IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

Fleet: A Case Study on the Comparative Analysis of the Mobile Application Framework

[1]Shreeya Gundamaraju, [2]Yugesh Reddy Sappidi, [3]Kusumalatha Karre
[1]Student, [2]Student, [3]Assistant Professor
[1]Department of Electronics and Computer Engineering,
[1]Sreenidhi Institute of Science and Technology,Hyderabad,India.

Abstract : With enormous enhancements in the field of mobile application development , we are at a time where multi-platform applications can be developed using a Unicode. This paper utlilises the case study; Fleet; A travel app implemented using Flutter and summarizes the importance of Flutter and outlines the advantages of flutter over other competent technologies like react native .Fleet,Travel at your feet, is a mobile application that uses the Google Maps API and Firebase to provide personalized and interactive travel experiences for users. The app features a search bar that utilizes the Google Places API to suggest relevant travel destinations, and an itinerary builder that uses the Firebase Realtime Database to store trip data. Additionally, the app provides personalized travel recommendations based on the user's location and trip itinerary using a recommendation algorithm. Navigation features using the Google Maps API, social features with Firebase Cloud Firestore, and secure authentication with Firebase Authentication service are also included. Overall, the Flutter Travel App provides a seamless and personalized travel experience for users.

I. INTRODUCTION

I. INTRODUCTION

Mobile phones have consistently attracted customers since their introduction in 1973 by offering numerous features on a single device. While there has always been a devoted following for cell phones due to their portability and ease of communication, this fan base grew exponentially after the introduction of Apple's iPhones with iOS and Google's Android operating systems. Due to the price reductions and technical advancements brought about by competition in this market, these devices were more accessible around the globe. The original, bulky device has evolved into these smaller, taller, and thinner models called "Smart Phones". Companies began offering us wonderful upgrades year after year in an effort to stand out. Due to this competitiveness, businesses are now trying to draw customers by providing hardware and software enhancements to solve non-existent problems. While smart phones having services like camera, music, and email can be ascribed to the convenience of not having to carry around several gadgets, features like front camera, huge computing power, foldable devices or ultra-fast charging are some of the upgrades that had more entertainment value than the convenience factor.
With multiple hardware and software enhancements being released every year, a mobile phone has transformed from being a portable telephone into a portal to a new world where almost every feature of the Smart Phone can either make or break an industry. For example, telephone directories and contact books make no sense now because we have huge storage in our device which is more accessible and convenient. Smart Phones also created an entire new type economy called “Mobile Economy” just in the form of Mobile Application Development. This economy is generating at least a couple of hundred dollars of revenue and does not include the manufacture, design and marketing of these devices.

II. LITERATURE SURVEY.

2.1 Mobile Application Development

Mobile Application Development can be defined as the art of turning an abstract idea into a tangible, interactive and engaging experience that is accessible to millions of people around the world through the power of mobile devices. It's a process of taking complex business requirements, creative designs, and cutting-edge technology, and transforming them into an intuitive and user-friendly interface that seamlessly integrates with the user's daily life. It's the ultimate expression of creativity and innovation, where developers use their technical skills to bring life to their imagination, and deliver an experience that delights users, simplifies their tasks, and exceeds their expectations. In short, Mobile Application Development is the magic that makes the digital world feel more human.

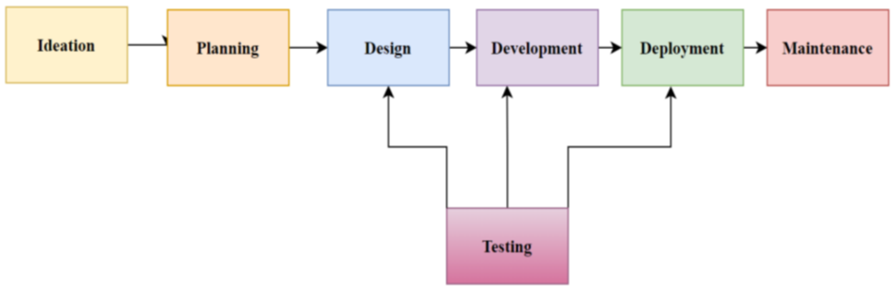


Fig-1: Mobile Application Development Process

2.2 Architecture of an Application:

The architecture of a mobile app typically consists of several layers or components that work together to provide the functionality and user experience of the app. The exact architecture can vary depending on the specific app and the technologies used, but here is a general overview of the common components:

