ITEM DELIVERY APPLICATION USING REACT NATIVE

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

By

Abhishek Mishra (191287)

Under the supervision of

Dr. Aman Sharma

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology

Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **" Item Delivery Application using React Native"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2022 to May 2023 under the supervision of **Dr. Aman Sharma** (Assistant Professor (SG), Computer Science & Engineering and Information Technology). The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)
Abhishek Mishra (191287)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Dr. Aman Sharma
Assistant Professor (SG)
Computer Science and Engineering

Dated:

Plagiarism Certificate

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ **Department:** _____ **Enrolment No** _____

Contact No. _____ **E-mail.** _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

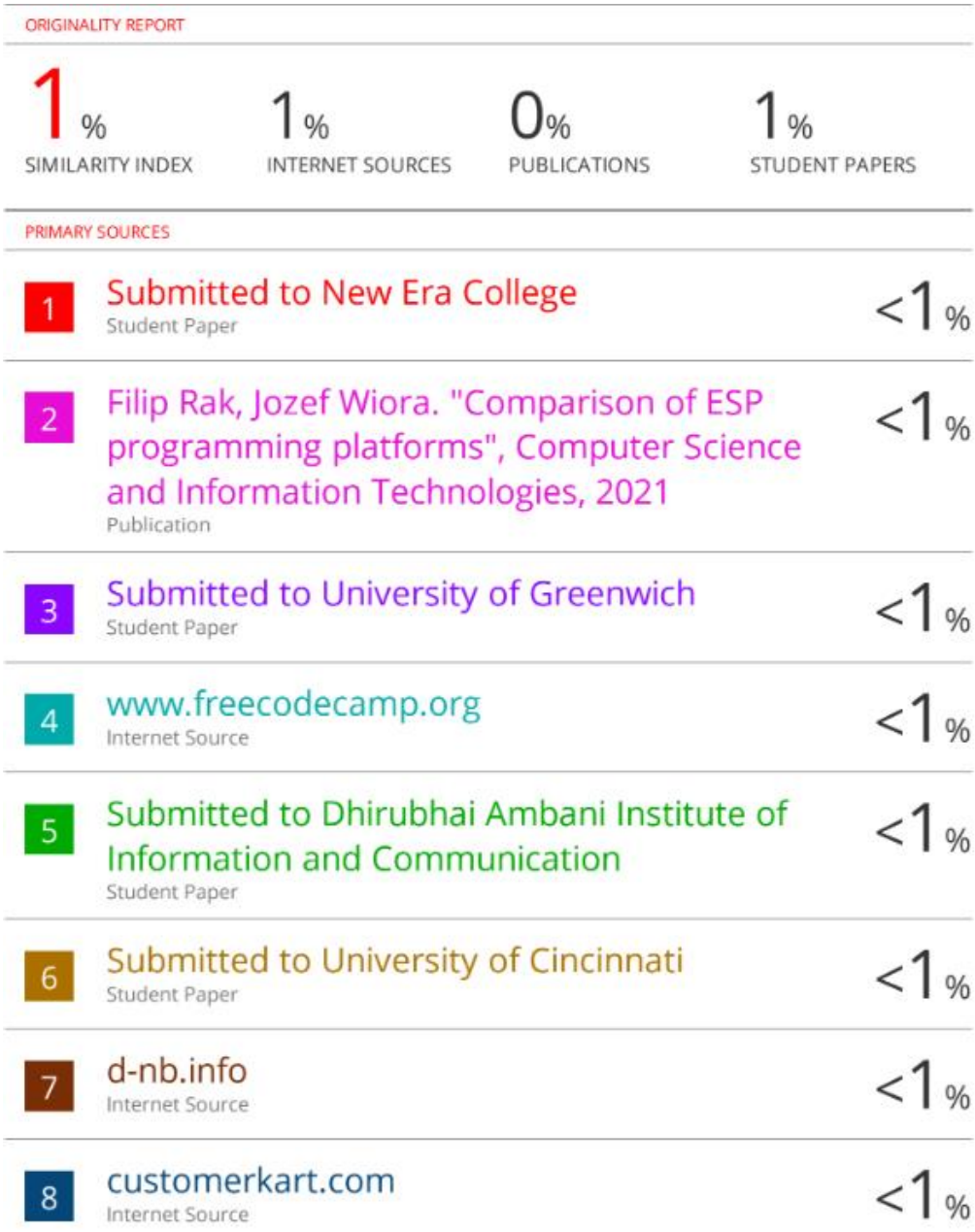
<table>
<tr><td>Copy Received on</td><td>Excluded</td><td>Similarity Index (%)</td><td colspan="2">Generated Plagiarism Report Details (Title, Abstract & Chapters)</td></tr>
<tr><td></td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td>Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

.....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

Originality Report



Acknowledgement

I would like to express my sincere gratitude and appreciation to all those who have helped me in completing this major project. First and foremost, I would like to thank my project supervisor (Dr. Aman Sharma) for providing me with guidance, support, and valuable feedback throughout the project. Your expertise and knowledge have been invaluable to me.

I would also like to extend my gratitude to Jaypee University of Information Technology, for providing me with the necessary resources and facilities to complete this project. Without your support, this project would not have been possible.

My heartfelt thanks to my family and friends, who have been a constant source of encouragement and motivation throughout my academic journey. Your unwavering support and belief in me have been my greatest strength.

Lastly, I would like to thank all the participants who took part in my study, for sharing their valuable time and insights with me. Your contribution has been instrumental in the successful completion of this project.

Thank you all once again.

(Student Signature)
Student Name: Abhishek Mishra
Roll no. : 191287

Table of Content

List of Figures

List of Tables

Abstract

The Item Delivery Application is a React Native-built mobile application that seeks to offer consumers a simple and effective platform for delivering and receiving goods from remote locations. This report covers the features, functionality, and architecture of the application's development. Users will find it simple to explore and finish activities because to the application's user interface's clear and simple design. The backend of the application, which manages user information and orders, was created using Node.js and Express. The application links users with delivery personnel who can quickly and effectively pick up and deliver the desired things. Users may monitor the status of their delivery in real-time and get alerts at various points.

In general, this application offers consumers a dependable and practical on-demand delivery service that satisfies their regular demands. Users looking for a dependable and effective approach to obtain goods from their chosen stores will find the application to be useful due to its features and performance. A top-notch, cross-platform software that works on both Android and iOS devices may be created with the help of contemporary platforms and technologies like Node.js and React Native.

Chapter - 1
Introduction

1.1 Introduction

With the growing numbers of consumers choosing the convenience of having goods and services delivered to their doorstep in recent years, which has resulted in a huge increase in the on-demand delivery market's appeal.. One such segment of the on-demand delivery market is the hyperlocal delivery service, which has gained tremendous popularity in urban areas. Customers of the well-known hyperlocal delivery service Swiggy Genie in India can send as well as receive items inside the city. Because of the service's accessibility, dependability, and reasonable cost, it has seen tremendous growth. This article will examine the creation of an app similar to Swiggy Genie. To enable clients to send and receive products swiftly and easily, we set out to develop an app that would offer a seamless hyperlocal delivery experience. We had to overcome a number of obstacles when designing the app, including creating a user-friendly interface, improving the program's speed, and guaranteeing the safety of information provided by users. We will go through the strategy we used to overcome these obstacles in this report, along with the fixes we put in place to build a productive hyperlocal delivery app. Along with insights into the lessons we learned along the way, we will also give an overview of the main features and functions of the app.

Overall, the process of creating the app was both resilient and gratifying. We anticipate that other companies trying to develop comparable apps to improve their hyperlocal delivery operations would find this report to be a helpful resource. Swiggy. Our app provides a delivery service that enables users to get anything delivered from anywhere within a city. Whether it's forgotten keys, documents, groceries, or even laundry, this application promises to deliver it all in just a few clicks. The app is simple to use, with a user-friendly interface and a range of features designed to make the delivery experience as

seamless as possible. Several stages of the development process were conceptualization, design, development, testing, and deployment. We encountered a number of challenges when developing the app, including enhancing its performance, guaranteeing data security, and connecting with third-party APIs and services [5]. The positive aspect is that we were able to overcome all these challenges and develop a useful item delivery application through thorough planning and efficient implementation.

The technology we are using to implement this application is React Native. Facebook developed the well-known open-source React Native framework for building mobile applications [6]. It enables programmers to create native mobile applications for both the iOS and Android platforms using a single codebase and is built on top of React, a well-known JavaScript toolkit for creating user interfaces. One of React Native's key benefits is that it enables programmers to create high-quality, effective mobile applications using well-known web development languages like JavaScript, CSS, and HTML. Because of this, it is the perfect option for web developers who wish to switch to building mobile apps without having to learn another programming language or framework. Hot reloading, which enables developers to view their modifications in real-time without needing to relaunch the programme, and a comprehensive set of built-in components which may be readily customised to match the particular demands of the application are just a couple of the helpful features offered by React Native. Due to its simplicity, adaptability, and ability to rapidly develop and refresh applications, React Native has grown to be an increasingly popular option for developing mobile apps. It has been used to create top-notch mobile applications by businesses like Facebook, Instagram, Airbnb, and Tesla. Developing software that is adaptable to several different kinds of hardware is known as cross-platform software creation. Microsoft Windows, Linux, macOS, or any combination of these operating systems may all be utilised by a cross-platform application. A cross-platform application is one that works exactly the same on any type of device, such as an internet browser or Adobe Flash. React Native is also quite configurable, giving programmers the freedom to design distinctive user experiences that meet the

needs of an organisation or project. This is made feasible by the adaptability of React Native's architecture and the simplicity of incorporating unique animations and effects.

1.2 Problem Statement

As more people rely on delivery services to have items delivered from one location to another within a city, the item delivery industry is expanding quickly. However, the users' needs for speed, convenience, and security are frequently not met by the current delivery services. The goal of this project is to utilise React Native to develop an item delivery application that overcomes these problems and offers users a seamless, effective, and secure delivery experience. Specifically, the project aims to solve the following problems:

- **Limited delivery options**: confined Delivery possibilities: Users of currently available delivery options are frequently confined to a select few service providers, which results in an insufficient number of delivery possibilities. Users who require prompt and effective delivery of their items may find this to be annoying.
- **Lack of real-time tracking**: Real-time monitoring is often absent from conventional delivery systems, making it challenging for customers to keep track of their goods and get updates in real-time. Which is handled in our project. This helps the user to continuously monitor the item which they have sent. Keeping track of the delivery partner.
- **Ineffective Communication:** Poor and inefficient communication might lead to delays, missing deliveries, and other problems between users and delivery workforce members. This leads to lack of belief of the application user over the delivery partners.

