Bachelor Thesis Project (BTP) Report

Vivek Kaushik

Roll Number: 1903140

Month of Submission: April 2023

Supervisor: Dr. Sharad Sinha, Computer Science and Engineering

This report is submitted towards partial fulfillment of the requirements for the award of the Bachelor of Technology (B.Tech) degree in Computer Science and Engineering at the Indian Institute of Technology Goa.

Introduction

- **Problem Statement**

  In today's fast-paced world, efficient transportation is crucial for the smooth functioning of daily activities, especially within large campuses like universities and corporate offices. The existing transportation management systems often face issues such as long waiting times, limited information about vehicle arrival, and a lack of understanding of available transportation options on specific routes. The inconvenience caused by these issues can lead to dissatisfaction among students, faculty, and staff, and result in them seeking alternative, less sustainable modes of transportation. The main problem this project aims to address is the development of an effective and user-friendly transportation management system that allows users to track vehicle locations, access information about nearby bus stops, and receive updates on waiting times.

- **Background and Purpose**

  Given the problem statement, the primary purpose of this project is to develop a comprehensive and user-friendly transportation management system that addresses the challenges faced by large campus environments. By leveraging modern technologies and frameworks, the proposed solution aims to provide real-time data on vehicle locations, nearby bus stops, and waiting times, enabling users to make informed decisions about their transportation choices. This will ultimately contribute to a more sustainable and well-managed transport system.

- **Objective**

  The main objectives of this project encompass several key aspects:

  1. Develop a user-friendly mobile application for riders that provides real-time information on vehicle locations, nearby bus stops, and waiting times.
  2. Create a separate mobile application for drivers, allowing them to update their location and access relevant information to improve route planning and communication with riders.
  3. Integrate a centralized administration system for managing users, vehicles, and other essential aspects of the transportation system.
  4. Utilize modern development tools, frameworks, and techniques to ensure a high-quality, scalable, and maintainable solution.

Updated Features and Functionality

- **Replacement of Google Maps API with 'flutter_map' Package**

  The project has replaced the Google Maps API with the 'flutter_map' package, which offers a more streamlined map integration experience. This change has led to a more lightweight solution, reducing the load on devices and providing better performance for users when interacting with the map.

- **Revamped Rider App Features**

  Several new features have been added to the Rider app, enhancing its functionality and user experience:

  1. Custom markers for landmarks: Riders can now see custom markers on the map, representing various landmarks and points of interest. This feature helps users easily identify and locate specific landmarks within the app.
  2. Centering the map on the user's location: The app now provides a button that allows users to center the map on their current location, making it easier to find nearby bus stops, routes, and landmarks.
  3. Toggling the visibility of landmark lists: Users can now toggle the visibility of the list of landmarks, enabling them to customize the map view according to their preferences and focus on specific points of interest.

- **Library Updates and Deprecated Library Replacements**

  To enhance the app's performance, stability, and compatibility with newer devices and operating systems, deprecated libraries have been updated or replaced with suitable alternatives. This ensures that the app remains up-to-date and continues to provide a seamless experience for users across various platforms.

- **Numbered Map Markers**

  Each marker on the map has been assigned a unique number, making it easier for users to identify and reference specific locations, landmarks, and stops within the app. This feature simplifies communication between users and the transportation system, enabling riders to quickly and accurately convey their desired destinations or pickup locations.

Framework/Tools used in the project

- **Flutter Framework and Dart Language**

  The Flutter framework, developed by Google, is a UI toolkit that allows developers to create natively compiled applications for mobile, web, and desktop platforms from a single codebase. In this project, I utilized Flutter to build a cross-platform app that runs on both Android and iOS devices. The framework provided me with a comprehensive library of widgets and tools, simplifying the app development process and enabling me to create a polished and user-friendly application. The programming language used in conjunction with Flutter is Dart, a language that combines the best aspects of several modern languages while maintaining a familiar syntax. Dart allowed me to develop an efficient and robust codebase for the app, ensuring optimal performance and maintainability.

- **Firebase Realtime Database**

  Firebase Realtime Database is a cloud-hosted, NoSQL database that allows for the storage and synchronization of data in real-time. In this project, I integrated Firebase Realtime Database to manage data related to user and driver locations, bus stops, and waiting times. Firebase provided me with the ability to quickly and easily create and manage database entries, ensuring a seamless experience for users as they interact with the app. Additionally, the real-time functionality of Firebase allowed the app to respond immediately to changes in data, such as driver location updates or changes in waiting times, providing users with accurate and up-to-date information.

