

BikeTracks App Development

COMPSCI 399 Capstone Course Project Report

**Team 8 Members:**

Yash Sood (801276455) Back-end Developer

Youlun Li (924217658) Front-end Developer

Yuyuan Wu (560718000) Project Manager

Aaron Fan (231048660) Back-end Developer

Zixuan Su (854672611) Back-end Developer

Shan Li (808991187) Front-end Developer

Sura Liu (872828739) Software Tester

Version Updated : 27/10/2021

**Executive Summary**

This report is intended to demonstrate our group project. We have designed a GPS tracking application for different vehicle types users have used for their daily commute. Our project is trying to address daily users who want to track their average daily commute speed and geometer location. For our BikeTrack application, it will record users devices sensor data for accelerometer, geometer, GPS location and Speed by using multithreading to monitor users action. In order to achieve these functions, we have used Flutter and Android Studio to develop our project.

During our design phase, we first designed a prototype of the user interface. It is necessary to design the prototype of the user interface before the implementation phase. Our team made a small group and selected a few people who have learned about human-computer interaction to start the prototype design first. It was considered that the designed prototype of the user interface should be aligned with design principles for mobile which include user control, feedback, aesthetics, consistency and forgiveness. Also, we defined the features scope of our application and determined all necessary pages of the user interface. For our user interface prototype, there are three pages which include homepage, setting and records. Users can be navigated to different pages to achieve relevant features through the navigation bar. And we firstly delivered two versions (network features or non-network features) of prototypes for our user interface. After meeting with clients, we finally determined the prototype with non-network features. Once we did the prototype it helped us easily to build the front-end interface. As the prototype will determine each page’s layout and functionality, our team members can just separate, build up each page, and then merge them together.

Moreover, for our database we decided we will use a single text file that stores inside users device for accessibility. The reason for it is because we want to increase the readability and ease of access for our application performance. For the record and tracking page, we have used package geolocator, Sensor\_Plus, and Intl in order to record users devices' sensor data. However, in order to enable the GPS function, users must enable the permission for our application to access their sensor data. Hence, we have added a function that tests the GPS permission for our application to encounter different scenarios. In addition, we have added a clear all records button that intends for users to clear all the records they have travelled in case the database becomes too large and slows the performance of our application.

After accomplishing our development process, we have finished our BikeTrack application that allows users to change the recording time interval for the data, records for users accelerometer, geolmeter, GPS location and speed information based on Android Environment.

For more future work, we could connect our application with the internet so that we could add a history route that users have travelled and we could add another weather forecast function to inform the user if they want to travel either by bike or scooter.

# Table of Contents

[**Introduction**](#_heading=h.1kix3uj07w16) **3**

[**Background**](#_heading=h.p4vb2rl8askl) **4**

[**Project Specification**](#_heading=h.38ca106gw7kv) **5**

[**Project Design**](#_heading=h.9170of5br6yw) **6**

[**Project Implementation**](#_heading=h.op708somm0pr) **10**

[**Results & Evaluation**](#_heading=h.en3hivmeiqo0) **14**

[**Future Work**](#_heading=h.j5jdbi3fi1) **15**

[**Conclusion**](#_heading=h.2304thi3y1rg) **1**8

[**References**](#_heading=h.grnx6nvr8knb) **21**

[**Appendices**](#_heading=h.jpl5twsr2px7) **23**

# 

# 

# Introduction

# 

In our final project, our team(ERROR [FileNotFound]) has created a Bike Track application that designed to track different types of users' daily commute and record the geometer location, accelerometer data from the user's devices while the phone is running(either in sleep or unlocked). The aim of this application is to encourage commuters to choose environmentally friendly ways to commute. Recently the environmental issue has become more serious and attracts much attention. A low carbon lifestyle is helpful to protect the earth's environment from saving good resources for the next generation. Commuting by bike or e-scooter or by foot is good for the environment and beneficial for people's physical health since nowadays most people lack exercise.

This application is targeted towards daily users who want to keep track of their daily commute speed and geometer location. The main target audience is the group of environmentalists, but our team also expects to use this application to attract more and more people to pay attention to environmental protection, hence broadly this application is targeted towards daily users who want to keep track of their daily commute speed and geometer location. It includes people who tend to walk or bicycle for their daily commute and other users who choose non-environmental friendly ways for work. We are confident that each of these parties will find it beneficial. It includes people who tend to walk or bicycle for their daily commute and other users who choose non-environmental friendly ways for work. We are confident that each of these parties will find it beneficial.

The scope for the project for us is to create an application with front-end and back-end and combine them into an Android environment application. The front-end is the most important part of our application. We have designed a pulldown list that allows users to choose which vehicle they are traveling. Next, we have a record page that stores all the pass records for users tracked, both of these pages are connected with back-end storage. Lastly, for the front-end, we allow users to change the frequency of collecting data.

To achieve the goals of our project, we have split the scope of our project into five tasks. First, we have made a small team of 4 that specially designed the prototype to produce a user-friendly interface homepage. Second, we complete the front-end design using Dart language and connect it with the back-end. Thirdly, we secure the sensor data with our database. Next, we record all the data and send it back to the Record page, which shows all the past records. Finally, we have saved the data into one text file for all records using the JSON framework to make it more readable.

Overall, our project functions normally as we expected in its scope, even when the user's device is in sleep, such that the users do not need to worry about whether the application is working. We have created as few features as possible intended to create ease of use for the user interaction for the front-end. We have used users' device sensor data to calculate the speed and location they are traveling to, so no internet is required to keep our Bike Track App running. Moreover, we allow users to change the time interval for collecting data which can help the size of the data.

# Background

At present, as the earth's environmental pollution is becoming more and more serious, more and more experts are calling for people to choose an environmentally friendly way of commuting, especially now that most people are busy with work and lack exercise. Choosing a bicycle to travel is not only beneficial to health but also beneficial to reducing carbon emissions and saving resources for the next generation. That’s the reason our team decided to create an application that records people’s commute routes and encourages people to travel more healthily.

However, there are few mobile applications designed for bicycle users on the Internet. Current applications are designed to track the user's location and speed. They are only used for fitness and fitness statistics purposes. Not used for daily commuting life. There is a mobile app called Zenly which kind of has similar features as our project. Zenly is a free smartphone application that can be used on iOS and Android phones. It allows friends and family to follow each other's trajectory and simplify their busy lives. With Zenly, you can view the location of the people you care about on a private map. But the disadvantage is that Zenly can't distinguish people’s vehicle type and doesn't record people’s routes. Zenly's positioning is different from our application, it is more like a social entertainment application, rather than a healthy life application.

The purpose of our team developing this BikeTrack application is to track the user’s commute route, distinguish the different types of vehicles used by the user; the speed and acceleration of the vehicle when the user is driving, and save all the collected data for use by the university’s research on environmental protection issues. The application we build can satisfy the Android system, access the user's smartphone sensor, and then collect the data and store it in the local database.

Our team hopes to help users realise the importance of environmental protection and increase their willingness to choose a low-carbon lifestyle. Since this is a healthier and more environmentally friendly way of working and traveling, we also help users track and record their ride data so that users can compare their changes by themselves.

Our team decided to choose Flutter as the software development kit (SDK) to build the UI framework for mobile applications. And the programming language of Flutter framework is Dart. After deep analysis with Java and Kotlin, our team noticed that Dart is more simple and powerful. Even though there are many text editors that can be used to write and build applications, Android Studio is the most suitable editor as it supports Flutter plugins. There is thermal overload in the flutter. Once the code has been changed in the code editor, after saving the changes, the UI on Android emu for iOS sim will refresh automatically. Initially, our team aimed to publish our application on two systems. But due to some problems, we can only connect to the Android system.

# Project Specification

**User requirements:**

The user should be able to choose the vehicle type they are using before tracking.

The user should be able to start and end tracking when they want.

The user should be able to get real-time tracking details feedback when tracking.

The user should be able to get an overview of the past tacking.

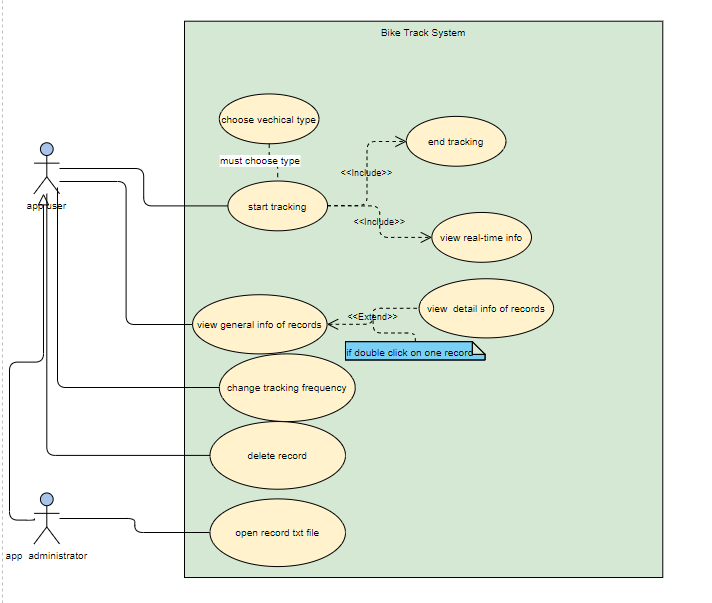
The user should be able to get the readable tracking’s details.

The user should be able to change the tracking frequency.

The user should be able to delete all past tracking records.