In order to solve these problems and give consumers a seamless, effective, and secure delivery experience, this endeavour aims to develop an item delivery application. I strive to give consumers a delivery service that fulfils their needs

and surpasses their expectations by creating an application that offers a wide selection of delivery options, tracking in real time, effective communication, and strong safety features. Consumers demand prompt and effective delivery of their goods in the rapidly changing environment of today. A trustworthy item delivery software that can provide clients with a flawless and hassle-free delivery experience is becoming more and more necessary as e-commerce and online shopping grow in popularity. However, the market's current delivery apps frequently experience problems like sluggish deliveries, inaccurate tracking, and subpar customer support. Customers become dissatisfied as a result, and the app experiences a decline in sales. Consequently, there is a demand for a product delivery application that can overcome all of these challenges and offer clients a quick, dependable, and open delivery option.

The fact that React Native employs native components instead of web-based ones indicates that it delivers superior efficiency and swiftness. React Native apps become more rapid and responsive as a result, offering users a smooth and seamless experience. React Native is also quite adaptable, giving developers the ability to construct distinctive user experiences that meet the needs of their company or project. This is made feasible by the adaptability of React Native's architecture and the simplicity with which custom transitions and effects may be included.

1.3 Objectives

The objective of our application is to provide an intuitive interface. The application should be simple to use, with instructions that are relatively easy to follow and a user-friendly UI that leads users through the entire process of delivery. To enable clients to send and receive products swiftly and easily, we set out to develop an app that would offer a seamless hyperlocal delivery experience. The objective of this application are as follows:

- **To offer a trust-worthy delivery experience :** This application should make sure that packages arrive on schedule and reliably, with

clear notification and tracking systems to keep users updated at all times. Therefore it offers a dependable delivery service.

- **Support a variety of delivery choices :** In order to satisfy the requirements of numerous users, the application should offer an extensive variety of delivery options, such as food, medicine, documents, tools e.t.c..
- **Delivery at reasonable cost** : The application must give users of this application reasonable service for delivery at an affordable price without sacrificing dependability or performance. Therefore, the cost of the delivery service will be depending upon the distance between the source and the destination.
- **Status of the delivery package :** The user can see the status of their delivery packages in real-time, This application provides the user to track the route of the delivery partner in real-time. This enables the user to keep track of the product they have sent at all times. monitoring the delivery partner.
- **Communication and customer service :** The user of this application would be able to contact the delivery partner and clear their issues, if the user wants he can also communicate with the admin or customer care if he is facing any delivery issues.

1.4 Methodology

In this section we will be discussing the methodology used in the project step by step. In this application we have used React Native as a framework as a front-end mobile application. Development and deployment of React Native applications may be done in two distinct ways: Expo Go and React Native CLI. Developing and deployment of React Native applications may be done in two distinct ways: Expo Go and React Native CLI.

The development environment Expo offers a number of tools and services which render it easier to create and distribute React Native apps. Expo Go is

one of such solutions, an application for smartphones that enables you to view and evaluate your React Native application without the need to set up a different emulator or device. Your application will be started in Expo Go on your smartphone by just scanning a QR code created by Expo CLI. A variety of additional resources are also offered by Expo, including a collection of already built components, a software development server, and a building service that streamlines the process of submitting your application to app stores. Expo is a great option for programmers who want to swiftly develop and test their concepts without needing to deal with the difficulties of creating and distributing a native mobile app, for novices, or for experienced developers.

In this application I have worked in Expo go for better representation of the concept and requirements of the project. After selecting the framework and the platform, the first step toward building our project is to gather all the information and requirements for this project. In this step we identify user demands, determining necessary features, and establishing the project's scope are all part of this process. Then the next step is design and wireframing. After understanding the requirement of the application, objectives and other aspects and putting it together we work on its UI/UX designs. This step includes conceptualising and prototyping the application. This involves establishing how the user flows, developing the interface's functionality and the user experience, and producing an aesthetic that is compatible with the application and the requirements of the intended audience. In order to achieve this we have to follow certain rules:

1. **Identify target users:** The key to developing an effective application is recognising those who are your target users. To understand your users' wants, objectives, and pain spots, perform user study and collect feedback.
2. **Keeping it simple :** The layout must be uncomplicated, clear, and straightforward to use. We should avoid using complicated navigation from one component to another and cluttering our interfaces.

3. **Consistency**: In layout, uniformity is very crucial. In order to provide a uniform and consistent user experience we should use the exact same layout, fonts, colours, and icons throughout the entire application.
4. **Use the right colours** : The choice of shades is crucial in layout. Use colouring that is suitable for your brand and the intended use of the product. To make the user interface easy to comprehend and navigate, refrain from using excessive colours and make sure there is adequate contrast.

Table 1.1 shows the technologies stacked for implementation of the application.

React Native	Front-end framework
Node.js	Back-end
Express.js	API
Stripe	Payment gateway
Firebase	Back-end authentication

Table 1.1 List of technologies stacked for application

After creating the UI of our application and understanding the needs of the users our next step is to stack all the technology requirements. The choice of the technological stacking comes next. The main technology employed to create the mobile application for Android and iOS targeted users is React Native. Node.js for API connectivity, and Firebase for backend services are some more technologies that might be leveraged. Google's cloud-based Firebase platform offers an array of resources and applications for creating and deploying apps for the web and mobile devices. A wide range of backend capabilities provided by Firebase enable programmers to create apps with greater efficiency and speed. There are numerous services provided by Google’s firebase like: Authentication, Cloud messaging Programmers may

deliver alerts to users over many platforms, such as iOS and Android, and the web, using real-time cloud communication. After choosing the technologies the user is then moved to the most important step of developing the code. The application's code is written during the development stage. Fig 1.1 shows. This involves building the application's front- end and back-end, integrating APIs, and testing it for faults and other problems. The application is tested throughout the testing and quality management stage to ensure it satisfies its functionality and operational criteria. Testing for units, validation of integration, and user acceptability testing are all included in this. Following recommendations, documenting all of the efforts made, and making sure the source code is adaptable and manageable to accommodate subsequent upgrades and improvements are crucial through the whole process.

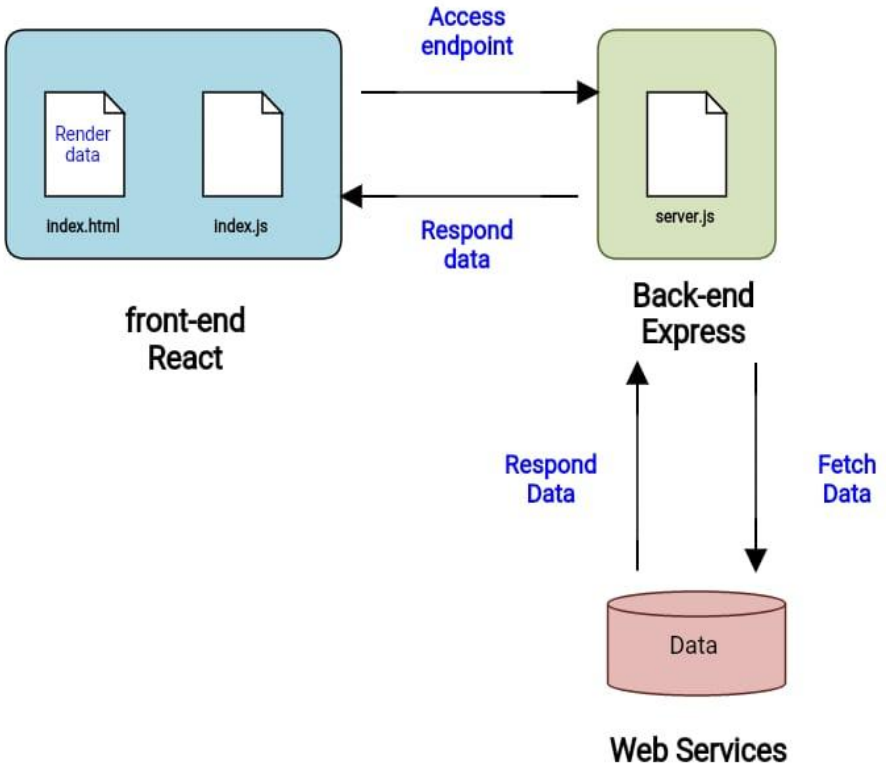


Fig 1.1 Data fetching from back-end and rendering on front-end.

Over the years the React Native is continuously trending in comparison with other technologies as shown in Fig 1.2. The most popular technology giving tough competition to React native is Flutter but due to some disadvantages of flutter and some extra features of React Native makes it more reliable, adaptive and flexible to use. Components, hooks, component's life cycle are some of the key features of React native which makes our development faster.

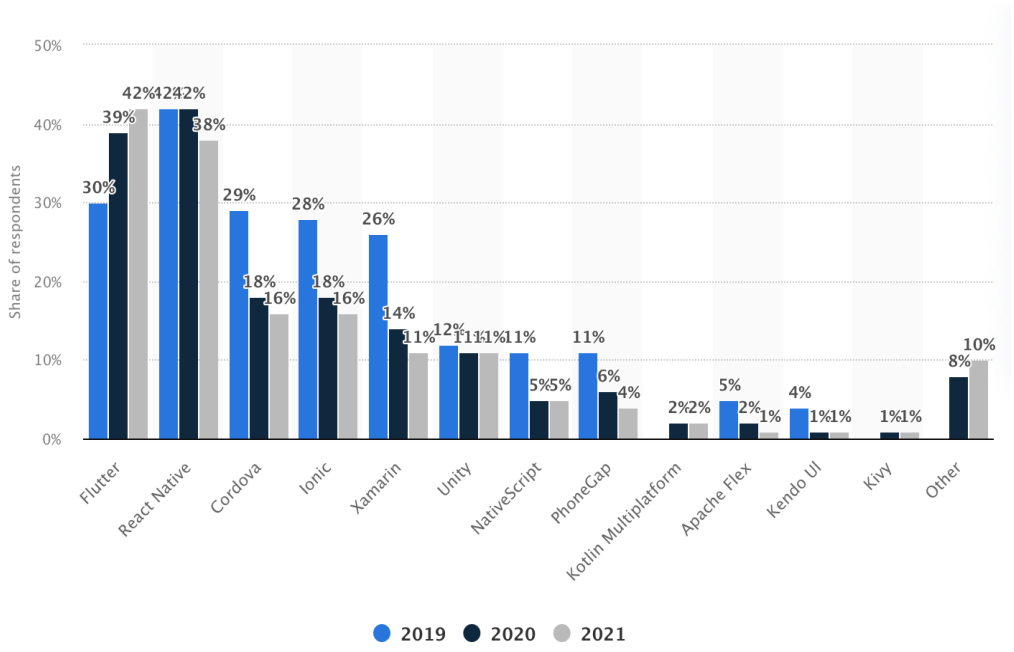


Fig 1.2 Comparison of different Frameworks

Chapter-2
Literature Survey

In this section of this report we will be discussing the literature review we have done to explore existing literature on React native application. A well-appreciated cross-platform programming framework called React Native is used to create mobile apps for both iOS and Android based devices. The framework is renowned for its use, efficacy, and adaptability. One of the most common use cases for mobile applications, and their popularity is increasing rapidly, is product delivering. This evaluation of the literature's current state focuses on React Native application for product delivering. A lookup on different academic databases, including Google Scholar, IEEE Xplore, and the ACM Digital Library, was done to conduct the aforementioned evaluation.