- **User interface (UI):** This layer is responsible for displaying the app's interface to the user and handling user input. It includes the visual design, layout, and components such as buttons, menus, and text fields.
- **Presentation layer:** This layer handles the business logic of the app, such as processing user input, retrieving data, and communicating with other components. It may also include libraries or frameworks for displaying data in the UI.
- **Data layer:** This layer manages the app's data, such as user profiles, settings, and content. It may use a database or other data storage technology to store and retrieve data.
- **Networking layer:** This layer handles communication between the app and external services or APIs, such as retrieving data from a server or posting data to a social media platform.
- **Security layer:** This layer is responsible for ensuring the app's security, such as encrypting user data and implementing authentication and authorization mechanisms to control access to secure data and functionality.

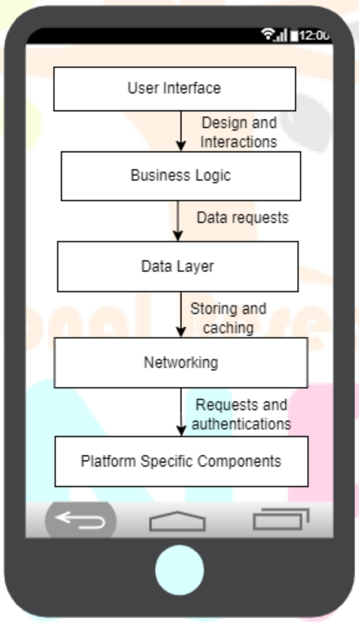


Fig. 2. General Architecture of a Mobile Application

These components can be organized in different ways depending on the specific architecture and design of the app. Other apps may use a client-server architecture, where the app communicates with a server to retrieve or store data. Ultimately, the architecture of a mobile app will depend on the specific requirements and goals of the app.

2.3 Type of Applications:

Based on the arrangement of the layers, there are three primary types of mobile apps: Native, Hybrid, and Web apps. Each type has its own advantages and disadvantages, and choosing the right one for your needs depends on several factors.

- **Native apps**: These are developed specifically for a particular platform or operating system, such as iOS or Android, and are written in the platform's native programming language. This means that they can take full advantage of the device's hardware and software features, resulting in fast performance, smooth animations, and a highly responsive user interface. Native apps can also function offline, as they can store data and content directly on the device. However, developing native apps can be more expensive and time-consuming than other types of apps, as separate versions must be created for each platform.

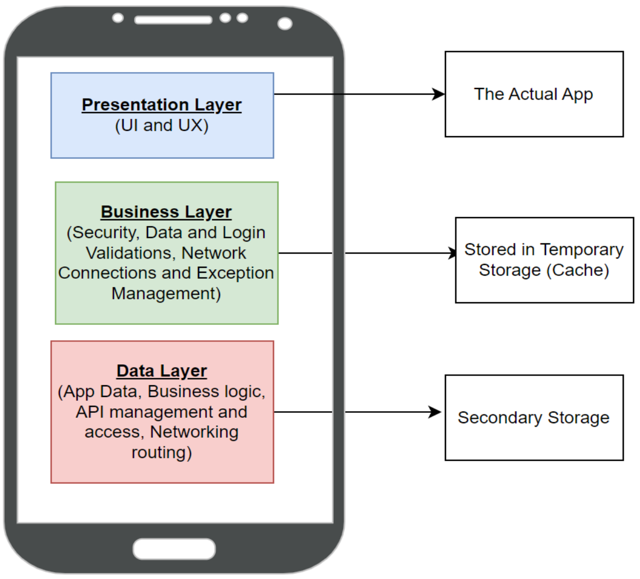

Presentation Layer
(UI and UX)
Business Layer
(Security, Data and Login Validations, Network Connections and Exception Management)
Data Layer
(App Data, Business logic, API management and access, Networking routing)
The Actual App
Stored in Temporary Storage (Cache)
Secondary Storage

Fig. 3. Native App Structure

- **Hybrid apps**: They are a combination of native and web apps. They are built using web technologies, such as HTML, CSS, and JavaScript, and are then wrapped in a native container that allows them to be installed and run on a device like a native app. Hybrid apps can access some device features, but not all, and their performance may not be as fast as native apps. However, they are easier and faster to develop than native apps, as they can be built using a single codebase that can be deployed across multiple platforms.