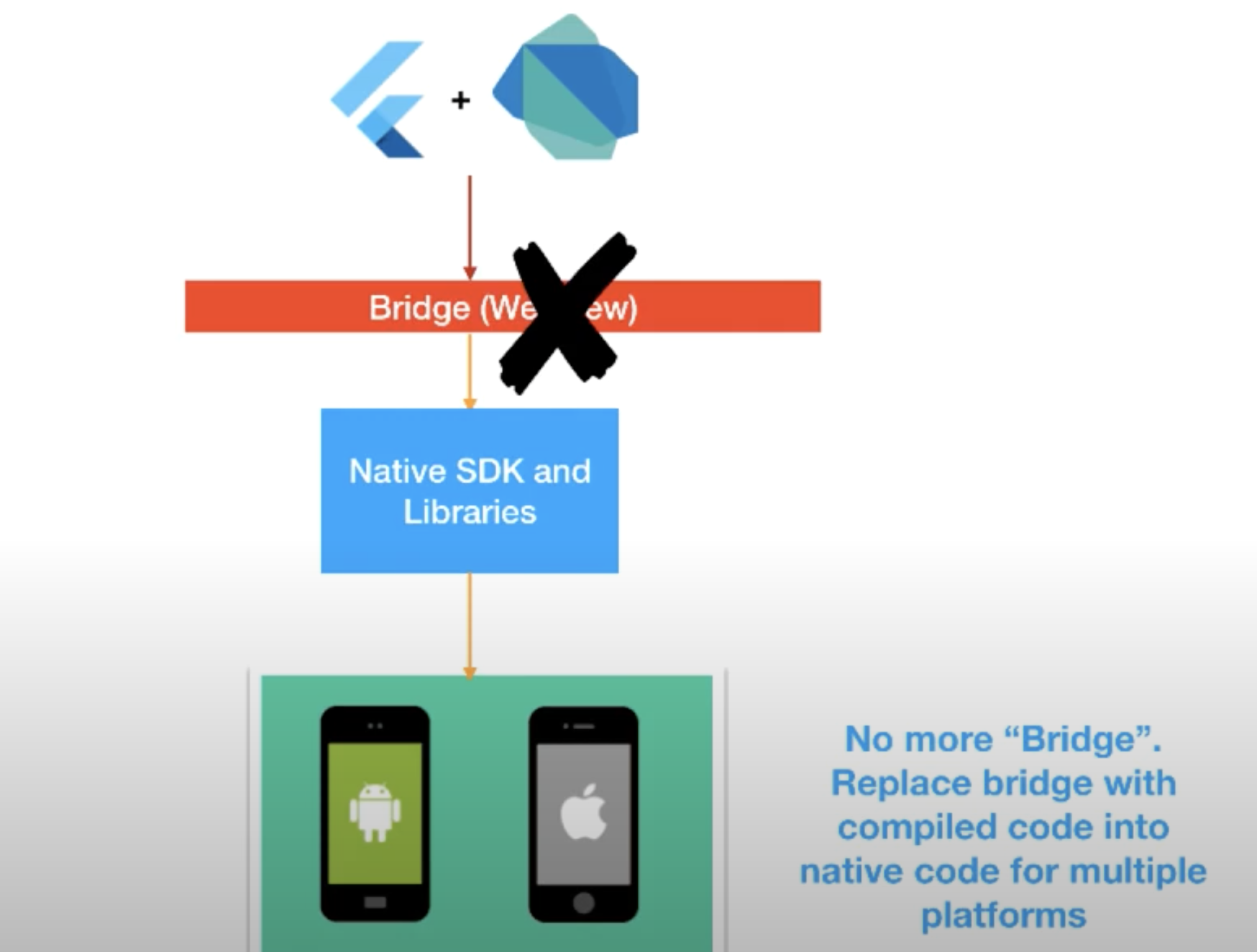
The admin user should be able to find and get the past tracking’ record txt file from the local path “/Android/data/com.example.bike\_tracking/files/data.txt”

**A use case diagram for the use case**

****

# Project Design

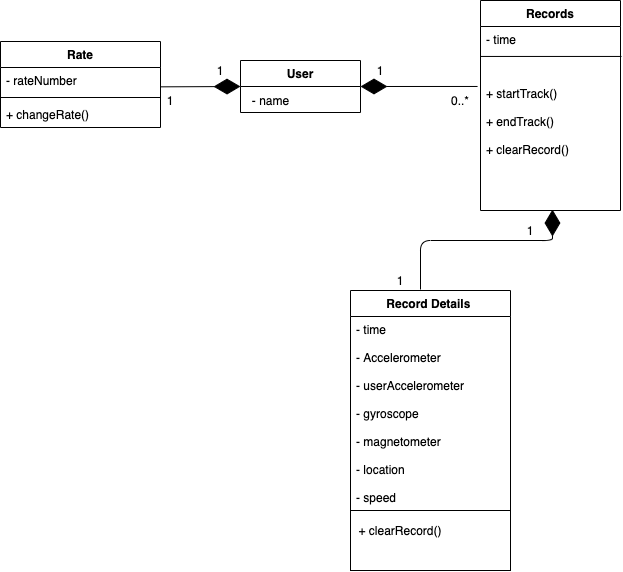
# For considering different development choices of the Bike Track application. Firstly, it is determined that the Bike Track application should be developed as a native application instead of a web application or hybrid application. Because the application needs to access the mobile sensor’s data and access local files to store data. It has huge advantages for native applications which can access the mobile sensor and read/write local files at a high speed. And it can also provide better user interaction and user experiences. After determining the native application development strategy, the Bike Track application is designed to be deployed both on Android and IOS versions. However, due to the time limitation of the development and scope of the project, the cross-platform development language is finally selected. React Native and Flutter are both powerful choices that can be used to develop an application cross-platform. But for React Native, there are some problems that are difficult to solve, such as performance problems with long lists and difficult to upgrade systems. So, Flutter and Dart are chosen as the primary programming languages for this application development. Also, Dart is the compiled code, and it makes codes closer to the native SDK and libraries.



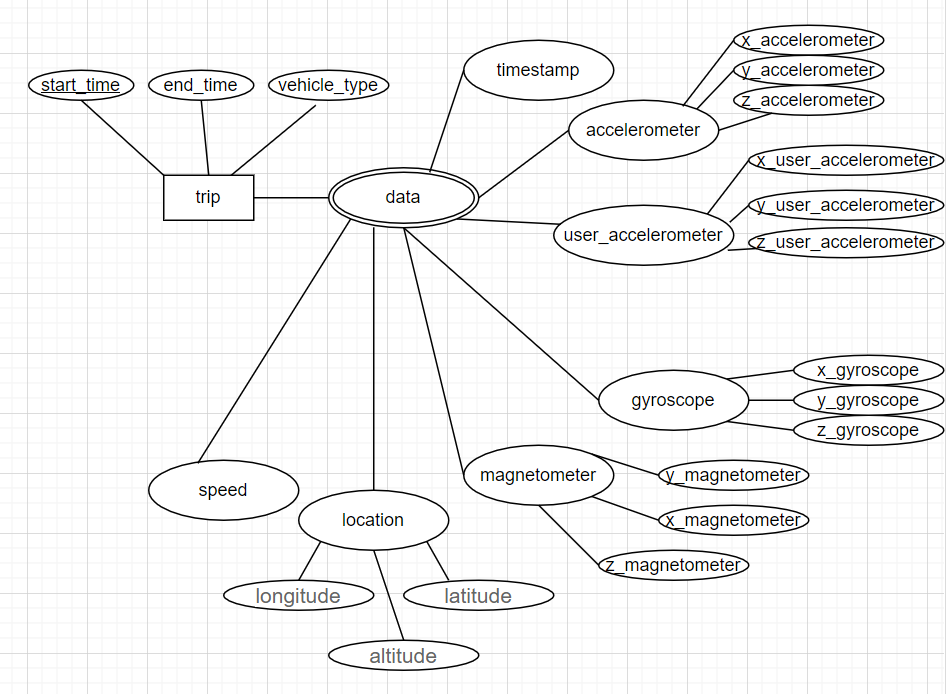
# 

# For system architectural design, by considering the cost and functional requests for this application, there is no need to deploy a remote application server and database. All speed and GPS information is designed to store in the local JSON file. The app is designed to fetch sensor data each 5 seconds initially. Users can change the data collecting rate by sliding the slider. Users can also review data records and the JSON format data will be transformed to readable format information. There are two diagrams below to show the system architecture and UML of the Bike Track application.