One research by Atul et al. (2021) investigated the creation of an application for product distribution using React Native. According to the report, employing React Native instead of creating separate iOS and Android applications cut development time by up to 30%. The application also offered real-time tracking of the package's progress and could manage several simultaneous requests. The customer experience of a React Native-based goods distribution application was the subject of an additional study by Singh et al. (2020). Users were able to complete delivery orders quickly and effectively because of the application's user-friendly interface, which the research deemed to be straightforward and simple to use.

Riaz et al. (2021) in research paper “Development of a React Native Mobile Application” shows The creation of a laboratory for clinical screening application utilising React Native is explored in this article. The research identifies the benefits of utilising React Native for creating mobile apps that run across platforms and offers details on the creation process. The findings of this literature research indicate there is rising interest in employing React Native to create applications for product distribution. The benefits of using

React Native have been emphasised in several studies, including its capacity to create applications for multiple platforms, which lowers development expenses and shortens time to market. Furthermore, React Native provides outstanding efficiency, which is essential for programmes that need to process and deliver information in real-time.

"Building Mobile Applications using React Native: A Study of Performance and User Experience" by P. Singh et al. (2020). In this study, an application for mobile devices built with React Native is tested for efficiency and user satisfaction. According to the report, React Native is a strong framework for developing applications for smartphones since it provides exceptional performance and consumer experience. "Cross-Platform Mobile Development with React Native: A Case Study" by S. Arndt et al. (2020). A case investigation of a multi-platform smartphone application created with React Native is presented in this paper. The research sheds insight into the design process and emphasises React Native's advantages for creating applications that run across platforms.

"Development of a React Native-based Mobile Application for Online Grocery Shopping" by S. Lee et al. (2020). This piece examines the creation of a smartphone application for purchasing groceries online that is built on React Native. The research focuses insight into the design process and emphasises the advantages of utilising React Native while creating applications for e-commerce. "A Comparative Study of React Native and Native Mobile Application Development" by S. Hasan et al. (2019). The paper contrasts the creation of native mobile applications to React Native in terms of performance, user experience, and design. According to the report, React Native has a number of benefits, such as more rapid development and improved user interfaces.

These academic articles illustrate the advantages of utilising React Native for creating cross-platform mobile applications and offer insightful information on how React Native applications are developed.

Chapter - 3
System Development

3.1 Analysis

In this stage we gather and analyse, recognise, collect and evaluate the specifications and functionality for the application. Finding the application's key components, such as registration of new users in our application, signing-in the registered user, locating the current users pick-up point and delivery point, calculating the cost associated with the delivery, payment processing, and delivery tracking, is necessary for this.

Based on these requirements we made a system design that covers the structure, interface for users, and database structure based on the specifications. Make a design paper that specifies the system's technical specifications. We need to create the system with the aid of essential platforms and technologies, including React Native and Firebase (for backend services). Design the user interfaces, combine the back-end services, and put the system logic into practice. After implementation we have to run tests on our application. The platform's construction utilising React Native, Firebase, and other pertinent resources and structures is the main emphasis of the implementation phase. React Native components are used in the creation of the user interface, while CSS is used for styling. Firebase is used to create the backend services and offers features like cloud computing, database storage, and authentication. Programming languages like JavaScript or TypeScript are used to carry out the system logic.

Firebase is a popular backend-as-a-service (BaaS) platform Firebase offers numerous services for mobile and web applications, such as cloud storage, real-time databases, verification, and more. There are a number of benefits of utilising Firebase in React Native projects:

- **Easy integration :** straightforward and user-friendly integration: Firebase offers an integration for React Native that is easy and straightforward to use, with pre-built frameworks and extensions that make it simple to use its services. Developers can easily get started and rapidly incorporate Firebase into their React Native apps thanks to Firebase's robust guidelines and assistance.
- **Scalability :** Firebase is highly adaptable, which enables it to manage enormous amounts of information and traffic without experiencing any performance concerns. Because Firebase offers automatic scaling, it will automatically increase its resources to withstand the load as flow and data volumes rise, ensuring quick and dependable effectiveness for your React Native applications.
- **Real-time updates :** Data may be instantaneously updated throughout every device in real-time thanks to Firebase's real-time database abilities. Applications requiring real-time synchronisation or cooperation between numerous users, such chat programmes or collaborative editing tools, might benefit from this.
- **Analytics and testing :** To track and improve the efficiency of React Native applications, Firebase offers strong analytics and benchmarking capabilities. Firebase Analytics gives developers in-depth knowledge on user behaviour, enabling them to improve user experience and monitor important KPIs. Developers may test their React Native applications on a variety of devices using the testing tools provided by Firebase Test Lab, guaranteeing that they function properly on different platforms and setups.
- **Cloud storage :** Firebase offers services for storing and retrieving massive volumes of data, including audio, video, and picture files. Given its scalability, dependability, and security, Firebase storage is a solid option for applications that need to be able to transfer and upload large amounts of data. Incorporating a variety of strong services provided by Firebase into React Native apps. Firebase is simple and offers advantages including flexibility, real-time updates, strong

authentication, cloud storage, and extensive statistics and testing features.

Numerous advantages of using Firebase in React Native projects include simple integration, scalability, real-time updates, strong authentication, cloud storage, and potent analytics and testing tools.

3.2 Design

This phase involves designing a technological strategy for the item's delivery application as part of the design process. This comprises the database schema, user interface, and architecture. The system's backend services, APIs (Application Programming Interface), and user interface components are all described in the structure of the system plan along with how they interact. To assure accessibility and ease of use, the user interface layout should adhere to best practices and guidelines for design. The data model, relationships between entities, and data storage techniques should all be specified in the database architecture.

In React Native, "StyleSheet" is a styling tool used that combines JavaScript with CSS-like syntax. A quick and effective approach to specify style for React Native elements is through the use of StyleSheet.

- **Inline styling :** Similarly to how styles are applied to individual HTML elements, React Native components can have inline styling added to them. For instance, you may explicitly define the component's colour, size of the font, and additional parameters. As shown in Fig 3.1

```
<Text
  style={{ fontSize: 17, fontWeight: "500", marginRight: 10 }}
>
  {item.card.brand}
</Text>
```

Fig 3.1 Inline styling

- **StyleSheet.create()** : You may use the StyleSheet.create() to generate more intricate styles and reuse them across many components. The above function accepts an object as an input, where the style objects are represented by the values and the style names are the keys.

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#fff",
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

Fig. 3.2 styling using create function

- **Using styles in components :** After defining the aesthetics using StyleSheet.create, one can use the style prop to add them on to React Native components. The code for this is shown in Fig. 3.3

```
<TouchableOpacity
  onPress={() => handleAddItem()}
  style={CustomStyles.addNewCardButtonContainer}
>
  <Text style={CustomStyles.addNewCardButtonText}>Add Card</Text>
</TouchableOpacity>
```

Fig. 3.3 Styling using style components.

- **Cascading and inheritance :** Like to CSS, React Native allows for the cascading and inheritance of styles. By default, child components take on the parent component's styles, though one can change or add to them as necessary. Combining inline styling with StyleSheet.create, which offers a quick and easy method to specify and apply styles to React Native components, is how styling is done in React Native.

Designing of the project is distributed into certain levels. These levels are :

- Authentication
- Product and location selection
- Payment

Fig 3.4 represents the ER model of user authentication. If the user is new to this application he needs to sign up and if an individual is already registered then he has to simply sign in. In both the cases the user can only validate himself with the mobile number [3]. The user gets an OTP on their registered mobile number.

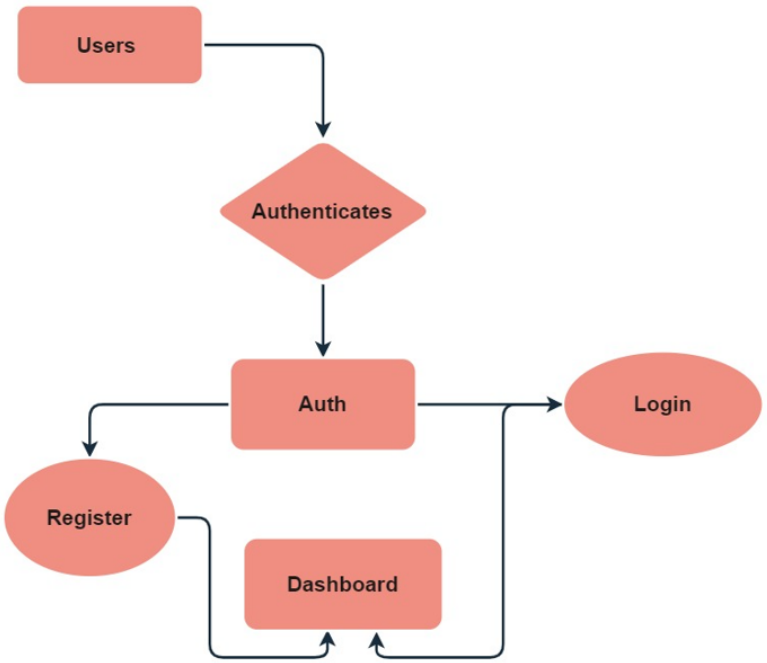


Fig. 3.4 ER diagram of user authentication

Product and Location Selection

Moving further in this application after signing-in the user will get an option of selecting the pickup and drop location for the product delivery. In this module of our application the user have to fill following details :

- Pick-up location
- Destination
- Details of the items which is to be delivered

- Some instructions for the delivery partner to deliver the product.

Fig 3.5 shows the flow of this process, first the user will select the pick-up point, as he clicks on this option a map will be displayed to select a location. For this map, I have used MapBox and MapView which are components of react native. This same step is followed in selecting a destination location. Then the user has to add the items he wants to be delivered. Users can also add some instructions for the delivery partner to deliver the items. It is a guideline for the delivery boy on how he should be reaching his destination.

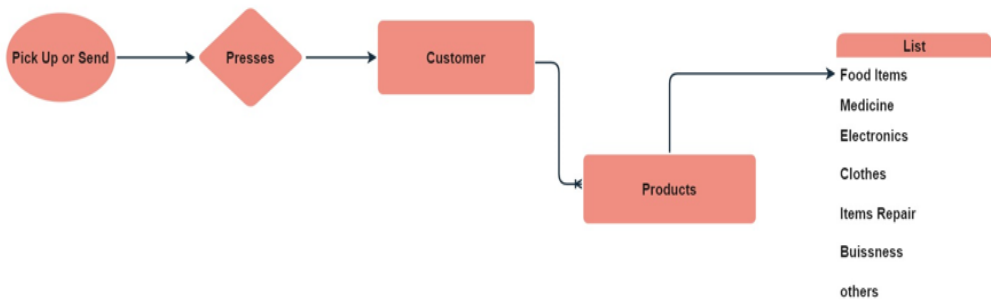


Fig. 3.5 ER diagram of product and location selection feature.