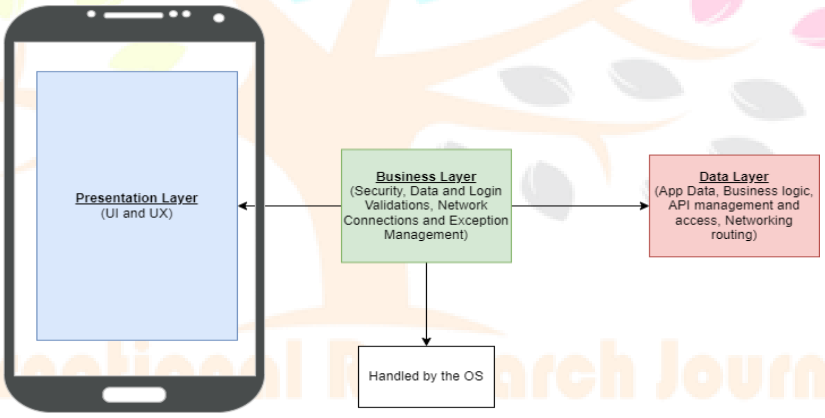

Presentation Layer
(UI and UX)
Business Layer
(Security, Data and Login Validations, Network Connections and Exception Management)
Data Layer
(App Data, Business logic, API management and access, Networking routing)
Handled by the OS

Fig. 4. Hybrid App Structure

- **Web apps**: Web apps are designed to be accessed through a mobile device's web browser, and do not need to be installed or downloaded from an app store. They are built using web technologies, and can be accessed from any device with a web browser, making them highly accessible and easy to maintain. However, web apps may not have the same level of functionality as native or hybrid apps, as they cannot access all of the device's features, and may not be able to function offline.

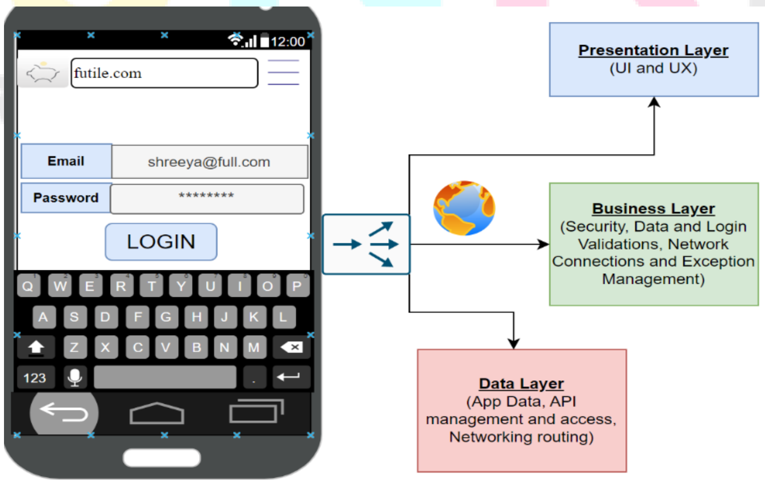

12:00
futile.com
Email
shreeya@full.com
Password
****
LOGIN
Presentation Layer
(UI and UX)
Business Layer
(Security, Data and Login Validations, Network Connections and Exception Management)
Data Layer
(App Data, API management and access, Networking routing)

Fig-5: Web Applications

When deciding which type of app to build, it's important to consider the specific needs and goals of the project. Native apps are best for highly specialized or complex apps that require access to specific hardware or software features, and for apps that require offline functionality. Hybrid apps are a good choice for apps that need to be deployed across multiple platforms, or for apps that do not require access to all of the device's features. Web apps are ideal for simple or lightweight apps that can be accessed easily from any device with an internet connection.

2.4 Application Development Frameworks:

Android and iOS are the two most common Operating systems commonly used across the world. Their architectural frameworks are very different. Traditionally, there were separate application developers for both of them, which was expensive and time consuming. But after a while, all the components were abstracted and the concept of Application Framework was introduced.
An application development framework (ADF) is a platform that provides developers with the tools and resources necessary to build software applications more efficiently.
Some of the key features of an ADF include:

- **Pre-built libraries and modules**: ADFs typically come with pre-built libraries and modules that can be used to accelerate the development process. These libraries and modules are designed to be highly reusable.
- **Integrated development environment (IDE):** IDEs are often included which provide developers with a set of tools and resources for writing, testing, and debugging code.
- **Data access and management:** Application Framework often include tools for accessing and managing data, such as object-relational mapping (ORM) frameworks, data access libraries, and query builders.
- **Security features**: Built-in security features, such as authentication and authorization frameworks, offered by the frameworks ensure that the applications are secure and protected against malicious attacks.
- **Cross-platform support:** They might support multiple platforms like web, mobile, and desktop applications.

