation paths possible.



Plan Select Activity:

The first activity the user will see is the Plan Select activity. This is the centre of the application and allows the user to navigate to any part of the app.
The actions the user may perform are:

- To select the workout type, where they will be taken to the NameWorkout Activity.
- To view progress depending on the option selected with the spinner, being taken to the appropriate progress activity.
- To navigate towards a daily, weekly and total overview of all their workouts by taking them to the WorkoutOverview activity.

When this activity first loads, the apps' permissions are checked for the fine location permission. If this permission has not yet been granted, the permission is requested.
At the top of the onCreate method the activity checks if the tracking service is running, if so, this means user is still working out. To return the user to the Log Workout activity the shared preferences are accessed to determine if the activity was destroyed. If it was, then a

new instance of the activity is created. Otherwise, it is next on the stack and so the Plan Select activity can finish to return the user to monitor their workout.

Name Workout Activity:

This activity allows the user to select one of three options before starting their workout. Here the user can select a workout already named, if they are doing the same route, name the workout and give it a new name, or proceed without naming the workout. This allows users to track progress for a given route, or if the user isn't going to do the route again, they might not want to name it, or name it at a later time. If the user selects a route they have done before, they are taken to the SelectExistingWorkout activity, otherwise they are taken to the LogWorkout activity to begin their workout.

Select Existing Workout Activity

This activity queries the database to obtain all the named workouts and display the name in a ListView without repeating names, allowing the user to select a workout name. This has two purposes within the app; to select a workout the user has already done and to select a workout that they wish to view their progress. Using bundle, the activity can obtain the parent activity to determine where they should send the user once they select a workout.

Log Workout Activity

The LogWorkout activity only has one button, to allow the user to start and stop their workout.
When the activity is first created, the bundle data is retrieved containing the workout type and name. A connection to the database is also established.
When tracking begins, if the user has not granted the required permissions a Toast message will be displayed informing the user of this, and logging will not start. Otherwise, the start time and date are recorded and the DistanceService is started and bound to. This allows the activity to interact with the service. Finally, the relevant TextViews are passed through to a DistanceBroadcastReciever, which is registered using a LocalBroadcastManager. Now the service is tracking the run and sending broadcasts to the receiver, updating the TextViews.
When the user wishes to stop their workout, they press the button again, triggering the activity to obtain the overall distance travelled from the service, using the binder. The service is then stopped and unbound. The LocalBroadcastManager unregisters the receiver and the run data is inserted into the database, before closing its connection. The activity then finishes and returns the user back to the PlanSelect activity.
To ensure all lifecycles are complete, when the app isn't user facing and the onPause method is called, the broadcast receiver is unregistered as the TextViews don't need updating. When the app becomes user facing again, the onResume function is called, and the receiver is reregistered.
When the app is destroyed, we want to continue tracking so the workout data is saved in the shared preferences for recovery when the app is restarted. The database connection is closed, the receiver is unregistered, and the service is unbound, but still running. When the activity is created again it will have been created by an instance with no bundle data. In the onCreate, if there is no bundle data then we obtain all the saved workout data from the

shared preferences, the database connection is re-established, the receiver is re-registered, and the service is re-bound.

Workout Overview activity:

The general statistics of distance, time and total workouts the user has done are calculated and displayed within this activity. These are daily, weekly, and overall statistics. For example, the individual exercises the user has done in a week is displayed with the overall total. The database is queried to obtain each workout. Each workout is then assessed to see if it has taken place that day or within the week. If so, the relevant statistic data is updated. Once all of the workouts have been evaluated the statistics are displayed to the user in TextViews.

View all Workouts activity

This activity displays the workouts in a ListView with a custom adapter according the option the user selected. For example, if the user selected to see all walks then every walk workout will be displayed. The user is also able to sort the list by distance, time or date.
They can select a workout from the list to view more details in the ViewSelectedWorkout activity.

View selected workout activity

When a workout has been selected, the ID of the workout is passed via a bundle. The activity can then query the database to obtain all the information surrounding the workout. These are displayed neatly to the user in TextViews. The user can go back, update the name or delete the entry. Pressing the “update name” button displays an alert box with an EditText where the user can enter the new name. Pressing “change” then updates the name using the content provider. The “delete” button removes the entry from the database and returns the user to the ViewAllWorkouts activity.

DB helper, DB content provider and DB provider contract

These three classes are used to provide all functionality needed for the database. The DB helper creates the database and tables with all the specified columns and the DB content provider is the main interface the app has with the database. The content provider allows the app to update, delete and insert new entries into the database. The DB provider contract contains all the constants used within the database to ensure no errors are present.
The database used in the app consists of 7 columns: “id”, “type”, “time”, “distance”, “timeDate”, “date” and “name”.

Distance Service

The distance service, implemented as a foreground service, performs the user tracking and timekeeping for the workout. The service is only created when the user’s workout begins, so as soon as the service is created we can start tracking. On creation, the starting time is

logged and a location manager is created and requests updates using a location listener. When a location change is detected, the distance between the user's last location and current location is calculated and added to the overall distance. The service will then keep track of the time and distance travelled, even when the user closes the app.
Every second, a local broadcast is sent containing the time, distance and speed of the user. Broadcasts must be sent every second to allow the time TextView to update.
The user is capable of returning back to the app with the notification sent out by the service. The service also makes use of a binder, to allow the activity to be bound to the service and interact with the service, meaning the activity can stop the service and get the data it needs.

Distance Broadcast Receiver

This broadcast receiver takes four TextViews and the workout plan on creation. When the receiver is registered and begins receiving broadcasts it is able to display the received data in the TextViews supplied. The receiver also uses the plan type and user speed to display if they are travelling too slow or fast according to averages found on google.