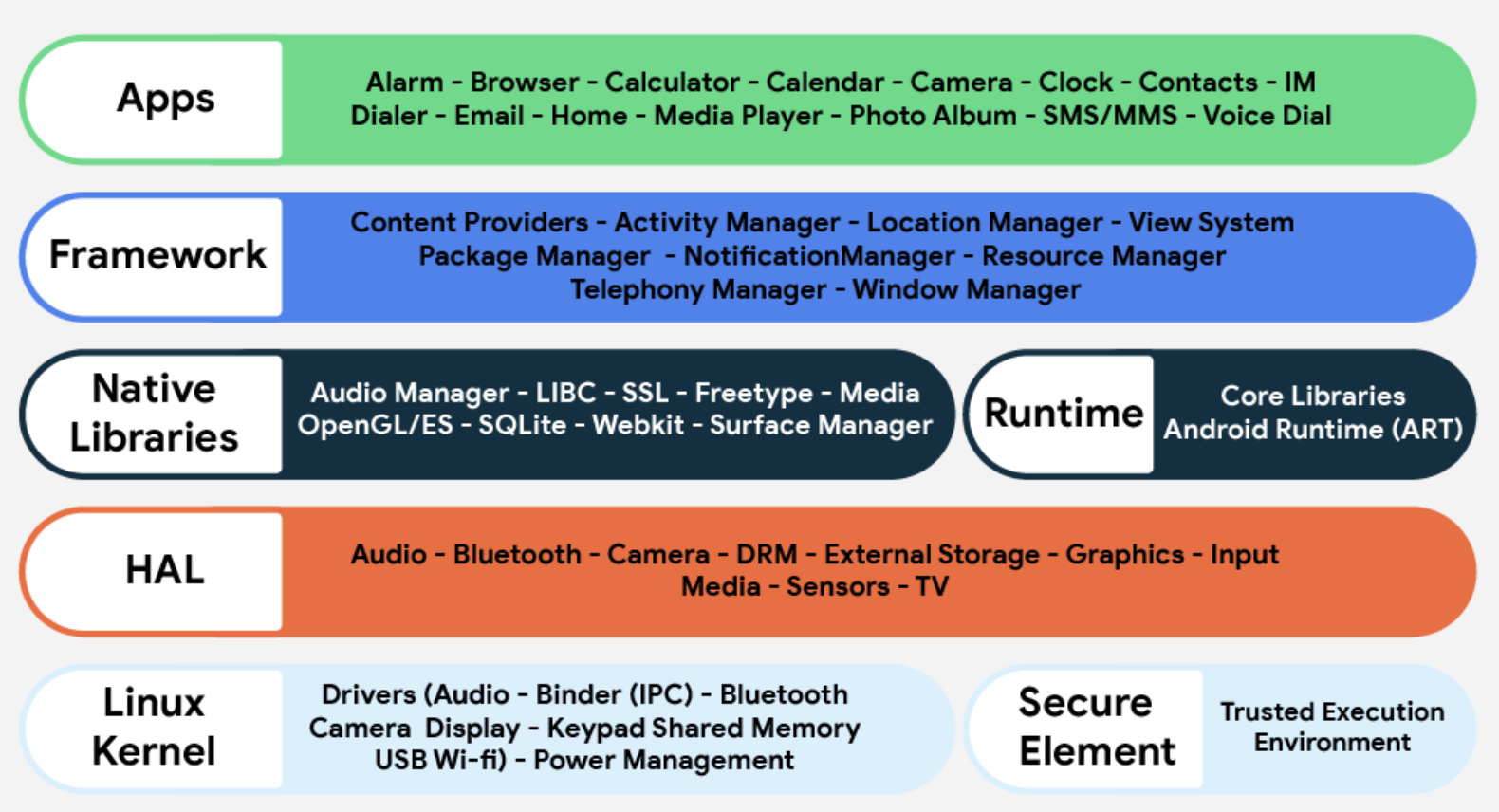
# 



The speed information should be stored as the format shown below

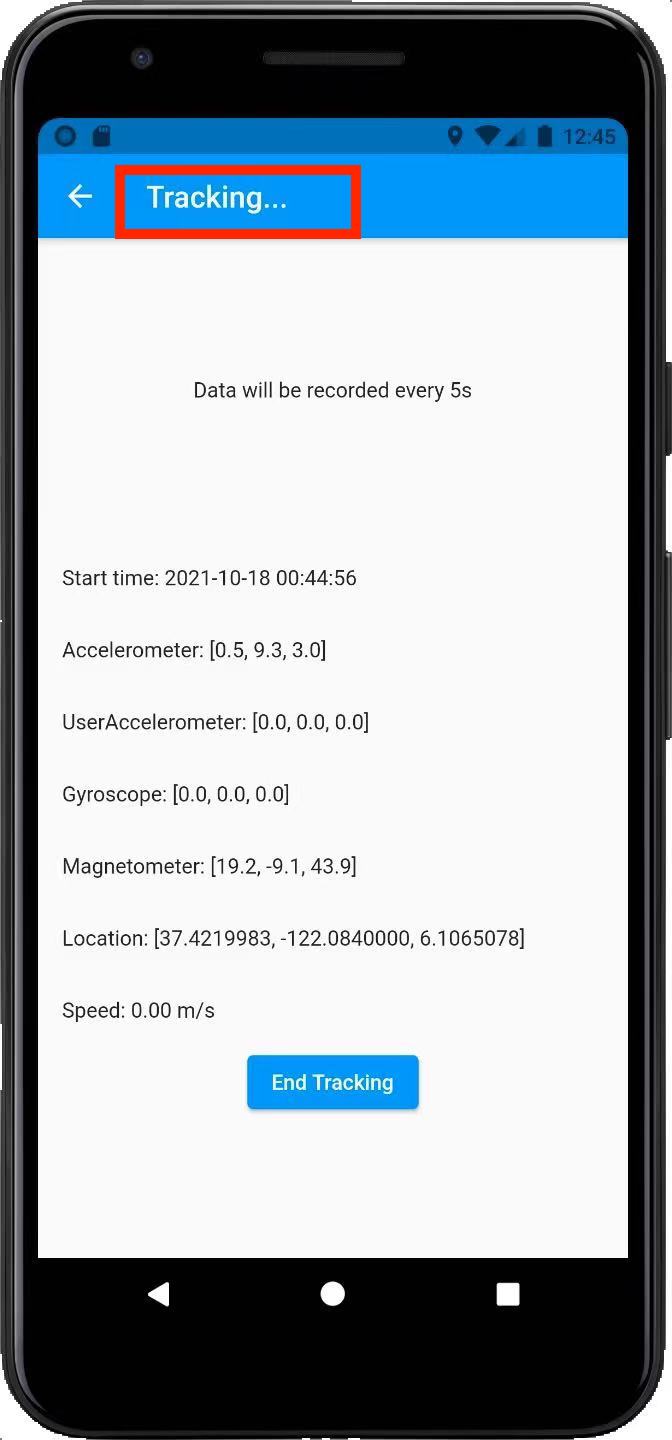
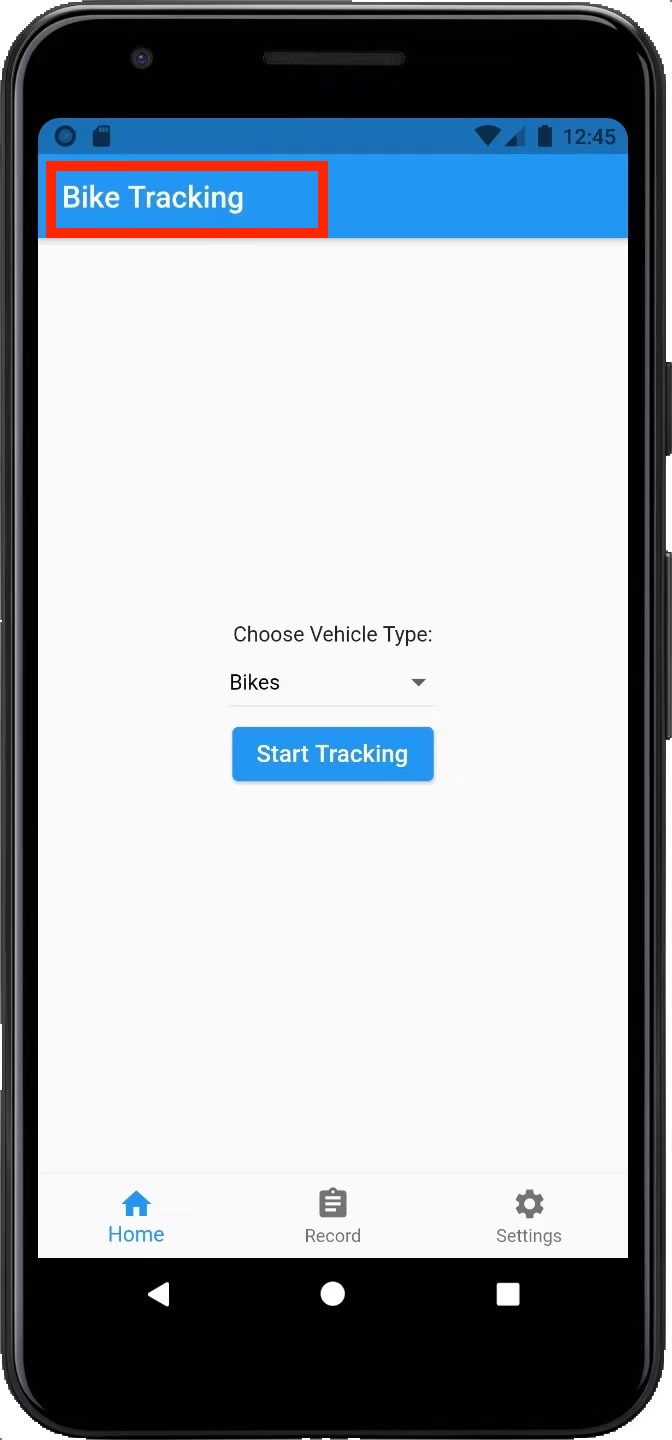


# Also, the diagram below shows the Android system architecture. It is useful to understand the logic and methods of Android programming more clearly, to have overall control over Android development.