III. EXISTING FRAMEWORK: REACT NATIVE

React Native is an open-source mobile application development framework by Facebook that enables developers to build mobile applications using the same codebase for both iOS and Android platforms. It is based on the popular ReactJS framework, which is primarily used for building web applications. It allows developers to create cross-platform mobile applications using JavaScript, which is one of the most widely used programming languages. It provides a set of pre-built UI components that developers can use to create highly responsive and interactive user interfaces. The framework also supports third-party plugins and libraries, which makes it possible to add functionality to the application without having to write any native code.

3.1 Architecture:

The present React Native architecture a set of three main fundamentals:

- **The JavaScript Thread:** The complete JavaScript code is compiled here. When an app is packaged for production, JavaScriptCore executes the package when the user launches it.
- **The Native Thread:** This is the place where the native code is executed. When the app has to update the UI, execute native functions, etc., this component manages the user interface and provides smooth communication with the JS thread. All of the native modules are loaded at startup, so if the user requests access to them, they will always be included.
- **The Shadow Thread**: The layout of your application is calculated here. With the backing of Facebook's own layout engine, Yoga, this cross-platform framework manages the process. It calculates, alters, and transmits flexbox layouts according to the app's User Interface.

3.2 Advantages:

- **Hot Reloading:** Hot Reloading feature that allows developers to see the changes they make to the code in real-time, without having to reload the entire application. This accommodates faster development process because iterations are done quickly.
- **Third-Party Libraries:** React Native has a vast library of third-party libraries that developers can use to add additional functionality to their applications. This can save time and effort, as developers don't have to build everything from scratch.
- **Fast Development:** React Native allows for rapid development, as developers can write code applications across multiple platforms. This reduces the amount of time and effort required to develop and maintain separate applications for different platforms.
- **User Interface:** React Native provides a smooth and responsive user interface, as it uses native components that are optimized for mobile devices. This leads to better user experience and higher user engagement.
- **Cost-Effective:** As React Native allows developers to build applications for multiple platforms using a single codebase, it can be a cost-effective solution for businesses and organizations looking to develop mobile applications.
- **Easy Integration:** React Native can be easily integrated with other technologies and frameworks, including Redux, GraphQL, and Firebase. This makes it easier to build complex applications that require integration with different services and systems.

3.3 Disadvantages of React Native:

While React Native offers many advantages, it also has some disadvantages that developers should be aware of before choosing it as their mobile app development framework. Here are some of the key disadvantages of React Native:

- **Performance**: React Native apps are very slower than the Native apps in many cases. This is because React Native uses a bridge to communicate between JavaScript and native code, which can introduce some latency and performance overhead.
- **Native API support:** Although React Native provides a large number of pre-built components, there may be some situations where developers need to access the native APIs directly. While it is possible to use native modules in React Native, the process is more complicated in a fully native app.

- **Learning curve:** Although React Native uses a similar programming model to ReactJS, there is still a learning curve for developers who are new to the framework. Developers may need to spend some time learning how to use the framework effectively and efficiently.
- **Outdated third-party library support:** While React Native has a large and growing ecosystem of third-party libraries and plugins, there are still some areas where the libraries is unable to replicate the native app development. This can be a limitation for developers who need to use a specific library or tool that is not yet available in React Native.
- **Debugging**: Debugging React Native apps can be more challenging than debugging native apps, due to the complexity of the bridge between JavaScript and native code.

As an alternative to this framework, Google released the Flutter Framework for building high-performance, cross-platform mobile applications which is discussed next.

IV. PROPOSED SYSTEM – FLUTTER

Flutter is an open-source UI toolkit developed by Google for iOS, Android, and the web. Flutter has gained popularity among developers due to its fast development cycle, ease of use, and ability to create beautiful, native-looking interfaces.