After selecting the source location and destination location the MapBox tool will find the appropriate and the fastest route to reach the destination and give us the distance between the two points, the price for the delivery is totally depending on the distance between the two points. As the user confirms the source and destination points he will be getting a detailed bill for the delivery and he will get a make payment page. Fig 3.6 shows the payment ER diagram. The user has the functionality to add a new card to make payment.

The user will be choosing each time from which card he wants to make a payment, if he wants to add a new card, a dialogue box will appear in the middle of the screen, where he needs to fill in the card details. There is a proper validation done in the front-end, if the card details are correct or not.

The card details whether the card exists or not, the card details match the existing card or not is done on the backend using Stripe, we will be discussing this technology in the further section of this chapter.

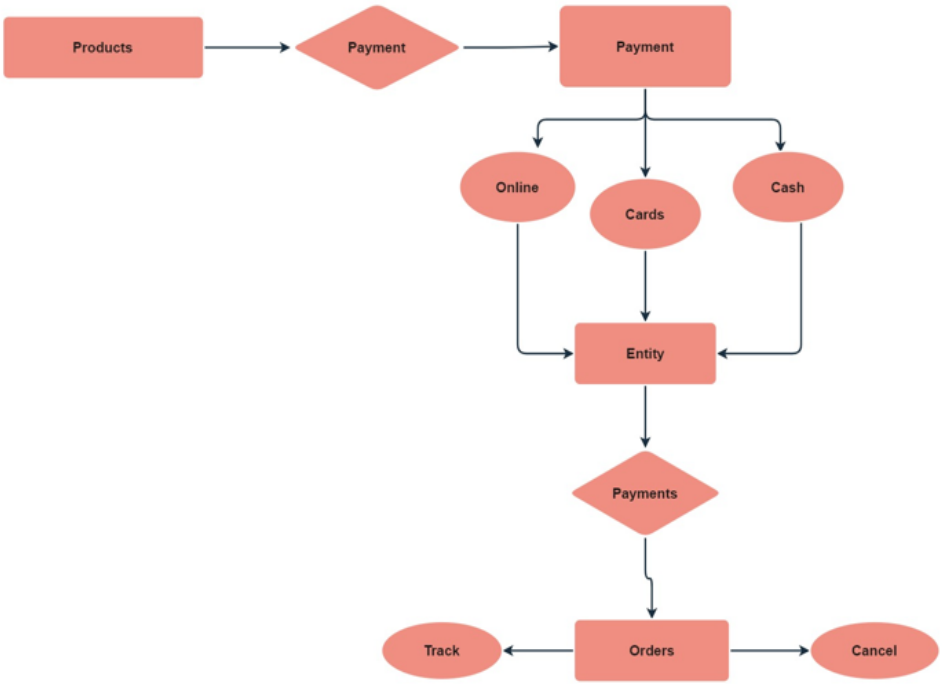


Fig 3.6 Showing the flow of payment methods

The user of this application will also get a number of features like, to see and edit profile, give rating and reviews to the delivery partner, order history and notification. As soon as the item reaches its destination, the user will get a notification about the delivery. The user is then able to rate the driver based on its behaviour and delivery timing and the user can also be able to write some reviews about the delivery partner. The user can also be able to see all the reviews and rating he has ever given to the delivery partner.

Each time when the user opens up the application he will be logged-in as a previous user or the last user. The logout feature will take the user back to the welcome page of the application and there, the user will be able to sign-in again with another mobile number or create a new user. The user should be

having the entered mobile number working to get an OTP on that particular number.

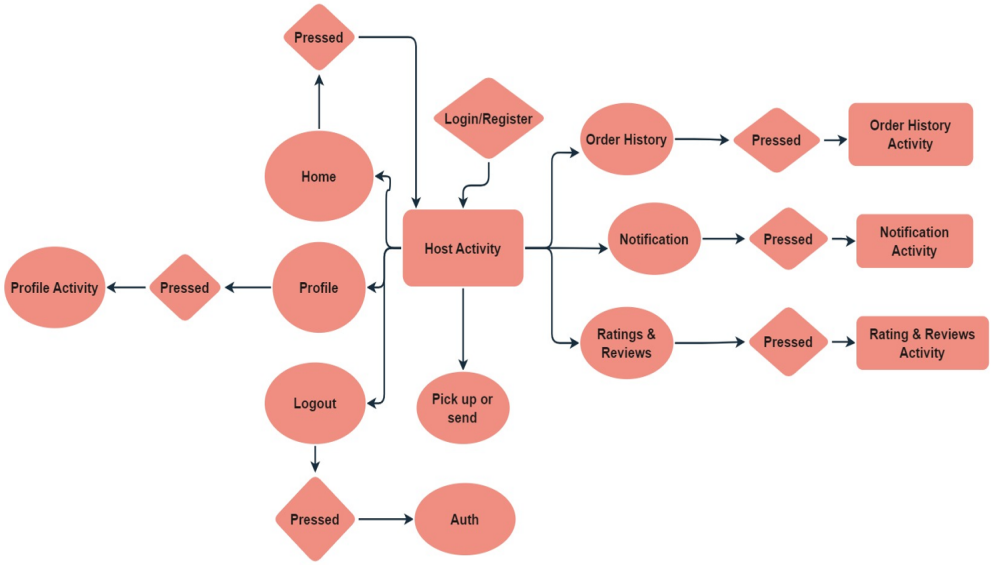


Fig 3.7 ER diagram of application

3.3 Development

This section focuses on thorough discussion of the building process of our application. It also contains the tools and technologies I have used, as well as the challenges I came across may all be found during the creation portion of a report on a React Native application. This contains the detailed information of the technologies used in this project.

- Expo
- Android Studio
- React Native
- Node.js
- Express.js
- Firebase

- Stripe

Now, we will discuss each and every technology in detail and how we have used it in our project from initialising our application to adding features to the application and use of these technologies in our application at various stages.

Android Studio

An Integrated Development Environment (IDE) called Android Studio has been employed to create applications for Android. Additionally, React Native, a well-liked framework for creating multi-platform mobile applications, is developed in this environment. The officially licensed IDE for Android, Android Studio, offers a number of features and tools designed to make the design and development process easier and more efficient. By employing one source code base, programmers can create native mobile apps for the iOS and Android platforms using the framework for JavaScript, also known as React Native[1, 2]. For designing React Native apps, Android Studio offers a strong development environment with tools for establishing a profile evaluation, and troubleshooting. To test and preview campaigns on various devices and operating systems, it also comes with a device emulator for Android. By means of an extension designated "React Native Tools," Android Studio supports React Native advancement. With the help of this plugin, the IDE gains support for a number of features and functions, including code emphasising and autocompletion, support for building and executing React Native projects, and support for developing, testing, and troubleshooting React Native applications.

Features of Android Studio for React Native:

- A built-in Android emulator is provided by this IDE to evaluate applications on various hardware and OS versions.

- Dynamic refreshing and hot reloading enables the developers to rapidly view modifications to their code without having to recompile the app every time.
- Dynamic refreshing enables the developers to rapidly view modifications to their code without having to recompile the app every time.
- For React Native elements and APIs, code automatic completion and syntax-highlighting features are provided. Tools for app assessment that analyse effectiveness and highlight areas for improvement assistance with creating native Android components which can be used into React Native applications.

React native tool plugin. React Native development-specific capabilities are added via the React Native Toolkit extension for Android Studio this is also provided by Visual Studio Code, in this also we can add-on extra plugins.

- Newly installed React Native applications are frequently made available right from the IDE.
- Resources for debugging React Native programmes to find and correct bugs.
- React Native CLI integration enables programmers to execute and create applications right inside Android Studio.
- It is possible to skip to component descriptions as well as the completion of code for React Native widgets and APIs.
- a comfortable environment for Android developers to work in a strong IDE with a wide range of capabilities and resources for developing apps. A variety of smartphones with Android and iOS and versions that may be utilised to simulate and test applications.
- Assistance with integrating native Android modules into React Native applications with Git and other version management systems that is efficient.

In addition to offering extensive assistance for React Native growth and development, Android Studio offers a strong and effective IDE for creating

Android applications. The IDE provides a number of features and instrumentation that can speed up the development procedure and assist programmers in creating high-calibre, mobile applications that work across platforms. Android Studio is a popular option among programmers for creating React Native applications because of its durable features and simplicity of use.

ExpoGo

Expo is a well-liked and potent suite of development and deployment tools for React Native applications. It offers a simplified method of development that removes the majority of sophistication and framework-specific requirements related to creating and distributing applications for mobile devices. Expo saves designers from bothering about infrastructure support so they are able to focus on creating interesting and exceptional applications. Expo's ability to support software development across platforms is one of its key advantages. Expo eliminates the distinctions between the Android and iOS operating systems so that programmers are able to develop programmes that work effectively on both operating systems. This is made possible by using a uniform API to access aspects of the device including the camera, push notifications, and connections.

The ease of setup is another important aspect of Expo. With a few straightforward keystrokes, developers can rapidly build up an entirely novel project, and Expo will handle the rest. Because there is no longer a need for difficult setting up and configuring procedures, programmers can easily begin developing in React Native. Additionally, Expo offers a variety of already assembled UI elements that are able to be quickly altered to meet the demands of the app you are developing. These parts include things like buttons, text inputs, and picture views, among others. Additionally, Expo offers a collection of APIs that can be used to access native device features like the camera, push notifications, and contacts in order to create complex and interesting mobile applications.

Expo can only be used with react native which is also made available by Facebook in the year 2015. But to work with React Native we first need to understand what Reactjs is, how the components in Reactjs are rendered. Concept of Hooks is also very important to understand the SPA (Single page application) mechanism of Reactjs. Developers describe what they want to happen, and React manages the how, thanks to the use of declarative programming, which is one of React's major characteristics. Because programmers can concentrate on the application logic without being concerned about its execution specifics, writing and maintaining algorithms is made simpler to do so. React is renowned for its capacity to manage significant volumes of data and change the user interface in instantaneous fashion without the need for a page reload.

React is incredibly effective and quick since it renders components using a virtual DOM (Document Object Model). React may reduce the amount of UI changes needed by using the simulation of the DOM, which is a thin approximation of the actual document object model (DOM). Components that we create in React can be of two types: Class based components and Function based components, the benefit of using function based components is that we can use Hooks with them.