# 

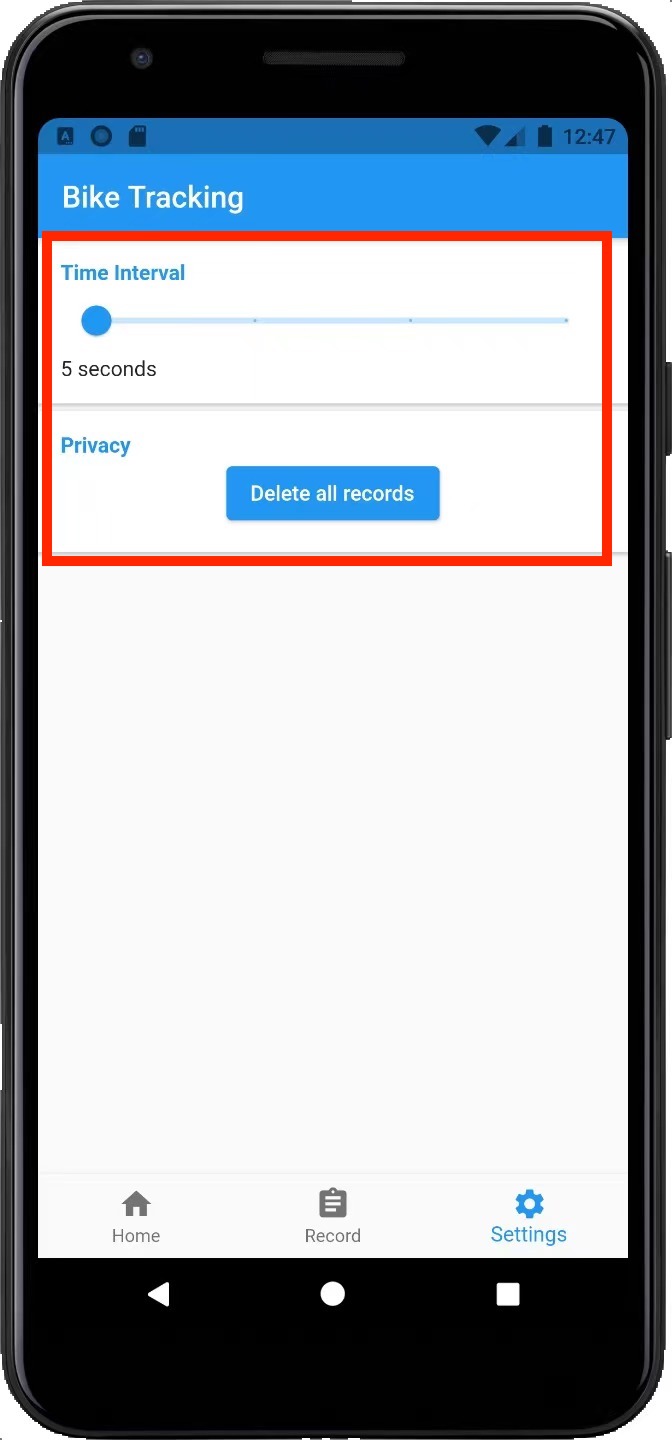
# For user interface, it is necessary to align the UI design with design principles for mobile. And one of the principles states that an application should always provide feedback for a user’s actions. As the screenshot shows below, when users click the “Start Track” button, it will move to the tracking page instantly and users can view the tracking information in real time. Also, the title of the app bar will change from “Bike Tracking” to “Tracking…”. It provides feedback for users and helps users understand the application is really tracking now.



# 

# 

# Another professional design of the user interface is the application always follows the user in control principle. As screenshots are shown below, users can always choose to stop tracking, exit the tracking page, clear records, or change the data collecting rate. And it only needs a few steps for the user to achieve these features instead of running a long process.

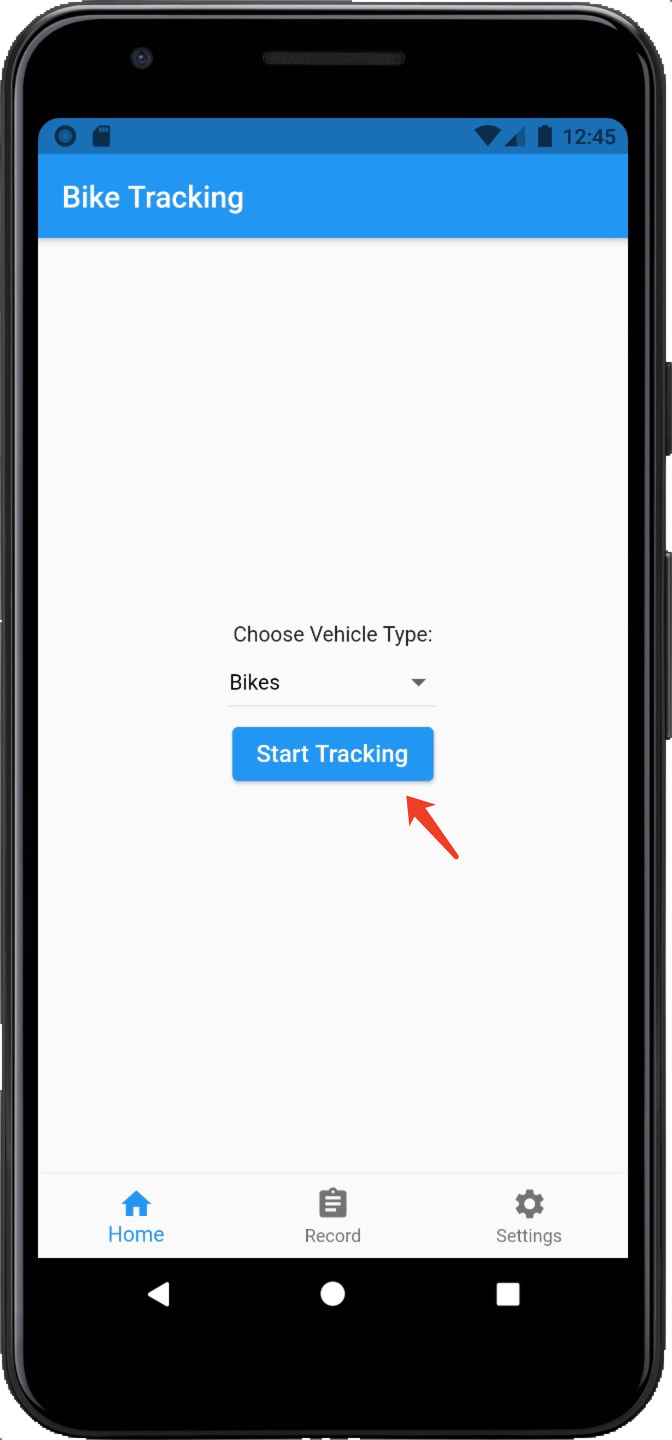
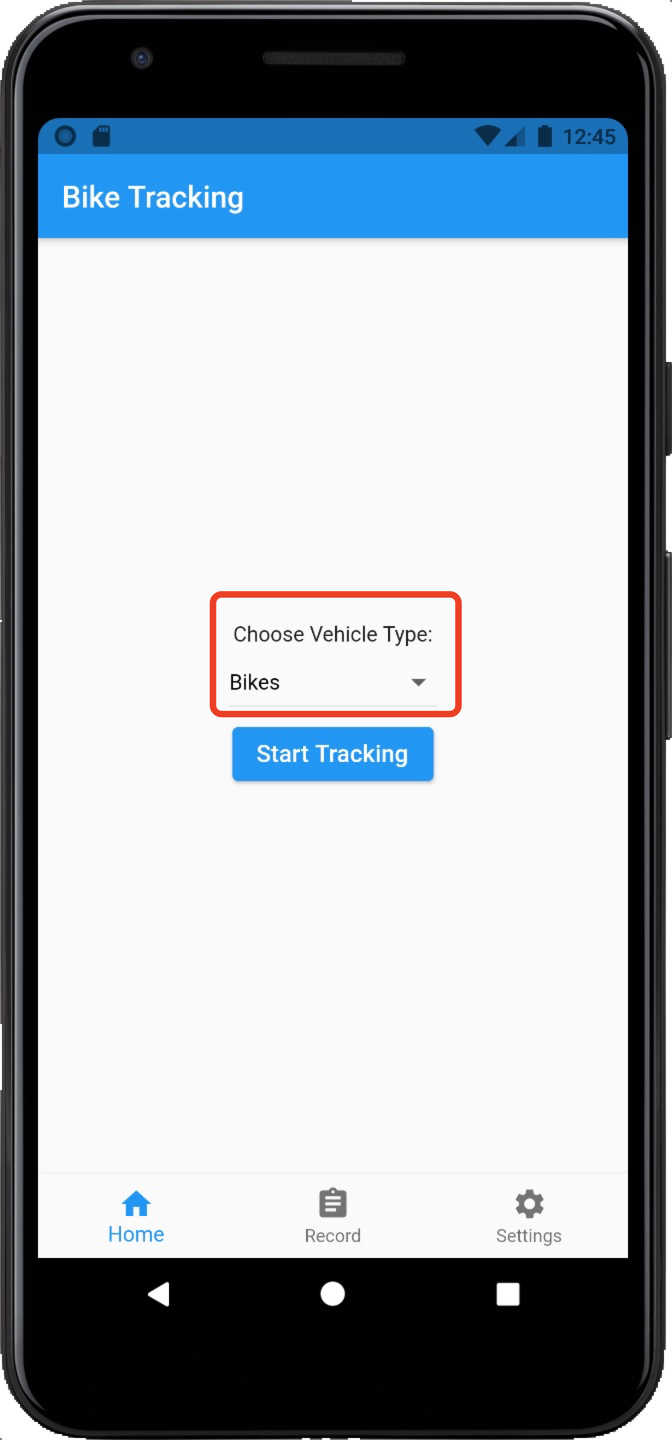
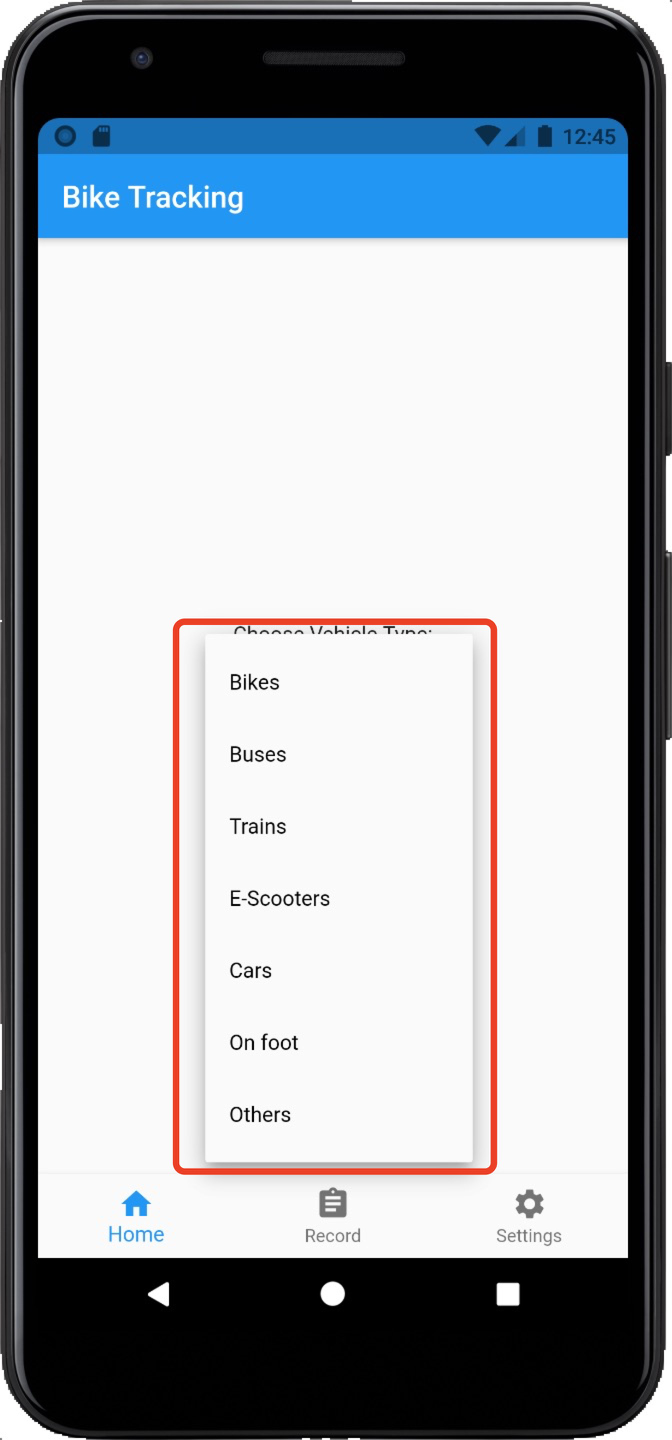