4.1 Architecture of Flutter:

Flutter's architecture is based on a reactive programming model that enables developers to build complex UIs with ease. At the core of the architecture is the Flutter engine, which is responsible for rendering the UI and handling user input. The engine is built on top of the Skia graphics engine, which provides a high-performance, hardware-accelerated 2D rendering engine. The Flutter framework provides a rich set of APIs for building UI components, handling user input, and managing state.

The key components of the Flutter architecture are:

- **Dart Framework**: Flutter framework is powered by the Dart, an object-oriented language with a C-style syntax, at the core of Flutter. It is responsible for providing the basic building blocks that developers use to build their applications.
- **Flutter Engine:** The Flutter Engine is a low-level platform that provides a set of core libraries for rendering graphics, interacting with the device hardware, and managing input events. It also includes a just-in-time (JIT) compiler and a garbage collector.
- **Widgets:** Widgets are the building blocks of Flutter applications. They are responsible for rendering the user interface and responding to user input events. Flutter provides a rich set of widgets that can be customized and combined to create complex UI layouts.
- **Material Design and Cupertino:** Flutter provides two sets of widgets, one for implementing Google's Material Design guidelines and another for implementing Apple's Cupertino design language. These widgets are built on top of the core set of widgets and provide a consistent and familiar look and feel for Android and iOS devices.

4.2 Advantages:

Flutter is an important technology in modern mobile app development for several reasons. Along with the general advantages of Application Development Framework.

- **Cross-Platform Development:** Flutter allows developers to build applications not only for different Operating Systems but also for different platforms like Web, Mobile, Tab etc using a single codebase. This reduces development time and costs, as developers do not have to write separate code for each platform.
- **Fast Development Cycle:** Flutter's hot reload feature enables developers to see changes to the code immediately in the app, without having to recompile the entire app. This makes development faster and more efficient.
- **Beautiful UI:** Flutter's widgets and animation APIs enable developers to create beautiful, native-looking UIs that are fast and responsive. This helps to improve the user experience and makes the app more appealing to users.
- **Large Community:** Flutter has a large and growing community of developers and enthusiasts who are contributing to the development of the technology. This community provides support, resources, and tools to help developers build high-quality applications.

In conclusion, Flutter is an important technology in modern mobile app development, offering cross-platform development, a fast development cycle, beautiful UI, and a large community of developers. Flutter's reactive programming model and rich set of APIs make it easy to build complex UIs and manage state in the application. With its growing popularity and strong support from Google, Flutter is poised to become one of the leading mobile app development technologies in the coming years.,

4.3 Enhancements Over The Existing System

Flutter and React Native are both popular cross-platform frameworks for mobile app development. While they share some similarities, they also have some differences in terms of architecture, performance, and development experience.

1. **Programming language:** Flutter uses the Dart programming language, while React Native uses JavaScript. Dart is an object-oriented programming language that is compiled ahead of time, whereas JavaScript is a scripting language that is interpreted at runtime.
2. **Architecture:** Flutter uses a widget tree architecture, while React Native uses a component-based architecture. In Flutter, everything is a widget, which makes it easy to build complex UIs by nesting and composing widgets. React Native, on the other hand, uses reusable components that are easier to manage and update.
3. **Hot reload feature:** Both Flutter and React Native have a hot reload feature that allows developers to see changes to their code in real-time without having to rebuild the entire app. This speeds up the development process and allows for quicker iteration.
4. **Web development support:** Flutter has built-in support for web development, which means that developers can build web applications using the same codebase as their mobile apps. React Native, on the other hand, has limited support for web development and requires additional tools and libraries to build web apps.

5. **Built-in widgets/components:** Flutter comes with a set of built-in Material Design widgets that can be used to build beautiful and consistent UIs across platforms. React Native, on the other hand, comes with built-in iOS and Android widgets that mimic the native UI of each platform.
6. **Customization:** Flutter widgets allow for more customization than React Native components. This means that developers can more easily create unique and custom UI designs with Flutter. React Native components, on the other hand, have more limited customization options.
7. **Performance:** Flutter is known for its strong performance and faster rendering compared to React Native. However, React Native's performance may vary based on the number of components and the complexity of the app.
8. **Community:** Flutter has a smaller community compared to React Native, which means that there may be fewer third-party packages and libraries available. React Native has a larger community, which means that there are more resources available for developers.
9. **UI design:** Flutter is generally better for building custom UI designs, while React Native is better for building native-looking UI designs. Flutter allows for more creative freedom when it comes to designing UIs, while React Native is more focused on providing a consistent and native look and feel across platforms.
10. **Learning curve:** Flutter has a learning curve when it comes to learning the Dart programming language which is very easy to study for beginners. React Native has a learning curve when it comes to learning JavaScript and the React framework which is complex without prior knowledge.