Function based components

Function components must return the React elements that make up the component's Appearance and are defined using a standard JavaScript function. They accept a configurable 'props' object as the argument they are given, which holds any information that the component's parent passes to it. Function components lack state by default, but they can add state with the 'useState()' hook. Any changes to state cause the component to be re-rendered because state is an object that holds data that is particular to the component. The majority of new React code repositories only employ functional components, which are currently the suggested method for defining components in React. Class-based components continue to be supported by React, and they can be

helpful in some circumstances, such as when connecting with libraries from outside the framework that need them or dealing with old code. The useEffect() function, that is called after the function component has been displayed the first time around and may be used to carry out side effects like getting information from a server or changing the document title, is one of the additional hooks that function components have access to. Function elements are quicker to speculate about and test since they are more portable and less complex than class-based elements. Since they lack the extra complexity of a JavaScript scripting class, they are also far more performant. Functional components supports following hooks:

- useState Hook
- useEffect Hook
- useRef Hook
- useCallback Hook
- useMemo Hook
- useContext Hook
- useReducer Hook

Class-Based Components

Using a JavaScript class to define a component is known as class-based components. Prior to the introduction of component functions in React 16.8, they were the main method of constructing components in React. Class-based components have to inherit from the 'React.Component' class and be declared using the 'class' keyword. They contain a function called "render()" that delivers the React fragments that comprise up the UI of the component. State is likewise a feature of class-based sections, and it is controlled via the'setState()' function. Any changes to state cause the component to be re-rendered because state is an object that holds data that is particular to the component. Fig 3.8 and Fig 3.9 shows the syntactical differences between

functional and class components. Additionally, decoration, organisation, and interaction with events are supported by JSX in React Native. To style your widgets, you may use inline styling or additional stylesheets, and you can utilise arrangement aspects like "FlexBox" to manage how your UI is laid out. Your elements can also include event handlers to react to user events, such as 'onPress' for a button's press.

```
import React from 'react';
import { View, Text } from 'react-native';

function MyComponent() {
  return (
    <View>
      <Text>Hello, world!</Text>
    </View>
  );
}
```

Fig 3.8 Function component of Reactjs

```
import React, { Component } from 'react';
import { View, Text } from 'react-native';

class MyComponent extends Component {
  render() {
    return (
      <View>
        <Text>Hello, world!</Text>
      </View>
    );
  }
}
```

Fig 3.9 Function component of Reactjs

Lifespan methods, or methods that are invoked at particular times during the component's lifespan, are likewise included in class-based components. ComponentWillUnmount(), which is called right before the widget is removed from the DOM, and componentDidMount(), which is called after a component has been displayed for the first time, are examples of these operations.

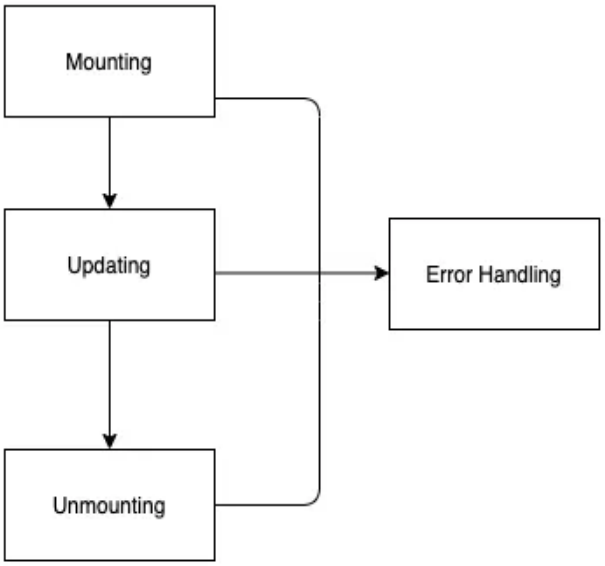


Fig. 3.8 Complete cycle of Expo

Fig 3.8 shows the Expo's real-time reloading function which is a significant additional advantage of using Expo. This eliminates the requirement to compile over again or rebuild the programme and enables developers to observe changes to their code in real-time. This may help your development process move along much more quickly and make it simpler to continue working on the application itself. Additionally, Expo has a function called over-the-air updates that enables programmers to push improvements to their applications without having to wait for users to install the latest version via the app store[5]. This makes it simple to update the application with fresh capabilities and address issues without affecting the user experience.

- **Cross-platform development:** By separating away platform-specific aspects and offering a standard API for gaining utilisation of device functionalities, Expo allows it to be simpler to design programmes that work on both the iOS and Android platforms.

- **Simplified setup** : Expo makes it unnecessary to do difficult setup and configuration procedures, making it simple for developers to begin working with React Native.
- **Expo's hot reloading:** Expo's hot-reloading feature enables developers to view updates made to the code in real-time without requiring them to recompile or recreate the entire application.
- **Pre-built components:** Expo comes with a number of pre-built user interface (UI) elements that may be readily modified to meet the requirements of your application. Examples of these include <Button> ,<Text>, <Image>, <TextInput> , e.t.c..

Node.js

A JavaScript runtime called Node.js was constructed on top of the V8 JavaScript engine. Since it enables programmers to run JavaScript code externally from an internet browser, it is ideal for developing applications that run on the server. Due to its event-driven, autonomous I/O approach, Node.js is renowned for its ability to manage several simultaneous links without causing delays for other applications. As a result, it is very versatile and ideal for developing applications that operate in real-time. Node.js is intended to be lightweight as well as quick, so it is able to run on affordable hardware while handling a lot of information without experiencing any kind of delay. A significant and enthusiastic programmer community produces and maintains a broad range of library components and modules that may be quickly added to Node.js programmes to add new features and functionalities.

The Node.js reinforces both relational and non-relational databases, which makes it simple to select the best database for your application based on your particular demands and specifications. Developers may quickly install, manage and update independent libraries and modules using the built-in package manager npm (Node Package Manager). Being a publicly available programme, Node.js codebase is readily available to everyone for viewing,

editing, and contribution. Its comprehensive documentation provides a simple way for developers to understand and utilise the platform, and those who have become familiar with JavaScript will find it very simple to pick up.

In this project Node.js is used for building the complete back-end of this project. The features of Node.js for choosing it over other technologies are as follows:

- **JavaScript Runtime:** Using the framework of Node programmers are able to execute the code that uses JavaScript independently of a web browser. As a result, programmers can apply JavaScript to create the server-side portion apps, command-line tools, and other kinds of software.
- **Event-Driven Architecture:** Node.js's event-driven, autonomous I/O approach enables it to manage many concurrent connections without delaying other requests. Because of this, Node.js is very extensible and ideally suited for creating applications that operate in real time, such as chat, gaming, and other kinds of apps that need data to be processed immediately.
- **Support for Multiple Databases:** Both relational and non-relational databases are supported by Node.js. This makes it simple to select the ideal database for your application in accordance with your unique demands and specifications.
- **Simple to Learn:** For programmers who are already comfortable with JavaScript, for them Node.js is rather simple to learn and implement. This implies that developers won't need to learn an entirely novel programming language or environment in order to get started immediately developing server-side apps leveraging Node.js.
- Node.js was developed on top of the V8 JavaScript engine, that has been substantially optimised for speed and is compact. Node.js is hence able to operate quickly and consume less of the system's resources compared to other server-side platforms. Asynchronous

programming, or asynchronous programming, is another feature of Node.js that enables programmers to create quick-running code.

- **Wide-ranging Modules:** Wide range of modules and packages are created and maintained by a huge and active community: Node.js has a tremendous and engaged community of contributors. To offer additional capabilities and features, these extensions may be readily incorporated into applications written in Node.js.

Express.js

Express.js serves as a quick, lean web framework based on Node.js that offers a number of advantageous features for creating APIs and online apps. It is based on Node.js and offers a simple API that makes the procedure of creating web and mobile apps easier. The server-side functions offered by Express.js may be utilised to carry out a variety of activities, including processing requests that come in, authorising users, and managing exceptions. Complex request-response flows may be made by connecting these services provided by middleware in a pipeline.

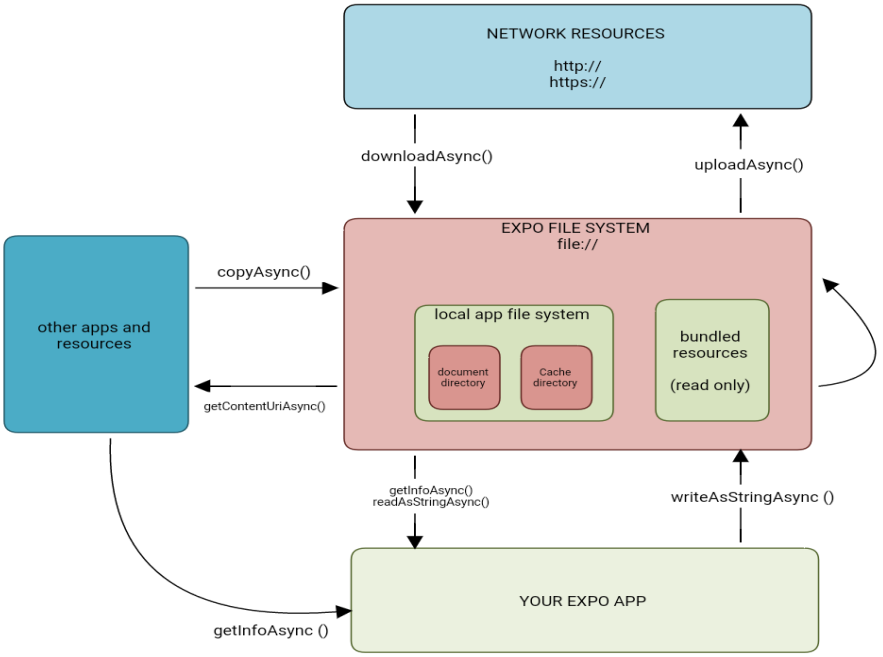


Fig. 3.9 Flow of requests and response from client to server

Express.js is frequently used to create APIs, or interfaces for application programming, which enable interaction between different parts of a programme. A common method for software to communicate data and services is provided by an API, which facilitates the development of sophisticated software platforms. Express.js offers a straightforward and understandable API that makes creating APIs simple. Developers may create functions that handle GET, POST, PUT, and DELETE requests by defining routes that correlate to the various HTTP methods. A variety of intermediate operations, including processing receiving JSON data, authorising users, and managing errors, are available in Express.js. A RESTful API, which is an architectural framework for developing APIs that adheres to a set of restrictions, may also be built using Express.js. RESTful APIs modify resources (like data objects) while offering replies in an accepted format (like JSON) via HTTP methods.