- **Flutter_map Package**

  The Flutter_map package is an open-source mapping solution for Flutter applications. It utilizes the powerful Leaflet.js library to provide a customizable and user-friendly map interface. In this project, I replaced the Google Maps API with the Flutter_map package, resulting in a more streamlined map integration and a more lightweight solution. The package allowed me to display maps with custom markers, calculate polyline routes, and implement various user interaction features, such as zooming and panning.

- **Geolocation and Geocoding Services**

  Geolocation and geocoding services play a crucial role in the functionality of the app, as they enable the app to determine and display the user's current location, as well as provide information about nearby landmarks and bus stops. In this project, I utilized geolocation packages like Geolocator to detect the user's location and monitor location changes in real-time. Similarly, I employed geocoding services to convert the user's coordinates into a human-readable address, allowing users to easily identify their location on the map. These tools and techniques were essential in developing an app that provides accurate, real-time location information to its users, enhancing the overall user experience.

Why flutter_map over google_map?

While Google Maps is a popular and widely used mapping service, it has certain disadvantages when compared to the Flutter Map, particularly in the context of a Flutter application. Some of the disadvantages include:

1. **Cost**: Google Maps can become expensive, particularly for applications with a large number of users or a high volume of map-related requests. The Google Maps API has a pricing model that charges for various features and usage levels, whereas the Flutter Map package is free and open-source.

2. **Platform-specific implementation**: The Google Maps API requires separate implementations for Android and iOS, which can lead to additional development time and effort. On the other hand, the Flutter Map package is built for cross-platform compatibility, making it easier to maintain and update for both platforms simultaneously.

3. **Customization**: Although Google Maps offers a wide range of features, it may not provide as much flexibility for customization as the Flutter Map package. Flutter Map allows developers to use different map providers, customize the appearance of the map, and easily add custom markers and other map elements, providing more control over the map's look and feel.

4. **Open-source and community-driven**: The Flutter Map package is open-source, allowing developers to access the source code and contribute to its development. This can lead to a more diverse range of features and improvements over time, whereas Google Maps is a proprietary service controlled by Google.

While Google Maps is a powerful and reliable mapping solution, its disadvantages compared to the Flutter Map package make the latter a more suitable choice for certain Flutter applications, particularly those that require a high level of customization or have cost constraints.