# Project Implementation

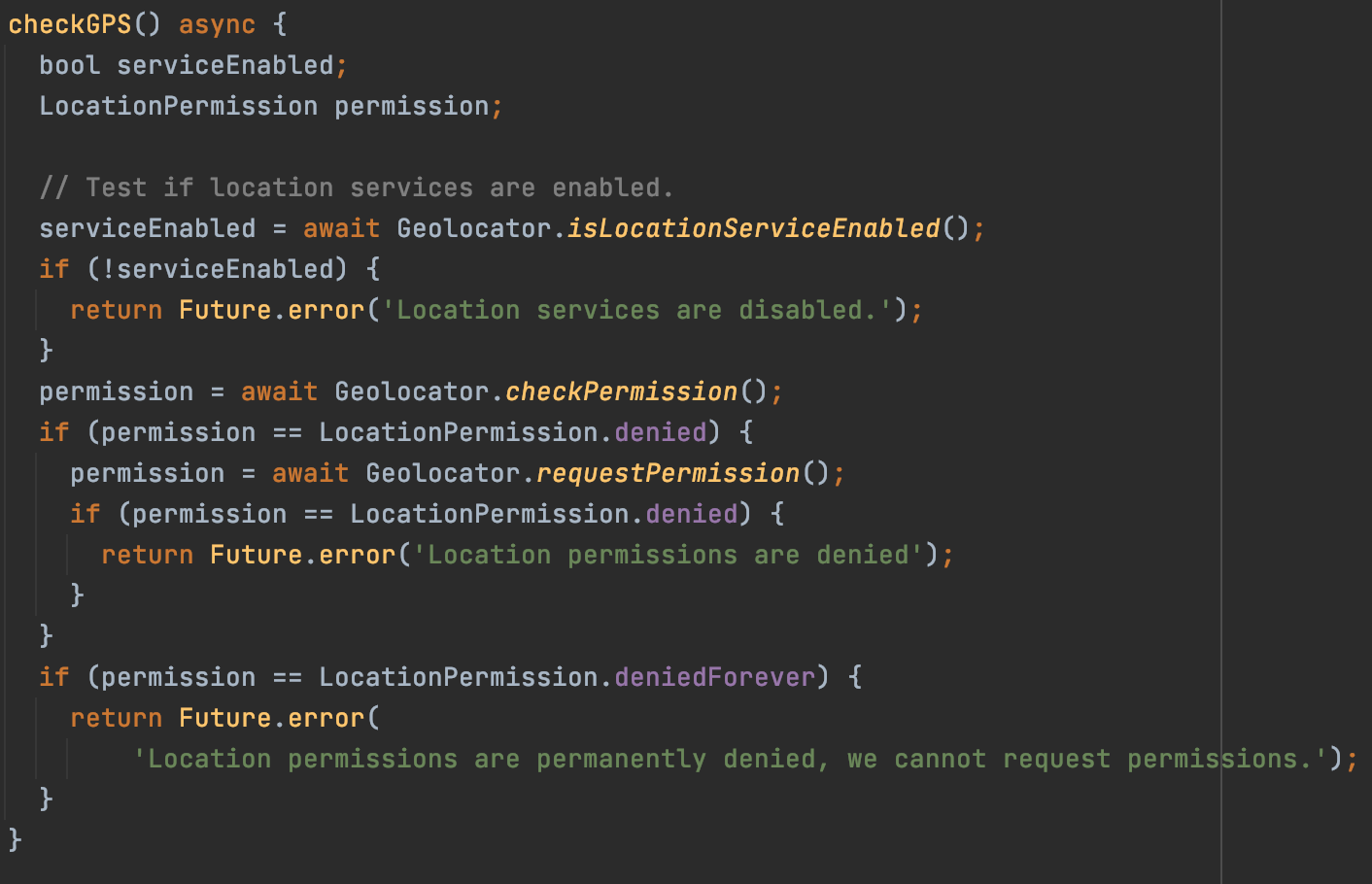
# 

Our app has four main pages for three different uses.

Start Tracking Page is also the default page of the app. This page contains a dropdown list that holds the vehicle types within it. The pulldown list can keep the vehicle types chosen by the users, and this information will be stored in the file after start tracking. There is a start tracking button on this page as well. Users will be navigated to the tracking page after clicking this button.



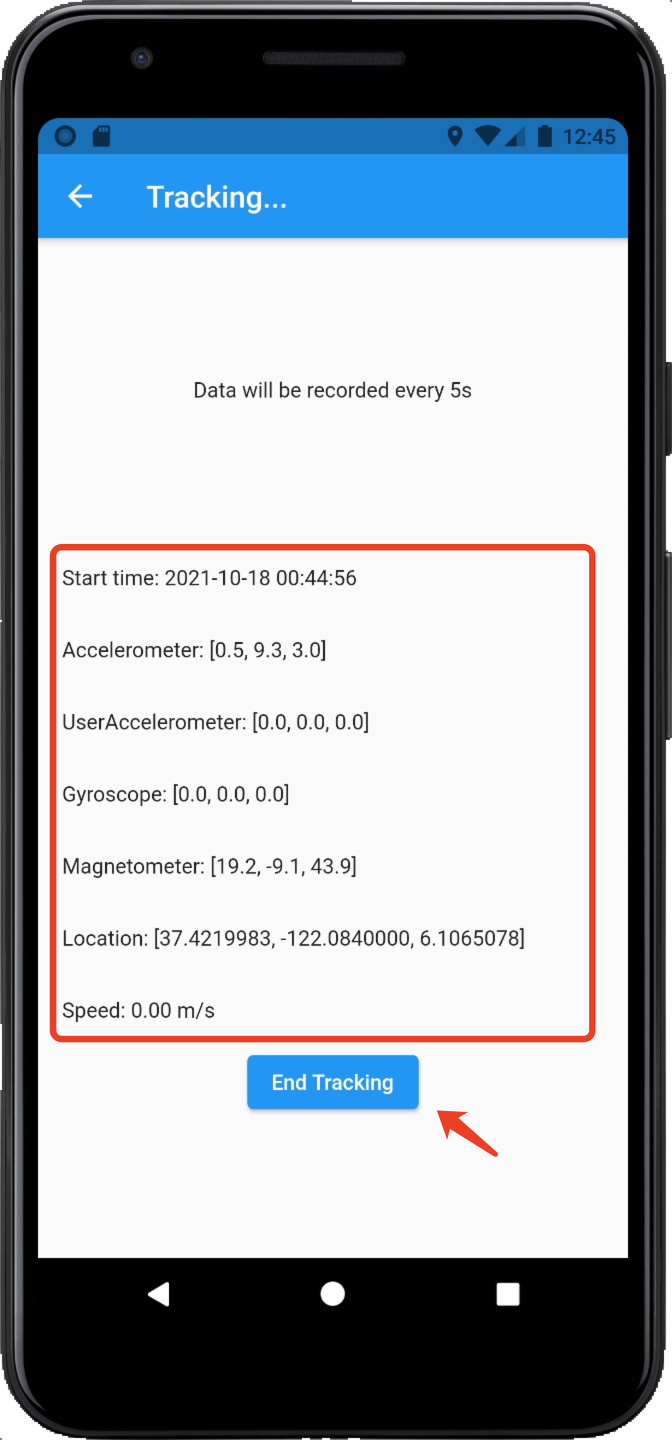
The tracking page contains real-time GPS information and sensor data. Geolocator package is involved in getting the current location of the device (baseflow.com, 2021). Also, the location service must be accessible to ensure this function can work properly. The availability of the location service will be checked first. If the location service is not accessible, the Geolocator will request permission. If the permission is denied, the error message will be printed. At this stage, our app cannot provide any tracking services. Sensors\_plus package is used to fetch the sensor data, including accelerometer, gyroscope, and magnetom (fluttercommunity.dev, 2021)