Security and scalability are crucial factors to take into account while developing an API with Express.js. Data in transit may be made more secure with the support for SSL/TLS encryption that Express.js offers. Additionally, it encourages rate limiting along with other attack-prevention strategies. Benefits of using Express.js for building APIs in React Native project:

- **Scalability:** Express.js is very scalable and can handle many concurrent connections without experiencing any lag. It can be set up on a group of servers or a platform that uses the cloud, like AWS or Google Cloud.
- **Security:** Express.js supports SSL/TLS encryption, which helps to protect data while it is being transmitted. Additionally, it encourages limitation of rate and other attack-prevention strategies.
- **Large Community:** Substantial and prominent developer community: Express.js has a sizable and active developer ecosystem that builds and maintains a variety of modules and libraries that are readily

incorporated into Express.js programmes to bring additional features and possibilities. Additionally, the platform has excellent documentation that makes it simple for developers to comprehend and use.

Google Firebase

Google has created a platform called Firebase that offers an array of tools for developing and expanding mobile and web applications. Due to this it provides a variety of services and tools that can enable development and quicken time-to-market, it is a well-liked option for developers using React Native. Fig 3.9 shows the list of features provided by the firebase in-built section.

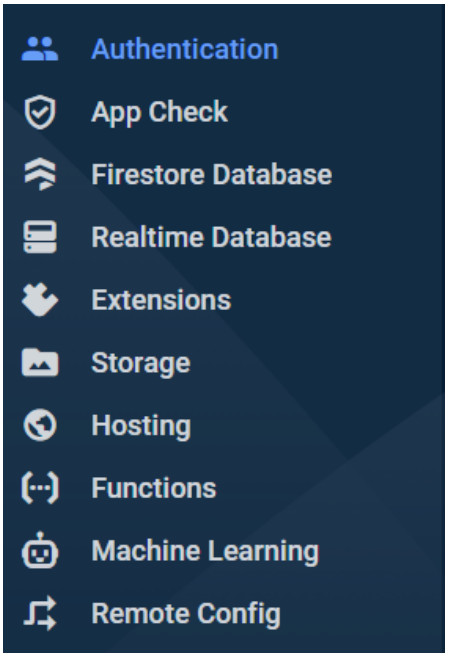


Fig 3.10 List of Firebase features

Key details regarding Firebase for React Native applications is provided below:

1. A cloud-hosted database called Firebase Realtime Database may be employed to save and sync data instantaneously. It offers a NoSQL database, making it simple to handle and maintain information in an open to change, scalable fashion[3, 4]. Applications that adapt in real time to data changes may be created using the Firebase Live Database, making them more interactive and appealing for users.

2. A quick and easy approach to add authentication to your React Native application is with Google Authentication. A variety of authentication techniques are supported, including social media, phone, email and password, and more[6]. You may create safe apps that need authentication from users and permission with Firebase Identification. In this project we are working only on one kind of authentication and that is phone number authentication. It generates OTP and sends it on the entered mobile number. This OTP has a variable life span for more security.

3. User-generated material with a large size, such photographs and videos, may be easily stored and served with Firebase cloud-based storage. It allows for both local and remote storage, allowing access to material from any location in the globe. For a full storage solution for your React Native application, Firebase Cloud Storage interfaces with other Firebase services like Authorization and Dynamic Database.

4. Google Firebase also provides a feature called FCM (Firebase cloud messaging). Delivering messages to clients on Android, iOS, and the web is made possible by the dependable and scalable Firebase cloud-based messaging service. It offers a variety of capabilities, including targeting, planning, and statistical analysis, making it simple to give people timely and relevant communications. It is a popular option for developers using React Native due to its simplicity of use, scalability, and interaction with other Google services.

Stripe for Payment

A service for handling payments called Stripe makes it possible for companies of any kind to receive and handle payments online. It delivers a variety of services and applications that enable businesses to make payments easily and effectively[5]. Many companies, which include entrepreneurs, small enterprises, and large corporations, use Stripe on a global scale[12]. Stripe works for all kinds of payments starting from card payment to UPI payments, card payments also includes all kinds of cards, i.e. master-card, visa-card, in debit card and credit card options. Some key features of Stripe are given below:

- **Payment methods :** Online payment processing is made simple and safe by Stripe. Numerous payment options, such as debit and credit card transactions, and payments via mobile devices are supported. All aspects of payment processing, such as identifying fraudulent transactions, reimbursements, and currency conversions, are handled by Stripe. Additionally, it supports memberships and periodic payments, making it simple for businesses to handle customers' payments over time.

- **Security :** Stripe prioritises safety and delivers a variety of tools to support organisations in keeping their transactions safe. It holds a PCI Level 1 Service Provider authorization, which is the highest kind of validation available to payment processors. Additionally, Stripe's platform offers fraud identification and avoidance tools like multiple-factor authentication, real-time risk assessment, and machine learning algorithms.

- **Global Reach :** With around 135 recognised denominations and over 40 operating nations, Stripe makes it simple for business establishments to take payments through clients all around the globe.

International payments are handled entirely by Stripe, involving conversions of currencies and observance of regional laws.

- **Billing :** A set of software programmes called invoicing makes it simple for companies to handle membership and payments that recur. It offers capabilities for creating price structures, maintaining users and payments, and responding to unsuccessful payments. To offer a complete billing solution, Stripe Billing connects with other Stripe services like Payment Processing and Hawkeye.

Organisations can handle and accept payments via the internet with ease thanks to a variety of products and services offered by Stripe. Stripe offers a complete platform for managing payments in your React Native application with payment processing, developer tools, security, global reach, billing purposes, and detection of fraudulent transactions solutions. Businesses across all kinds choose it because of its versatility, protection, and simplicity of use.

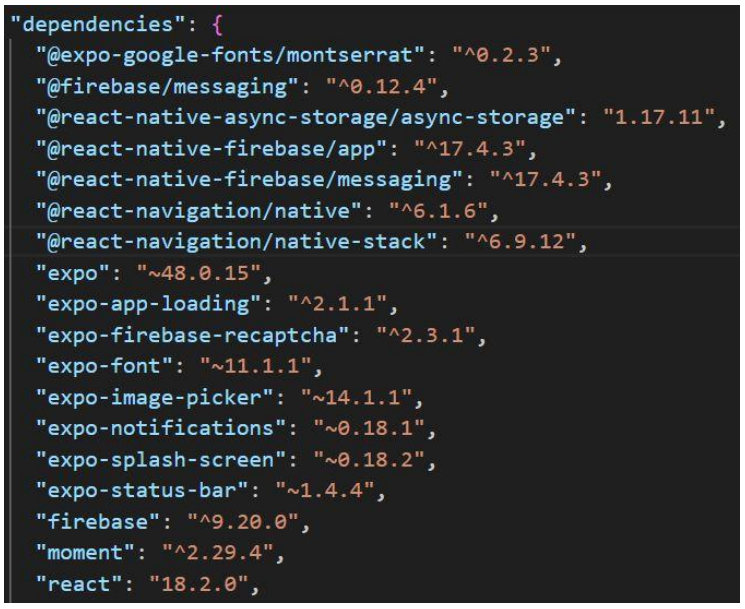


Fig. 3.11 Dependencies included in this project

Chapter - 4

Experiments and Result Analysis

Firstly we initialised our React Native project inside a desired folder, For initialising React Native application we need to have node and yarn installed on our system. After installing node and yarn we need to install react-native-cli globally on our system[14, 13]. Installing react native cli globally will add this package automatically into our application whenever we initialise a new React Native project. Fig 4.1 shows the code for globally installing the react-native-cli.

```
npm install -g expo-cli
expo init AwesomeProject

cd AwesomeProject
expo start
```

Fig 4.1 Commands for installing cli and initialising app

The latest version of node, npm and yarn are used in the development of this project to keep the requirements and functionality of this project. Fig 4.2 shows the version of the same.

```
C:\Users\Abhishek Mishra>node --version
v16.13.2

C:\Users\Abhishek Mishra>npm --version
8.1.2

C:\Users\Abhishek Mishra>yarn --version
1.22.19
```

Fig 4.2 Versions of technologies used

There are several modules and pages in this project starting from the Welcome Screen we have and many, numerous screens or "views" that collectively make up your application's user interface are typical. Each screen serves as an

individual component of your mobile application and may have various features and functionalities. Table 4.1 shows the list of screens given below.

S. No.	Screen Title	Functionality
1.	Welcome Screen	It gives the user an option to register or an existing user to sign-in to their respective accounts.
2.	Login Screen	In this screen the user needs to enter the registered mobile number and press submit and he will move to the next screen.
3.	Register Screen	If the user is not registered (new user), he has to enter a mobile number to get himself registered.
4.	Verification Screen	In this screen there is a user input of 6 digits in which the user needs to enter the OTP which he got via SMS on the entered mobile number.
5.	Home Screen	In this screen the user can press on “Modal” for different functionalities, here he gets the button to set pick up and destination locations.
6.	Task Screen	In this screen the user needs to enter the destination and source locations along with the order that needs to be delivered.
7.	Confirmation Screen	This screen is for rechecking the source and destination location and confirming the address.
8.	Order Details Screen	This screen shows the user the details he has filled in the previous pages along with the delivery charges and proper billing details.
9.	Payment Screen	If the user confirms the details he is then landed on the payment screen where he can add a new card or make payment with already added cards just by filling necessary card details.
10.	Track Order Screen	The user is then able to see the real-time

		tracking of the delivery partner on a Map.
11.	Feedback Screen	After the item gets delivered the user of the application is then able to rate and write some reviews about the delivery partner.
12.	Edit Profile Screen	This is a feature screen of this application where the user can edit its profile except for his mobile number.
13.	Notification Screen	This screen shows the notification of the items that have been delivered or picked up.
14.	Chat Screen	The user will be able to chat with the admin and communicate about his problems or any other delivery issues.