App Architecture and Design

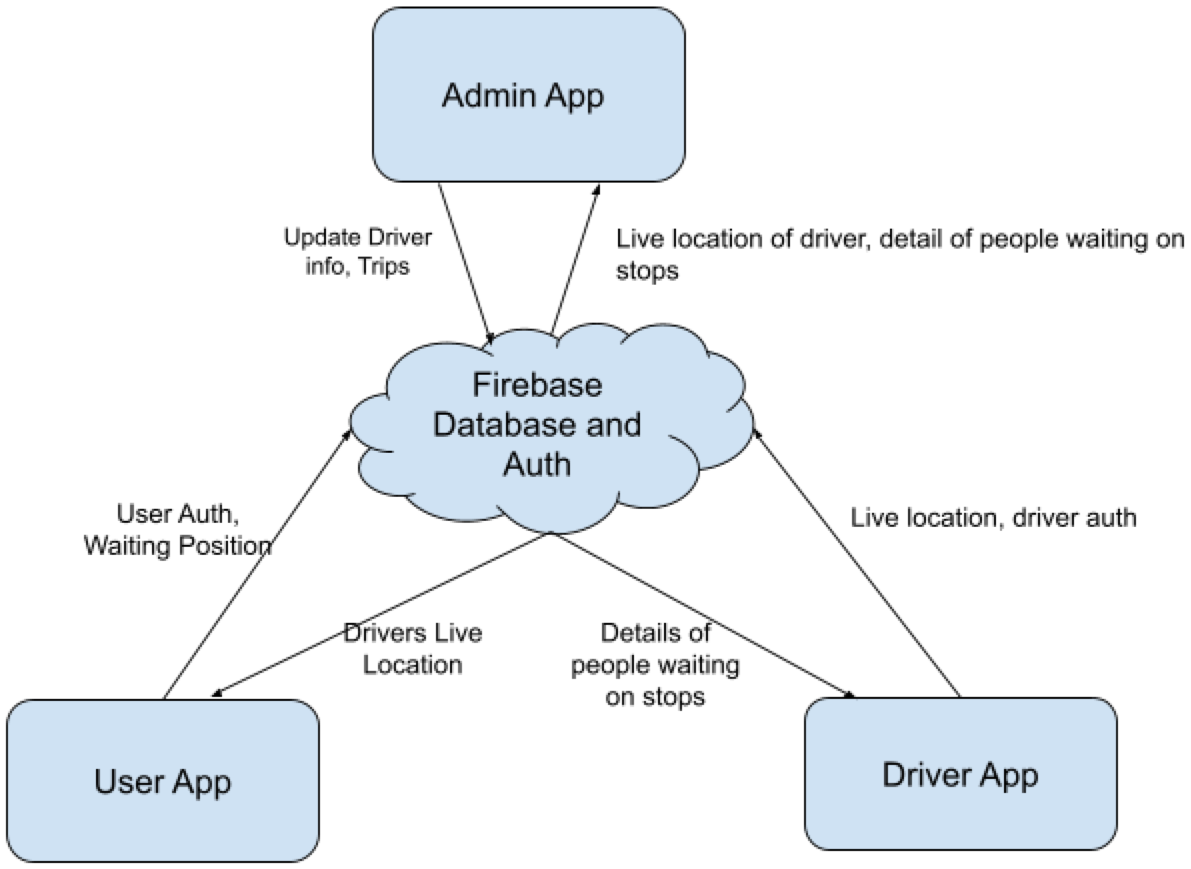

Admin App
Update Driver info, Trips
Live location of driver, detail of people waiting on stops
Firebase Database and Auth
User Auth, Waiting Position
Live location, driver auth
Drivers Live Location
Details of people waiting on stops
User App
Driver App

Fig 1. The Architecture of the App

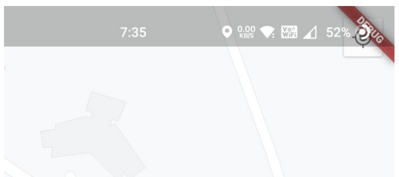
The changes were made to two main parts of the transportation app: the user app and the driver app. The user app enables students and faculty to view the location of buses on campus in real time, providing them with valuable information to make informed decisions about their transportation options. The driver app allows drivers to log in and manage their trips, ensuring accurate and up-to-date information is available to users.

The app uses the Flutter_map package to display an interactive map, which has been enhanced with custom markers and improved functionality compared to the previous implementation. Users can now see the location of buses, as well as landmarks and stops on the map, with each marker assigned a unique number for easy identification.

How marker were added to the map?

Here's a brief explanation of how I added markers using the flutter_map package:

1. **Import the package**: First, I imported the flutter_map package into my project by adding it to the pubspec.yaml file and importing it into the relevant Dart file.
2. **Create a map widget**: Next, I created a FlutterMap widget in my app. This widget displayed the map, and I customized it by setting its initial position, zoom level, and other properties.



3. **Add markers to the map**: To add markers to the map, I created a list of Marker objects. Each Marker object represented a single marker on the map and required a LatLng object for its position (latitude and longitude). I also set various other properties for the marker, such as its icon, size, and anchor point.
4. **Add markers to a MarkerLayerOptions object**: After creating the list of markers, I added them to a MarkerLayerOptions object. This object was used to add markers to the map.
5. **Add MarkerLayerOptions to the map**: Finally, I added the MarkerLayerOptions object to the layers property of the FlutterMap widget. This ensured that the markers were displayed on the map.

By following these steps, I successfully added custom markers to the map in my transportation management app using the flutter_map package. These markers greatly improved the user experience by allowing users to quickly identify and reference specific locations, landmarks, and stops within the app.

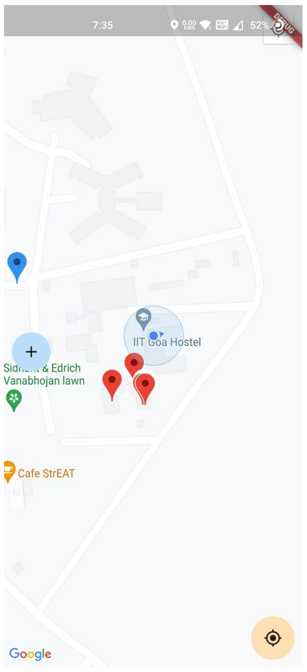
How are we getting the current location?