The collected data will be stored in the file automatically. Build\_runner was used to generate the files. Path\_provider was used to set the path of the generated files. Json\_seriablizable was used to convert the data into JSON format. (google.dev, 2021)

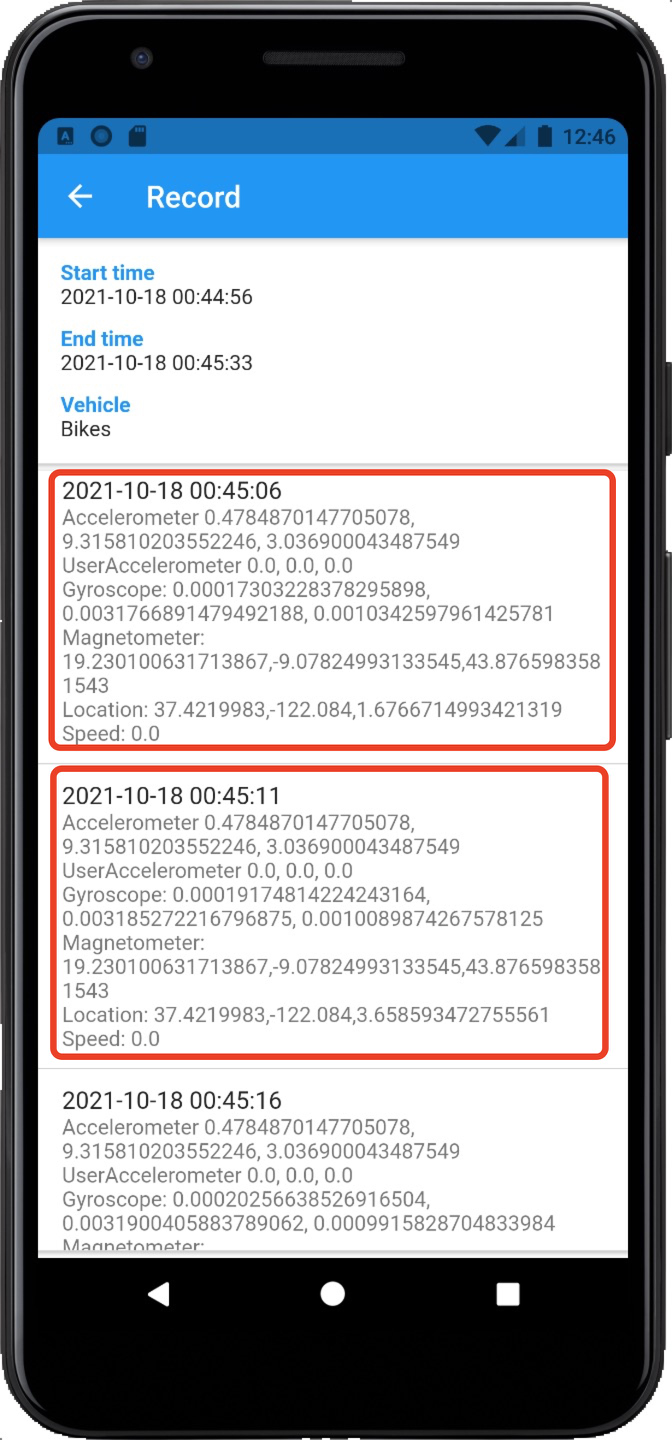
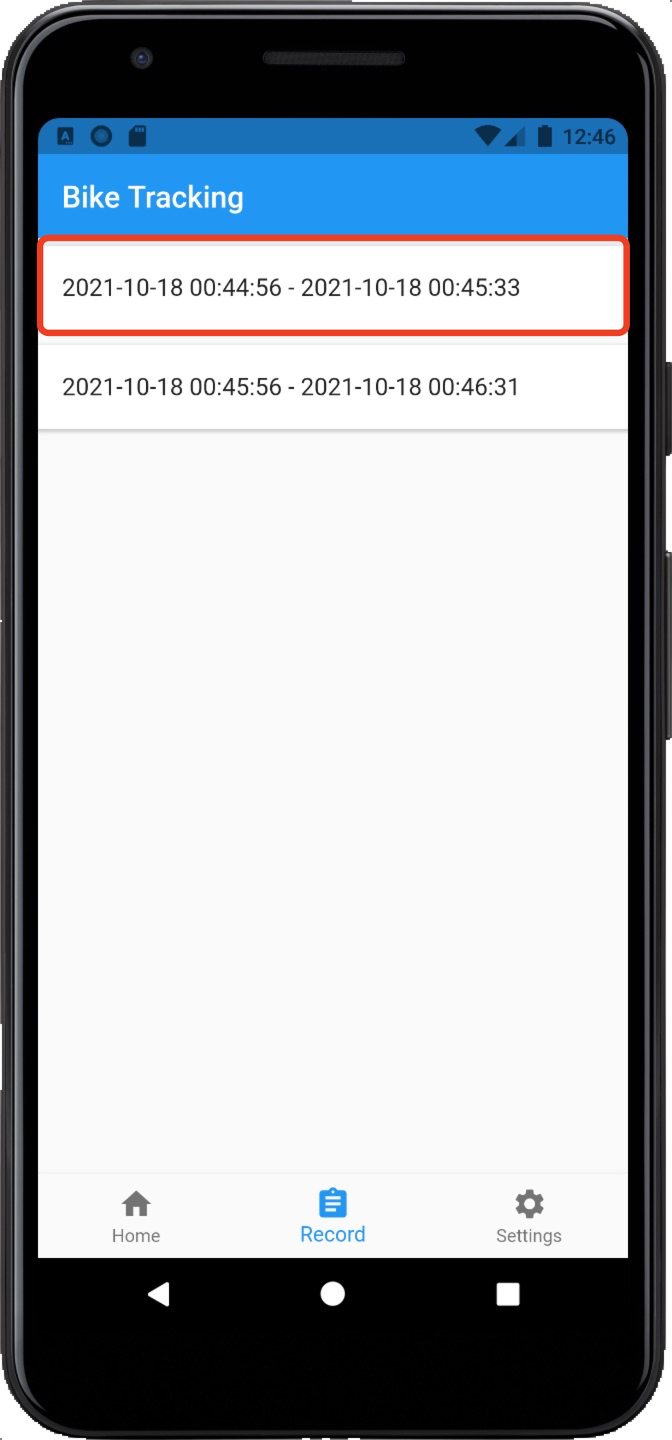
When the user clicks the "Start tracking" button, the UI jump to the "TrackingPage" where we use "StreamSubscription" to listen to the "gyroscopeEvents" "userAccelerometerEvents" "magnetometerEvents" through "sensors\_plus" also the "position.latitude" "position.longitude" "position.altitude" "position.speed" from "geolocator". Besides, set the value of the above variable on the UI so that the UI can show the real-time details. In addition, we use "Timer.periodic" to get the sensor data periodically with the frequency the user set before and store them in a list.

If the user clicks the "End Tracking" button, we will dispose of the existing "StreamSubscription" to stop tracking. And store the current data and trip information in the correct local place by "path\_provider". (flutter.dev, 2021)