Flutter and React Native are both popular frameworks for mobile app development, and both have their strengths and weaknesses but Flutter's superior performance, customizable widgets, and easier learning curve are some of the reasons why Flutter may be considered to be better than React Native. In view of all these advantages, we have decided to build our Travel Application with Flutter instead of React Native.

V. FLEET: *TRAVEL AT YOUR FEET*

5.1 Introduction

Travel apps are mobile applications designed to assist and enhance the travel experience of users. These apps can provide a range of features and services, such as flight booking, hotel booking, itinerary planning, local recommendations, translation, transportation navigation, and more. They can also offer real-time updates on travel-related information, such as weather forecasts and flight delays. Travel apps aim to make the travel experience more convenient, efficient, and enjoyable for users by providing them with the necessary tools and resources to plan and manage their trips. There are many alternatives for Travel Apps in the market for booking apps and Travel Itinerary apps but there are very few with intensive information and real-time suggestions without any paid partnerships in the Market. Thus came the idea for Fleet- Travel on Feet, an app which provides user with information about local attractions along with real-time and personalized hotel recommendations.

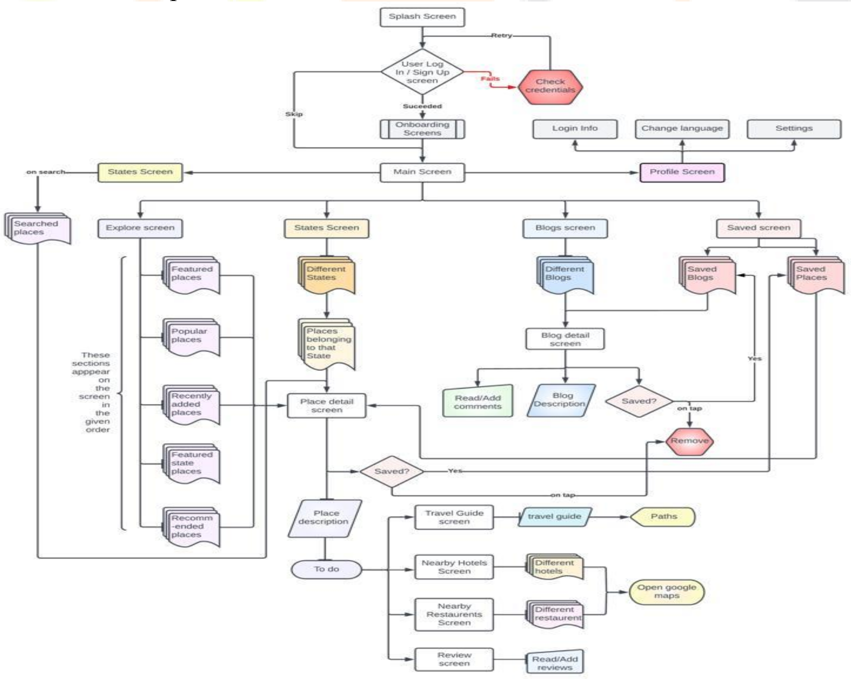


Fig-6: Use Case Diagram

5.2 Choosing the Technology:

In the initial stage, we are developing the app with information on nine states in India so that we can test the waters. But we would like to increase the scope of the app across India and may be across the globe. This would require us to add a lot of different modules in iterative process which means our app should be scalable. Also, travel apps should always be updated in order to attract the customers so needed something that is customisable. Along with this our app also has two special features: Real-Time Hotel