Table 4.1 List of screens

User Authentication using Firebase

Google Firebase offers us a number of options for authenticating the user providing a guaranteed security to the user. In this application I have only used mobile authentication because of easy implementation, better security as well as it provides better user experience. The usage of the mobile OTP in Google Firebase offers mobile applications a safe and practical authenticating technique that may boost user experience while also lowering the danger of credential fraud. This generates a six-digits OTP and sends it to the entered and existing mobile number and expires after 5 minutes. After initialising a new project in google firebase and choosing the platform, Google Firebase provides us configuration content which contains following fields as shown in Fig 4.3

1. API - Key
2. Auth Domain

3. Project Id
4. Storage Bucket
5. Messaging SenderId
6. Application Id

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyCVr12d0Fr5y6ViJRNIn3tWNsTFumQzHII",
  authDomain: "testing-9ec39.firebaseapp.com",
  projectId: "testing-9ec39",
  storageBucket: "testing-9ec39.appspot.com",
  messagingSenderId: "1017245611535",
  appId: "1:1017245611535:web:1aa3b2ed1e32b07f00d57a"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
```

Fig 4.3 Config file generated by firebase

After the user login to their respective accounts the most important and challenging task is to keep them logged-in until they logout from the application. In any application if we are not preserving the Auth state of the user, the individual will automatically get logged out of the application. To achieve this we have to keep the user Auth preserved and pass it on throughout the application [5].

Fig 4.4 shows how the application's Login and Registration screen will look like. Functionality of the login screen is ,it gives the user an option to register or an existing user to sign-in to their respective accounts. The Function of Register screen is, if the user is not registered (new user), he has to enter a mobile number to get himself registered. This is generating a One Time Password sending through firebase and sending this to the user's phone to authenticate the user. The same step is being applied to both, first time registering people as well as already registered accounts.

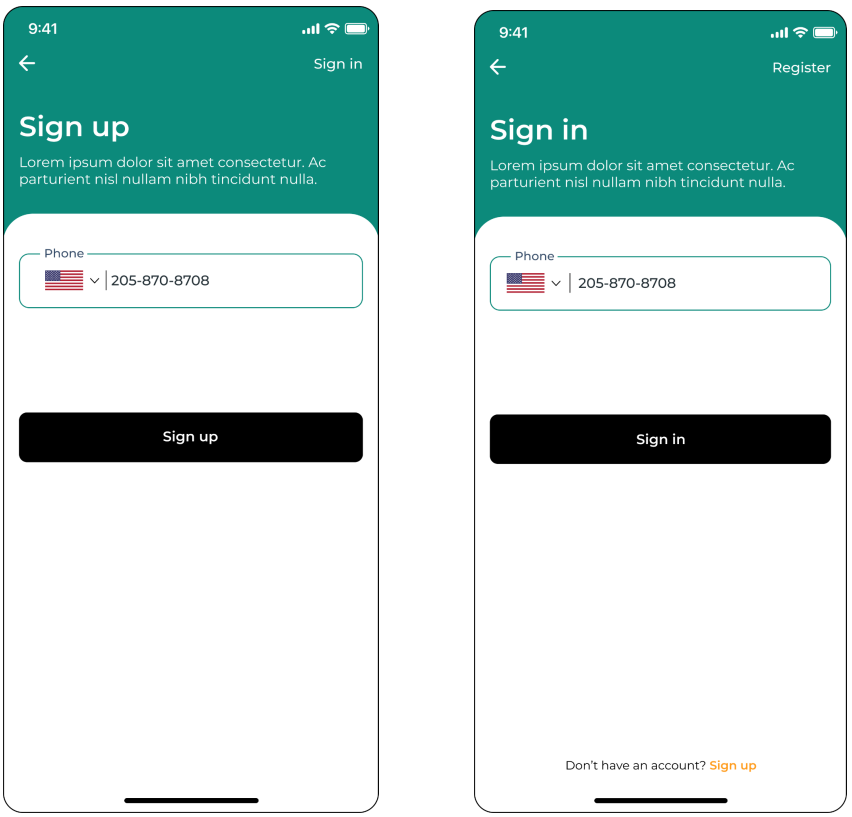


Fig 4.4 Login and Register screens of application

This above challenge can be done by four different methods : first one is prop drilling in which we will be sending the Auth of a particular user to different components, second is by using Context API and making all the fields of the user configuration a global variable and the third method is by using Redux Toolkit. In this application I have done this by prop drilling and using context API. Another method to achieve our aim is to use Async-Storage provided by React Native.

It's crucial to keep in mind that AsyncStorage has several drawbacks, like its restricted capacity for storage and incapacity for handling massive volumes of data.

AsyncStorage isn't intended to be used as a reliable archive solution, thus developers should be mindful of this and store private information using stronger storage alternatives.

```
import { getAuth, onAuthStateChanged } from "firebase/auth";

const auth = getAuth();
onAuthStateChanged(auth, (user) => {
  if (user) {
    // User is signed in, see docs for a list of available properties
    // https://firebase.google.com/docs/reference/js/firebase.User
    const uid = user.uid;
    // ...
  } else {
    // User is signed out
    // ...
  }
});
```

Fig 4.5 Code for persisting the user

Dashboard and Modal

A modal is a React Native component that enables programmers to build pop-up dialogues, notifications, alerts, and confirmations by displaying an element or material above the currently shown screen. Modals are frequently used to request input from individuals, display important data, or request confirmation of operations. Fig 4.3 shows the Home Screen and a “Modal” opening over the screen showing the number of features to the user of this application as mentioned in table 4.1. This “Modal” is a react native component which provides us with an animation to a particular view. When a user clicks on the menu icon this “Modal View” will make itself visible. It also provides a number of props which helps us make our application more effective, user friendly and more responsive. For e.g. onRequestClose() function provided by this lets the user dismiss the modal when he clicks the back button of his device.

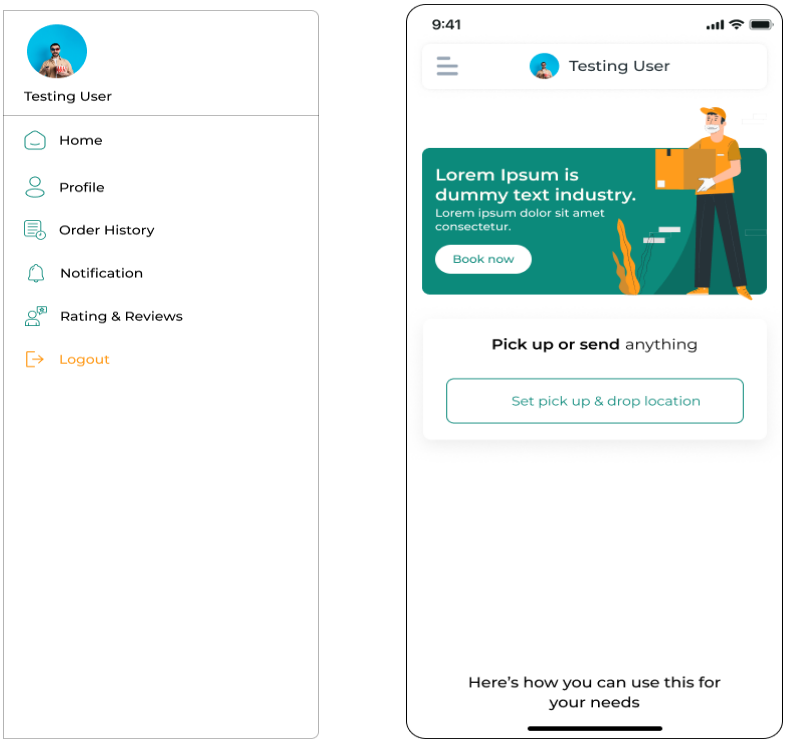


Fig 4.6 Home screen of application

Google Maps

I have integrated Google Maps in my project, the user can select the pick-up location and he will see the auto-fill feature with the help of google maps, with the help of auto-fill feature the user can better locate the pick-up and destination points. Moreover he is able to set a marker on the locations and also get to pin-oint the current location. It gives users access to a wide range of features and capabilities, including interactive maps, geotagging, directions, and locations. When the user is on the Order Tracking screen after making the payment he would be able to see the direction of the delivery person right on the screen with the help of Google Maps API[3, 4].

Other options for integrating maps into our application is MapBox and MapView .The React Native's react-native-maps library's MapView component enables us to include maps GUI in this project. To track down the path of the delivery partner I am taking the latitude and longitude of the delivery person, each time the latitude and longitude changes we update the

position in our array, we have put a logo on the map as the rider moves the latitude and longitude changes so as the image of bike shown in the map[10].

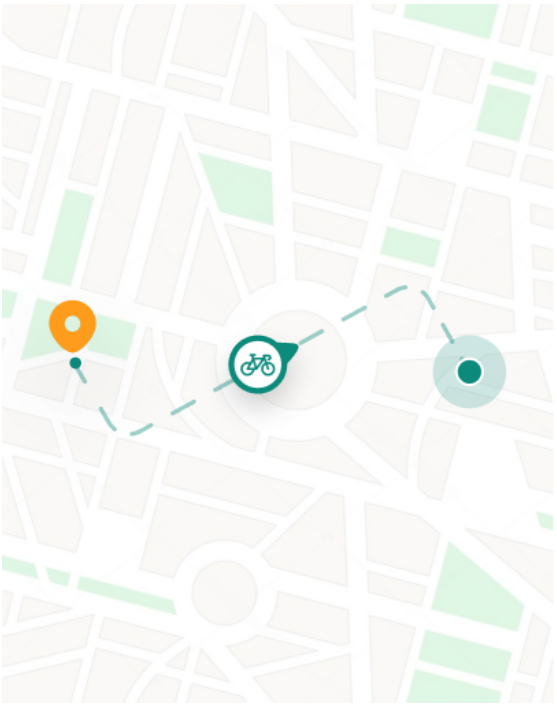


Fig 4.7 Order tracking using MapView

Order Details Screen

Once the user confirms the pickup and drop location and presses on the "Submit" button he will be taken to the next screen where he would be getting the order details, here by clicking on the map marker he can change the pick up and drop location. As shown in fig 4.7, he can also add some instructions that he wants to tell the delivery driver. The user of this application will get a detailed screen of the invoice. The fare for delivering the item from one place to another totally upon the distance between the two points. The distance between the pickup and drop location can be calculated with the help of Google API which will first fetch the latitude and longitude of the source and destination and return the fastest route between these two locations.

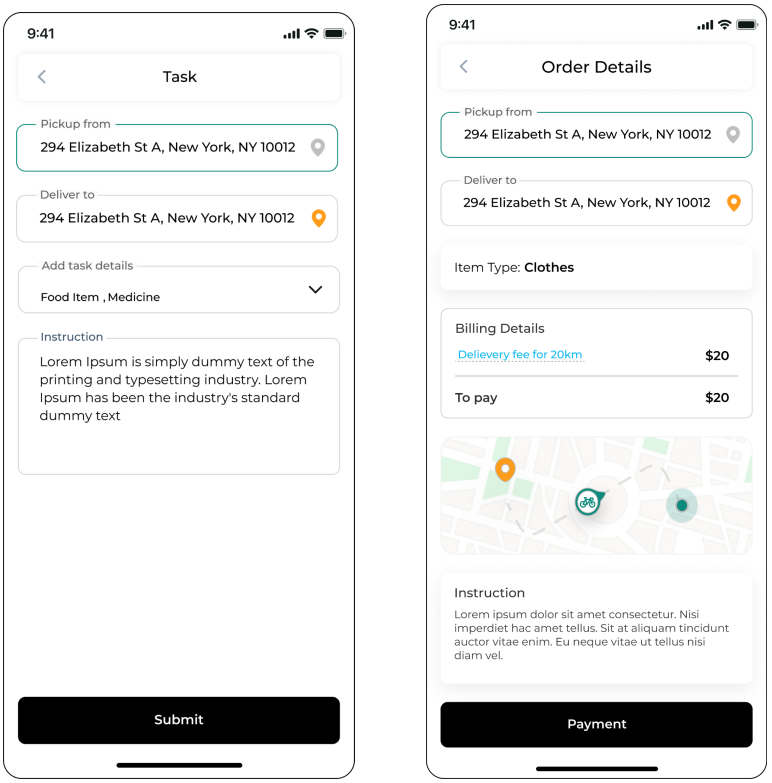


Fig 4.8 Order details screen

All the order and user related information is getting stored on the backend of the application. If a new user creates a new account all his details will be saved at the servers. If the user makes any changes or edit his profile it gets saved at the backend of the application. We are fetching all that information and displaying it at the front-end of the application. For fetching this we have a built-in function called “Fetch” provided by React Native.