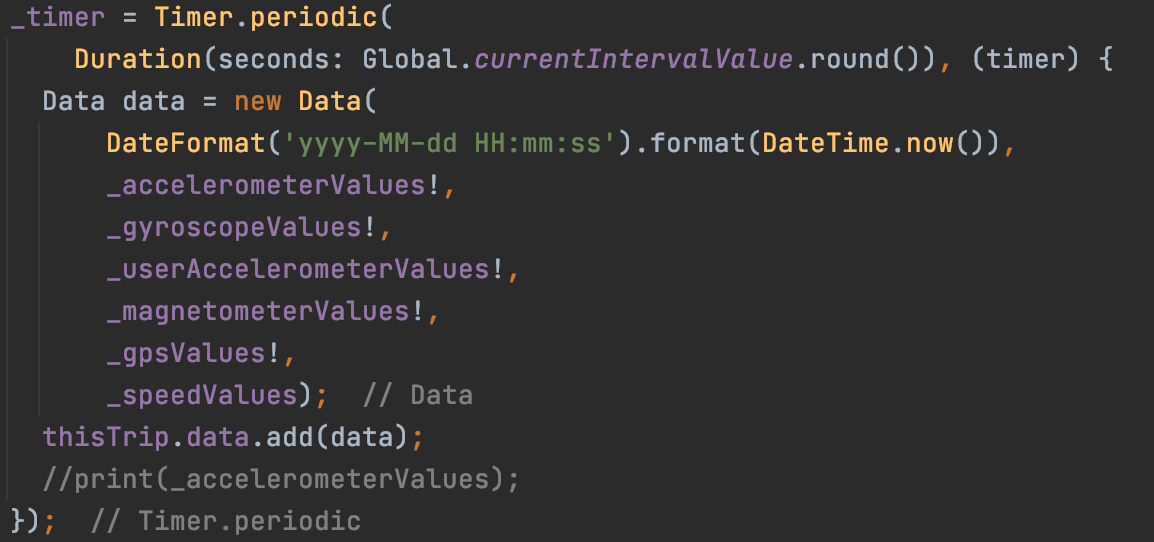


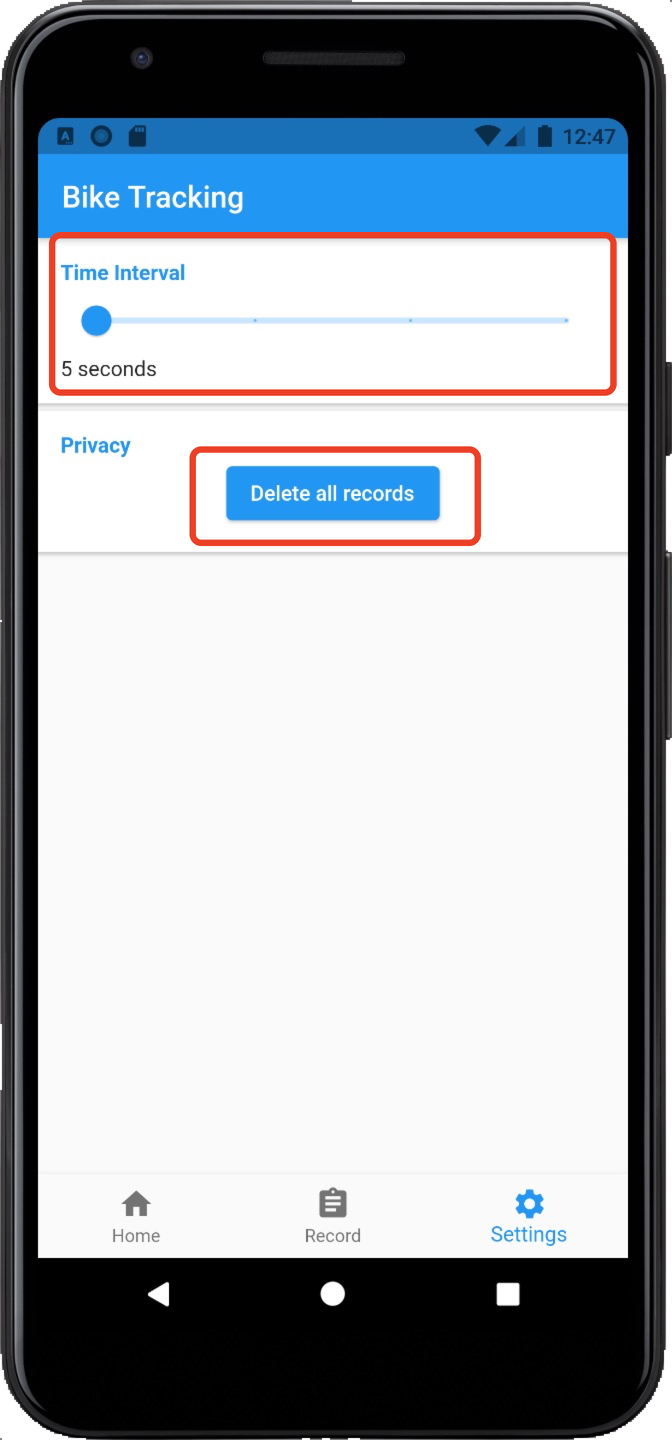
The record page consists of a scrollable ListView with ListTitle inside that shows all the recorded trips on the phone. The title is starting time and ending time which is accessed with the use of the intl package. The records will be listed according to generated time. Users will be navigated to the corresponding trip page by clicking each piece of history. More details will be available on the trip page, including GPS data and sensor data fetched between the interval set by the user on the setting page. The trip page also contains the start time, end time, vehicle time, and data recorded time.





The setting page contains a slider to change the data collecting rate for tracing and recording. The default interval value is 5. This value will be held by Global.currentIntervalValue and will be used to set the timer in tracking\_page.dart to change the duration for data collecting. A clear all data function is on this page as well. If the data file exists, the file will be deleted and the records on the record page will be cleared simultaneously. If not, an error message will be shown.





We also apply asynchronous functions to our code to reduce latency.

**unforeseen problems that we encountered during implementation:**

For getting GPS data from the sensor, we decided to use the “location” library first, but that is an ancient and challenging library to use. If we want to accomplish it, we need a significant workload on configuration. We decided to find other libraries that are more straightforward. Fortunately, we found a “geolocator” library that needs to set the service permission. This function was achieved soon.

Permission management problem, our app needs the permission of storage, location, and GPS. In the beginning, we intended to use “simple\_permission” to access the location service, but some errors occurred. The problem might be the latest Android version does not support simple\_permission. Then we went through a lot of online resources and then found another method from StackOverflow that was to use “permission\_handler” to avoid the previous problem. Permission\_handler can work properly without any problem accessing location permissions. Then we added the checkPermission and requestPermission to our code to accomplish the use of the built-in method of “geolocator”. Moreover, we tested the app under several Android versions to ensure the app could operate well in different situations.

**Results & Evaluation**

The primary goal for our project was to design and deliver a mobile application that uses inbuilt sensors to record and store the commute data of users. The preliminary data set to be recorded during the planning phase was GPS Data and Accelerometer Data. The team achieved all the project goals to their fullest extent. We successfully collected all necessary GPS and accelerometer data, which included, Latitude, Longitude, Altitude, Accelerometer, Gyroscope, Magnetometer and Speed readings. The next goal for our project was to store the data locally, which we also successfully achieved. The project stored the data in local files, achieving the functionality of not needing internet access. The only goal that was not achieved fully was the need for an Android & iOS implementation. Even Though we used a framework that allowed for the same codebase to be used for both operating systems, during the project development phase, we faced a few problems with the iOS side of the development, which caused it to be unfinished. On the contrary, every other project goal was fully achieved.

The system was tested first on the inbuilt simulator in our development environment. We used the virtual environment motion testing features to test if the accelerometer readings were accurate. We were also able to test the GPS sensors through a feature on the IDE, which allowed us to change the virtual location of our device. These sensor readings were confirmed to work through the virtual features and were also tested on an actual device. The plan from the start for testing was to finish implementation first and then run tests, both as we develop and finalise.

The strength of our project was the depth of information stored; the app can record and store 5+ different types of sensor information that can be stored at a time frame that the user can choose. This means that the user can choose based on the type of commute they are recording and the timeframe they would like the data to be stored in. The main weakness of our project is the inability to fully function on the iOS version due to the restrictions on local data access. Another weakness of the app was possibly the lack of choice of where the data is stored. The user may want to have access to where the data is stored, and as of now, there is a set location.

In conclusion, the app met all the aspects of a minimum viable product (based on what the client expected) and delivered more. The project faced some problems working on the iOS operating system but fully functioning on the android side. The app successfully recorded and stored sensor information based on the user's timeframe preference.

# 

# 

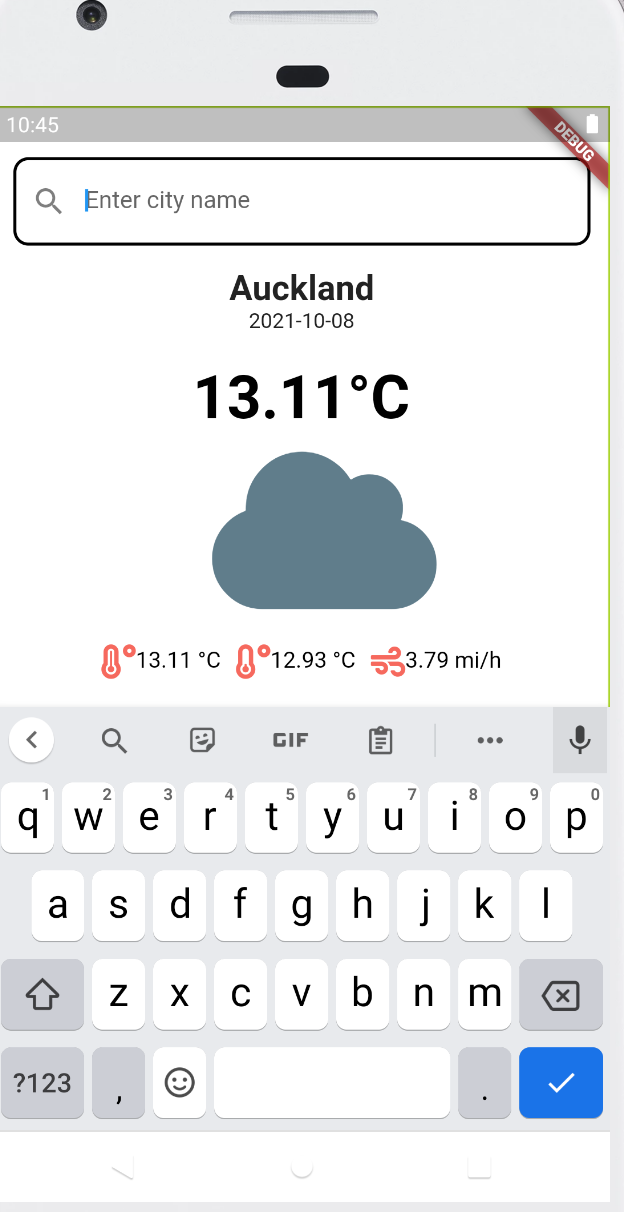
# Future Work

# Firstly, we were thinking of adding Google Map API into the user interface to track their accurate locations. As the application was supposed to run without the internet, the properties of Google Map API cannot work offline. So we removed that part in order to suit the current requirements. Besides, the climate forecaster section was removed since it depended on the internet as well. Plus, in the previous stage, we decided to use Flutter to build the application because only one language was needed, and Flutter could generate the applications that suit both Android and IOS separately. However, we found it was hard to build the IOS version application because of the closed source systems. It was hard to store, fetch and modify those data. So finally, we only had the Android version done with all functionalities. We made a dropdown bar to allow users to choose the vehicle type they would travel in instead of analyzing the data using machine learning. But this part can be replaced with the machine learning analysis for further implementation later. In addition, the font-size picker and theme color picker were added to the user interface to improve the user experience. After that, we figured out these functions were not topic-related, so we decided to delete them to make our application briefer and more straightforward.

