Technical details

This is a standalone Multi-Page Application. Each page has a corresponding dart file for page rendering and logic control. The entities saved to the internal storage as a JSON file.

Model

This program contains 3 nested models in this program. `TripList` is a collection of Trip, which is generated every time we start a new tracking. Every single `Trip` contains metadata like starting time, ending time and the type of vehicle. Each trip includes a list of recorded `data`, including timestamp, accelerometer, userAccelerometer, gyroscope, magnetometer, location and speed. Here's a sample model in JSON format.


```
{
  "trip": [
    {
      "startTime": "2021-07-20 20:18:04",
      "endTime": "2021-07-20 22:18:04",
      "vehicleType": "Bikes",
      "data": [
        {
          "timestamp": "2021-07-20 20:18:04",
          "accelerometer": [
            1.5,
            9.5,
            -1.5
          ],
          "userAccelerometer": [
            1.5,
            2.5,
            3.5
          ],
          "gyroscope": [
            1.5,
            2.5,
            3.5
          ],
          "magnetometer": [
            41.5,
            -3.5,
            -26.5
          ],
          "location": [
            54.5978125,
            14.7855416,
            30.1879124
          ],
          "speed": 15.05
        }
      ]
    }
  ]
}
```

Every time the app is launched, it firstly checks if the history data is saved in the internal storage (for Android) and tries to initialize the Model. We use `json_serializable` to generate to/from JSON code for the model. The class is annotated with `@JsonSerializable()` and the JSON key member is annotated `@JsonKey(name: 'name_of_key')`. Then run the code generator to generate the missing `.g.dart` generated dart files.

The model object will be loaded from the local file when the app is launched and saved to the file when tracking is ended, or the user manually deletes all the records.

View

In Android, a page corresponds to an Activity. In iOS, it's a `ViewController`. In Flutter, a page is just a widget. Screens and pages are called routes. This app uses `Navigator` to navigate to different routes. There are 5 views in this program, HomePage, RecordPage, SettingPage, TrackingPage and TripPage.

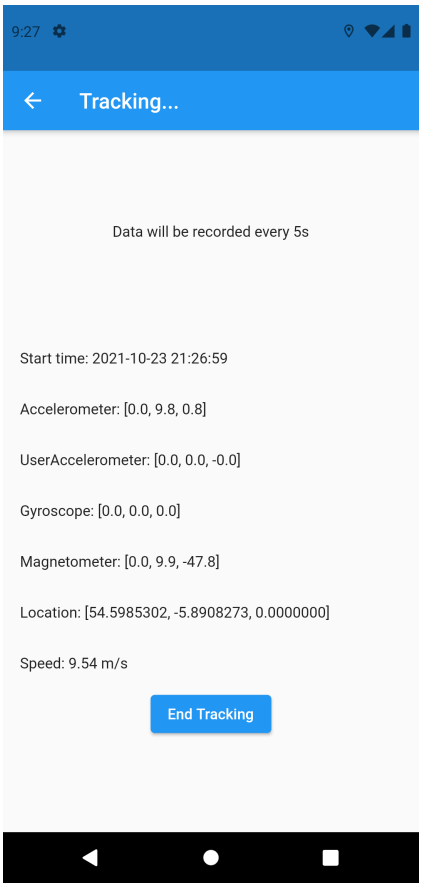
Homepage

This page only contains a dropdown list to choose a vehicle and a start tracking button, which will navigate to the TrackingPage.

TrackingPage

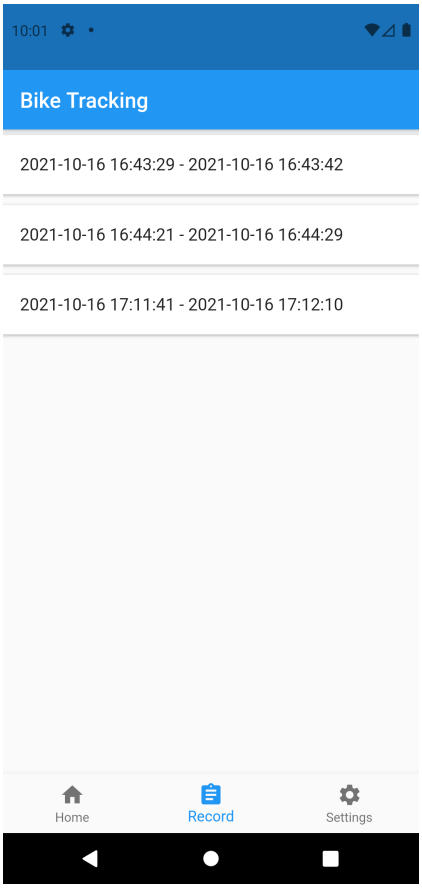
The tracking page shows the real-time sensor and GPS data. Note that GPS data is provided by `geolocator`, which is a cross-platform API for generic location functions. To use this function properly, the user must allow the storage and location permission. The sensor data is provided by `sensors_plus`, which is a Flutter plugin to access the accelerometer, gyroscope, and magnetometer sensors.

Each of these sensor events is exposed through a `BroadcastStream`. So this page is a StatefulWidget. All the streams can be accessed via StreamSubscription.



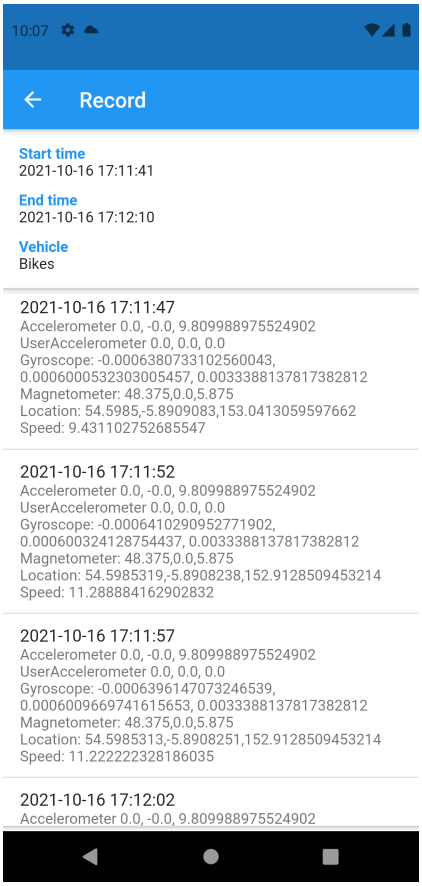
RecordPage

This page consists of a scrollable ListView with ListTile inside that shows all the recorded trips on the phone. The title is the starting time and ending time. Clicking each tile will navigate to the corresponding TripPage.



TripPage

This page shows the details of a trip. It consists of a container with some time and vehicle and a ListView, which displays all the interval recorded data in a raw data form.



SettingPage

This consists of a slider to change the time interval for recording and a button to delete all the saved data.

Controller

To achieve simplicity, all stateless widgets constructors visit the model and render the components directly. There are 2 controllers in total, which loads data from a JSON file, save data to a JSON file and delete the file when the user decides to.

The controller is in the Global class, which is a singleton class. It also maintains some shared data between the pages.