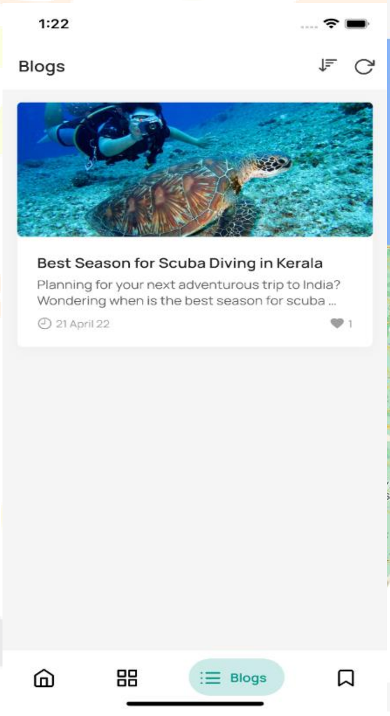
Recommendations based on Live Location and a Travel Blog where people can post their beautiful stories and experiences in our trip. This means, the performance of the app also plays a very important part for the application.
After a lot of considerations on other options like Native applications or Web applications, we decided that our app should be Hybrid because of the storage requirements for the information and performance requirements for real-time recommendations. We also wanted to be ready for future enhancements that might be required. So, our choices narrowed down to React Native and Flutter. React Native seemed like an appropriate choice at first because of the modular architecture which allows future upgrades but after a few trials with React and Flutter, we decided Flutter was the better choice for our app.

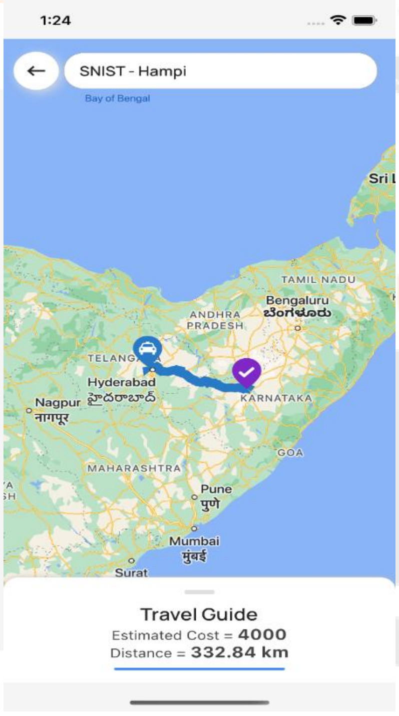
5.3 Initial Iterations:

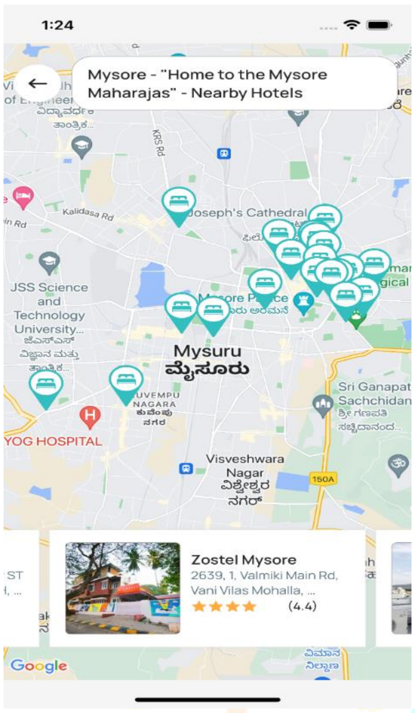
During our initial trials, we tried to build a small portion of the app with both React Native and Flutter in order to compare our apps. Both React Native and Flutter had Hot Reload features where we can see the changes instantly which saved us a lot of time in this experiment. However, Flutter's mechanism allowed us more accuracy and speed while keeping track of the changes. So, here we were convinced that Flutter has the better Hot Reload Feature.
The User Interface of the app also seemed a lot better on Flutter. Also, since Flutter offered support for Material Design and Cupertino, the design guides of Android and iOS respectively, it was lot easier to build the UI in Flutter.
After that, it came down to the most important part of the decision, which is Performance. We tried to see how React and Flutter worked with the Blog and Real time location so we tried to build that page with React and Flutter. While working with React we noticed that while React Native performed almost similar to Flutter in the initial simple app, the performance was affected quite a bit after adding this feature. It has crashed multiple times and also was slow when it worked. Flutter was easy to build with and did not have this issue.
Thus, after these iterations, we have decided to use Flutter as our Framework for the app "Fleet: *Travel At Your Feet*" given the better performance, UI and easier to code as well.