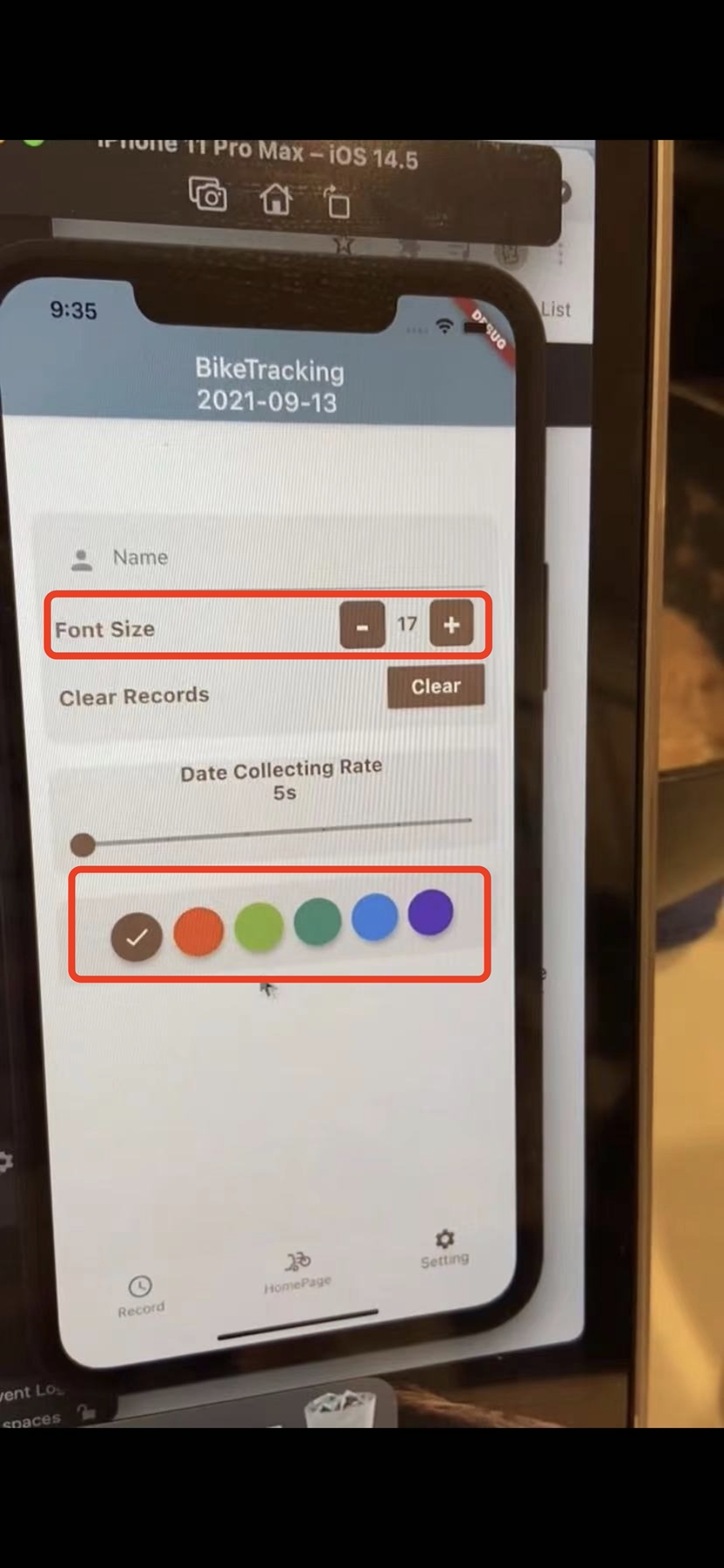
# 

# Our application can be improved by some internet-related functionalities in the future, such as Google Map API and climate forecaster section as mentioned above. Google Map API can show users the exact locations they are currently in, and the user can get the navigation service from the app as well. The climate forecaster part has been completed already and can be added to the application as long as the application is allowed to get connected with the internet. Furthermore, a user login section should be involved, and the data can be stored under different users’ accounts separately. Moreover, as mentioned above, the dropdown list of vehicle types can be removed. Only the accelerator and geometer will be stored in the plain text file and be replaced with data analyzing methods. For example, using machine learning to manipulate the stored data. Give the computer a significant amount of collected data, training the computer to distinguish different vehicle types according to the collected data during daily commuting. And also can instruct the machine to figure out if the user is traveling to work or is going back home. Also, the font size changer and the theme color picker can be added back to the app to improve the user experience.

# 



Weather forecaster



Font changer and color picker

# 

# 

# 

# 

# 

# 

# Conclusion

As we mentioned in the introduction, we aim to build an application that can run under Android and/or IOS systems **before 28/10/2021**. This application should allow us to track users' daily commute and record the geometer location data and accelerometer data through the user's device and store those data locally. This application should also automatically distinguish the different vehicle types, such as e-scooter, motorbike, car, and bike.

The first finding is, we did not achieve the function that allows our application to distinguish different vehicle types by itself automatically, but it was replaced by a dropdown bar on the homepage. Every time the user starts their route, they will be asked to choose the vehicle type that they are using to travel. As it involves many machine learning algorithms, the basic principle of machine learning algorithms is to use the historical data as input to predict new output values. To achieve this, we need to conduct many practical tests/experiments outdoors. We used to plan to do the machine learning section at the beginning of the semester. However, in reality, due to the Level four lockdown in Auckland, the government does not recommend that we go out for unnecessary events.

Another key finding while making this application is that using Java as our SDK will be much easier than using flutter. Because Dart is a new programming language for all of us, none of us used this language before. We decided to build both versions at the very first beginning and flutter, because flutter can generate the application which suits both the IOS system and Android system. But when we were doing the actual building process, we found that as the IOS system is a closed ecosystem, we cannot give access to the source code, so we can not see or modify it in any way. So, if we choose another SDK first, it would be much easier for us to build this project.

So far, our application has achieved most of the functions that we expected at the beginning. We have a user-friendly interface homepage, and it also contains a record page, and the setting page and those meet the visibility of the system status feature. Users can achieve the essential function of our application on those three pages. Users can start or end their route tracking on the home page, and It can begin recording users' commute or end up recording commute as the user wishes. They can also choose the vehicle type they are using to travel after they click the start button. This design shows the user controls and freedom feature. The users can also change the rate that the data generated from the sensor on the setting pages through swiping the data generated rate button, which shows the flexibility efficiency of the use feature. Our app can also generate the accelerometer, magnetometer, and geometer data from users' devices and record all the data. The application also has a record page that allows our user to see all the past records from this page and the format that we use to store the data is JSON format, and it is easy to read and understand. As the client is required to retrieve the data that we store, we stored the data locally. The file that we store can be found in the filesystem, in which the user can obtain them directly.

Overall, most of our projects' functions worked properly as we expected, even some unnecessary functions that I mentioned above did not achieve. However, after we communicated with our client, he said those would not be a problem. So, I can say our application is quite successful in general.

# 

# 

# References

# 

# 

baseflow.com. (2021, Oct 22). *geolocator 7.7.1*. geolocator | Flutter Package. https://pub.dev/packages/geolocator

flutter community.dev. (2021, Oct 5). *sensors\_plus 1.2.1*. sensors\_plus | Flutter Package. https://pub.dev/packages/sensors\_plus

flutter.dev. (2021, Sep 17). *path\_provider 2.0.5*. path\_provider | Flutter Package. https://pub.dev/packages/path\_provider

google.dev. (2021, Oct 22). *json\_serializable 6.0.1*. json\_serializable | Dart Package. https://pub.dev/packages/json\_serializable

# 

# Declaration of Authorship

# 

# 

# 

| **Section** | **Author(s)** | **Signature(s)** |
| --- | --- | --- |
| **Title Page/Logo** | **YASH SOOD** | **YS** |
| **Executive Summary** | **Youlun Li / Aaron Fan** | **YL/AF** |
| **Introduction** | **Aaron Fan** | **AF** |
| **Background** | **Sura Liu** | **SL** |
| **Project specification** | **Yuyuan Wu** | **YW** |
| **Project design** | **Youlun Li** | **YL** |
| **Project implementation** | **Yuyuan Wu / Shan Li** | **YW / SL** |
| **Result & Evaluation** | **YASH SOOD** | **YS** |
| **Future Work** | **Shan Li** | **SL** |
| **Conclusion** | **Zixuan Su** | **ZS** |

# 

# Appendices

| **A. Gantt Chart**    **B. Link to demo video/website**  [**https://drive.google.com/file/d/1ECnVEs\_LNGpBPVDFl4N0EtnEPikawcb1/view?usp=sharing**](https://drive.google.com/file/d/1ECnVEs_LNGpBPVDFl4N0EtnEPikawcb1/view?usp=sharing)  [**https://drive.google.com/file/d/1WiXIma8cem5nJhywxU\_InKvsHoM1RLPL/view**](https://drive.google.com/file/d/1WiXIma8cem5nJhywxU_InKvsHoM1RLPL/view)  **C.Link to final Build**  [**https://github.com/uoa-compsci399-s2-2021/Team8-Bike\_Track**](https://github.com/uoa-compsci399-s2-2021/Team8-Bike_Track) |
| --- |