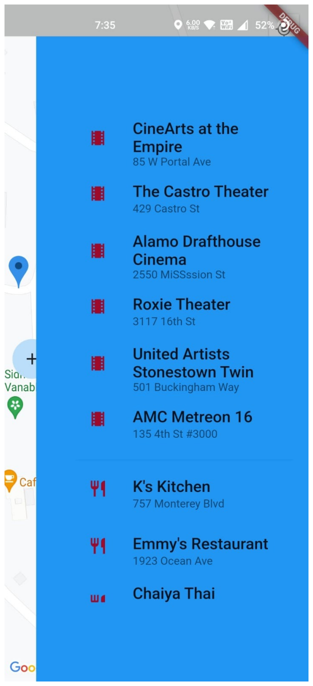
We are using the geolocator plugin for getting the current location of the user/driver. First, we check that location permission is granted or denied. If it is granted, then we are checking whether the location is enabled or not. If not enabled, then we are asking users to enable their location. Then using the geolocator plugin, we are accessing the current location in the form of latitude and longitude.

UI/UX Design and Enhancements

Fig 2. The Main Screen **Fig 3. On click Screen**

UI of Rider App





CineArts at the Empire
85 W Portal Ave

The Castro Theater
429 Castro St

Alamo Drafthouse Cinema
2550 MiSSssion St

Roxie Theater
3117 16th St

United Artists Stonestown Twin
501 Buckingham Way

AMC Metreon 16
135 4th St #3000

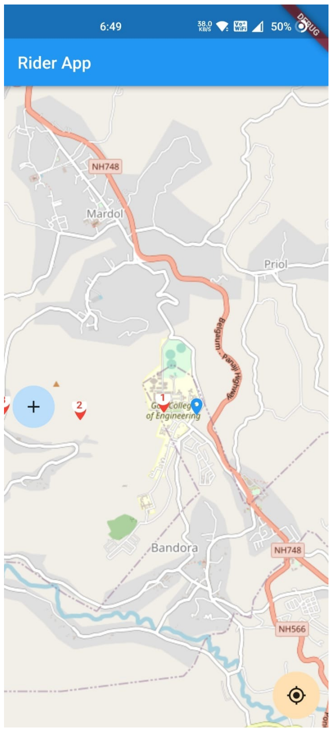
K's Kitchen
757 Monterey Blvd

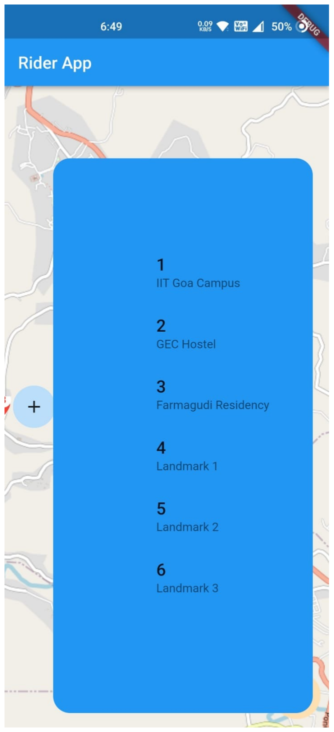
Emmy's Restaurant
1923 Ocean Ave

Chaiya Thai

Fig 2. The Main Screen **Fig 3. On click Screen**

Previous UI of Rider App





1
IIT Goa Campus

2
GEC Hostel

3
Farmagudi Residency

4
Landmark 1

5
Landmark 2

6
Landmark 3

Fig 4. The Main Screen **Fig 5. On click Screen**

Updated UI of Rider App

In this project, a key focus was placed on enhancing the user interface (UI) and user experience (UX) design to provide a more intuitive, visually appealing, and user-friendly experience. The following improvements and design elements have been implemented to achieve this goal:

1. **Layout and Visual Design:** The overall layout of the app was redesigned to ensure a clean, modern, and minimalistic look. This approach streamlines the user's experience by clearly displaying the necessary information, reducing visual clutter, and making it easier for users to navigate through the app.

2. **Color Scheme and Typography:** A consistent color scheme and typography were chosen to create a cohesive visual identity for the app. This choice not only improves the aesthetic appeal of the app but also enhances readability and usability, making it more accessible to a broader audience.

3. **Custom Icons and Graphics:** Custom icons and graphics were incorporated into the app to provide users with visual cues that help them understand the app's functionality more effectively. These visual elements serve to guide users through the app, allowing them to access features and information more efficiently.

4. **Navigation and User Flow**: The app's navigation structure was reorganized to provide a more intuitive user flow. This redesign aimed to make it easier for users to access key features and information with minimal effort, reducing the learning curve and enhancing the overall user experience.

5. **Responsiveness and Compatibility**: The UI/UX design was created to ensure responsiveness and compatibility with a wide range of devices and screen sizes. This approach ensures that users can seamlessly access and use the app regardless of their device, offering a consistent experience across various platforms.