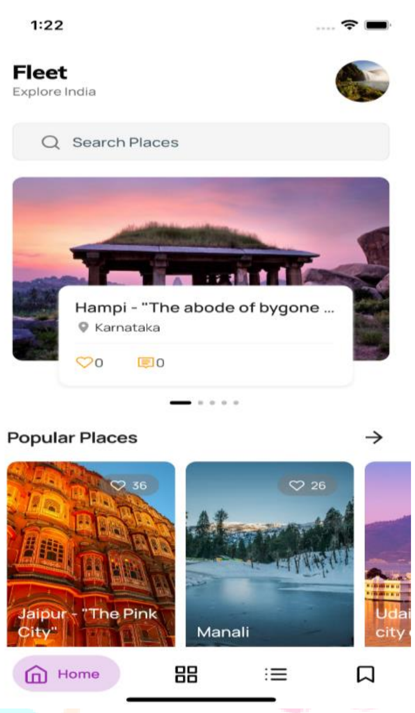
VI. RESULTS

After the development of app, we could understand that Flutter along with Firebase Services was the apt choice for our project. This is because the UI was very good and the performance was also competent. After adding all the information, Flutter along with Firebase was working up to the par. The customisation was easy. We included a Google API for the real-time recommendations which was smooth and it took very little time to load. We also added a blog because both Flutter and Firebase were very easy to code and it took lesser time than we have planned. The app was uploaded in Google Playstore, iTunes and also works as a web application. We have also included a multi-way authentication page by including Facebook and Apple id sign-in including email.


1:22
Blogs
Best Season for Scuba Diving in Kerala
Planning for your next adventurous trip to India? Wondering when is the best season for scuba ...
21 April 22
1
Blogs




1:24
Mysore - "Home to the Mysore Maharajas" - Nearby Hotels
Mysuru
Zostel Mysore
2639, 1, Valmiki Main Rd, Vani Vilas Mohalla, ...
(4.4)
Google



VII. CONCLUSION

Fleet: *Travel at Feet* is a combination of a Travel Itinerary and Local Guide apps with the best recommendations and easy- to-use User Interface and very good list of recommendations. The most novel features of the application are the real-time recommendations which are given based on the live location. The app also has reviews and an inbuilt blog which all include unpaid promotions. It is a free version now but we would like to commercialize the app by adding payment gateways for the bookings and offering the app at a price. The idea of the app is not to monetize with the dishonest paid promotions but to offer true reviews even though at a nominal price. We also would like to expand the app to offer global tour and cater to weekend getaway audience. Fleet definitely helps us to scale our app. Given its advantages and architecture, we are sure that it forms an effective core for the further enhancements of the application.

REFERENCES

[1] A A. Sarkar, A. Goyal, D. Hicks, D. Sarkar and S. Hazra, "Android Application Development: A Brief Overview of Android Platforms and Evolution of Security Systems," 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2019, pp. 73-79, doi: 10.1109/I-SMAC47947.2019.9032440
[2] A. Almisreb, H. . Hadžo Mulalić, N. Mučibabić, and R. Numanović, "A review on mobile operating systems and application development platforms", Sustainable Engineering and Innovation, vol. 1, no. 1, pp. 49-56, Jun. 2019
[3] Xanthopoulos, Spyros, and Stelios Xinogalos. "Mobile app development in HTML5." AIP Conference Proceedings. Vol. 1648. No. 1. AIP Publishing LLC, 2015
[4] The Use of Mobile Applications for Travel Booking: Impacts of Application Quality and Brand Trust,Journal of Vacation Marketing (IF 4) Pub Date: 2021-12-14 , DOI:10.1177/13567667211066544,Tahir Albayrak 1 , M. Rosario González-Rodríguez 2 , Meltem Caber 3 , Sezer Karasakal 4
[5] Programming Flutter : Native, Cross-Platform Apps the Easy Way / Zaccagnino, Carmine - Raleigh, NC : The Pragmatic Bookshelf, 2020 - 275 p. - ISBN: 9781680507645 - Permalink: http://digital.casalini.it/9781680507645 - Casalini id: 5241237
[6] https://www.theseus.fi/bitstream/handle/10024/146232/thesis.pdf?sequence=1
[7] https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/
[8] https://developers.google.com/learn/topics/flutter
[9] https://docs.flutter.dev/
[10] https://www.theseus.fi/bitstream/handle/10024/148975/thesis_Jakhongir_Fayzullaev.pdf?sequence=1&isAllowed=y