6. **User Feedback and Iteration**: Throughout the development process, user feedback was gathered and incorporated into the design to ensure that the app meets the needs and preferences of its target audience. This iterative process allowed for continuous improvement and refinement of the UI/UX design, resulting in a more user-centric solution.

By focusing on these UI/UX design aspects, the app offers a more intuitive, visually appealing, and user-friendly experience that caters to the needs and preferences of its users. This attention to detail ultimately contributes to the app's overall success, providing users with an efficient and effective transportation management solution.

Results

The updated implementation of the Rider and Driver apps has shown promising results in terms of improved user experience, enhanced functionality, and overall app performance. The use of the 'flutter_map' package, along with other library updates and replacements, has contributed to a more lightweight and stable application that caters to the users' needs effectively. The addition of numbered map markers and new features in the Rider app has streamlined navigation and communication between users and the transportation system.

Discussion

The process of updating and enhancing the apps has provided valuable insights and learning opportunities. Some key takeaways from this project include:

1. The importance of staying up-to-date with the latest technologies and libraries to ensure app compatibility and performance across different platforms and devices.
2. The significance of user interface and user experience design in creating an intuitive and visually appealing application that caters to users' needs and preferences.
3. The value of adopting efficient and lightweight solutions, such as the 'flutter_map' package, to improve app performance and reduce load on devices.

Challenges Faced

During the course of this project, several challenges were encountered and overcome, which led to valuable learning experiences. Some of the key challenges faced included:

1. **Integrating Folium in Python**: Initially, an attempt was made to use Folium, a Python library for creating interactive maps, to implement the mapping features of the app. However, this approach proved to be challenging due to compatibility and integration issues with the Flutter framework. Additionally, adding custom markers to the Folium maps was not feasible, which limited the app's functionality and usability.

2. **Switching to Flutter Map**: To address the challenges faced with Folium, the decision was made to use the 'flutter_map' package instead. This switch required extensive research and understanding of the new package's capabilities, which was time-consuming but ultimately led to a more efficient and lightweight solution for the app's mapping features.

3. **Learning Flutter from Scratch**: As a beginner in the Flutter framework, the learning curve was steep. To gain the necessary knowledge and skills, several resources, including Udemy courses, were utilized. This process was challenging but rewarding, as it resulted in a solid understanding of the framework and its potential for cross-platform app

development.

4. **Balancing Time and Effort**: Managing the time and effort required for learning a new framework, researching suitable solutions, and implementing changes was a challenge. However, through careful planning and prioritization of tasks, the project was completed successfully, and valuable lessons were learned that will be beneficial in future app development projects.

What Did I Learn by Doing This Project?

Through the process of updating and improving the Rider and Driver apps, several important lessons were learned:

- **Gaining experience in the Flutter framework**: This project provided an opportunity to learn and work with the Flutter framework, which is a powerful tool for developing cross-platform applications. This experience has expanded my skillset and can be applied to future app development projects.

- **Understanding the importance of user-centered design**: The project highlighted the significance of focusing on user needs and preferences when designing app features and functionality. This understanding will be valuable in future projects, ensuring that users' needs are always prioritized.

- **Learning to update and replace deprecated libraries:** The project involved updating and replacing deprecated libraries, which is essential for maintaining app performance, compatibility, and stability. This experience has emphasized the importance of staying up-to-date with the latest technologies and libraries in the ever-evolving field of software development.

- **Developing problem-solving skills:** The process of identifying issues, finding suitable solutions, and implementing changes have helped improve problem-solving abilities, which are crucial in the field of software development and can be applied to various projects in the future.

References

How to deal with deprecated libraries. Medium.
https://medium.com/@matthiassma/flutter-how-to-deal-with-deprecated-libraries-d469b9d6b377

Flutter & Dart - The Complete Guide [2023 Edition]- Udemy Course
https://www.udemy.com/course/learn-flutter-dart-to-build-ios-android-apps/

Flutter Geolocation and Maps Integration. Medium.
https://medium.com/flutter-community/flutter-geolocation-and-maps-integration-8e7edec84266

Flutter_map package documentation.
https://pub.dev/packages/flutter_map

Flutter documentation.
https://flutter.dev/docs

Flutter for Android
https://flutter.dev/docs/development/platform-integration/android

Location Package
https://pub.dev/packages/location

Geolocator Package
https://pub.dev/packages/geolocator

Location in Flutter
https://medium.flutterdevs.com/location-in-flutter-27ca6fa1126c