```
const Notifications = () => {
  const [data, setData] = useState();
  const getArticles = async () => {
    try {
      const response = await fetch(
        "http://192.168.1.35:3000/notification/list_notifications"
      );
      const json = await response.json();
      setData(json.notifications);
    } catch (error) {
      console.error(error);
    }
  };
```

Fig 4.9 Syntax of fetch function to GET from API

The response object that is returned by the fetch request function resolves a Promise. Response data, including the content of the response, its status, and headers, is stored in the Response object. The data may then be extracted from the response using techniques like.json(),.text(), and.blob().

Online Payment

Leading payment service Stripe enables entities to take transactions online. An array of processes are taken in the back end when an end user makes an online payment on a mobile application or website that makes use of Stripe. Fig 4.9 shows what happens at the backend when somebody initiates a transaction.

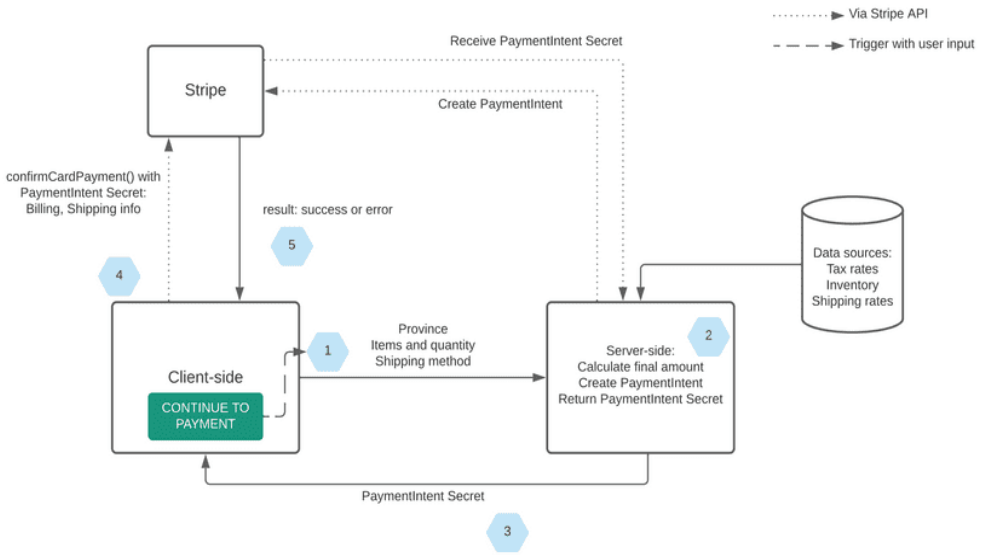


Fig 4.10 Backend of user initiated online payment

In my project I have made only card payment options to make payments online using Stipe. To initiate a payment, the new user first needs to add a card to the application to make payment. This card payment can be done via credit card or debit card. The user has to fill out the following information.

- Card Number ,Card Holder's name, CVV, Expiry Date
- This Information get validated first on the front-end
- If the information is validated by the Stipe.

Fig 4.10 shows that when the user clicks on the "Add New Card" a dialogue box opens up and shows the card details that are needed to be entered by the user to add a new card for making payment.

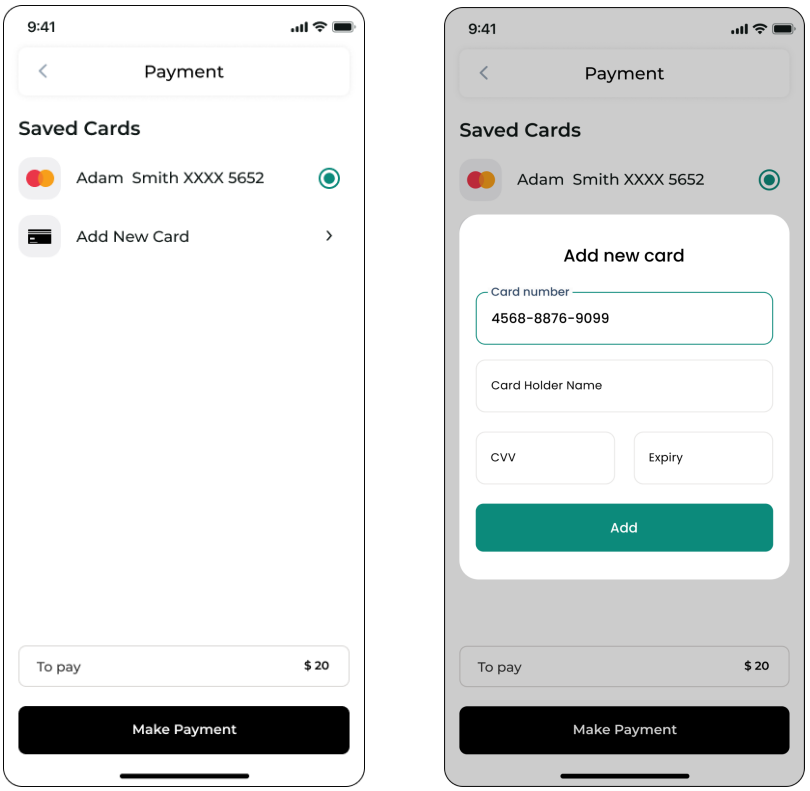


Fig 4.11 Payment Screen of application

After making the payment the user is done with his side of the application, now, when a delivery partner accepts the request to pick up the stuff he will get all the information regarding the item that is to be delivered with the pickup and drop location with this.

Chapter - 5
Conclusion

5.1 Conclusion

The building of a fully functioning full-stack application is developed at the best level which is resolving a lot of problems that are not available in the application similar to this. It can also become a better alternative as it would be providing a customer service, chat option, live tracking and a lot of options that are not provided by other leading applications.

This application was integrated with the backend and we are able to get and fetch the responses from there, every response, rejection and error was handled in such a way that it didn't slow down the application. In this report, we've addressed the main components and functionality of this product delivery application, such as push alerts, tracking in real time, and how to make payments. The advantages of utilising React Native for the development of applications, including code reuse, quicker development cycles, and a big designer community as a whole have also been emphasised.

This product delivery application that uses React Native is an outstanding instance of how people can use smartphone and tablet technology to streamline processes, boost customer happiness, and ultimately improve revenue. We can anticipate seeing more cutting-edge applications created utilising React Native and other advanced technologies as the mobile environment stays ongoing to change. Through this application the users whether the customer or the driver both would be able to add their images for better experience and recognition. For this we have used "ImagePicker", another powerful library supported by react native. This library helps us in adding an image through our device. For doing any of these above mentioned functionality Android and iOS devices require a permission to do so, this makes the application more secure for the user.

Google Firebase where the users are stored is also counting the number of reads and writes we are making each day. As shown in Fig 4.1 and Fig .2 , we can see the count of reads and writes. The dotted lines also show the average number of reads and writes per week.

Fig 5.1 Reads per day underdeveloped.

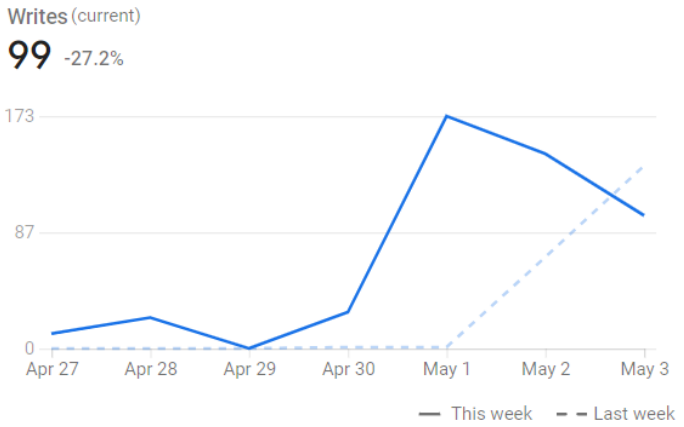


Fig 5.2 Writes per day underdeveloped

At the initial stage we started to work on the dummy data and in fig 5.3 that the user is getting stored at the backend and is handled very efficiently. The delivery partner is getting stored at the backend of the application along with

all their details, how many orders they have ever got out of which how many of them were successfully delivered.

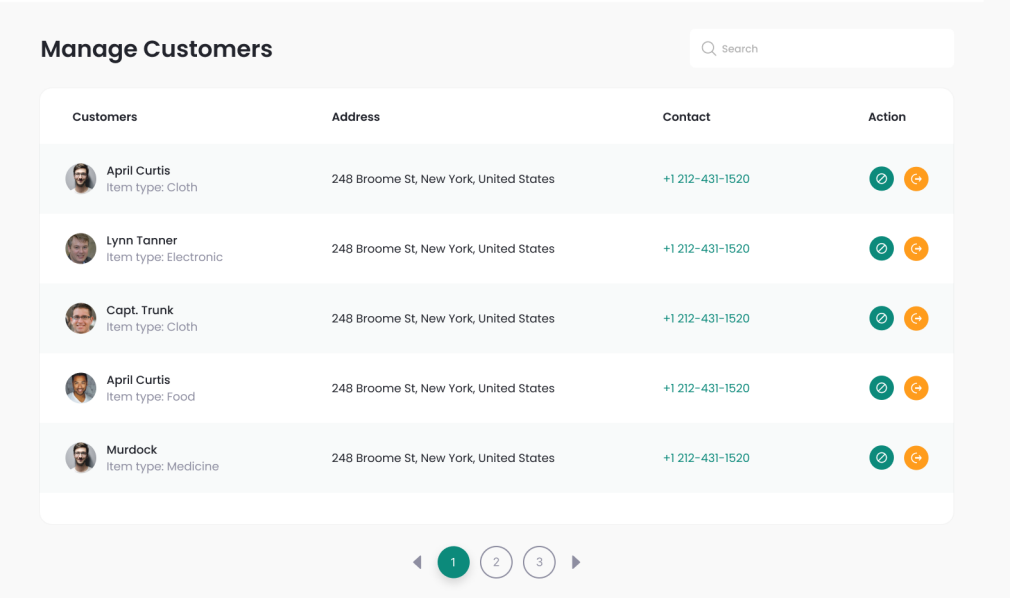


Fig 5.3 Dashboard of admin-side

5.2 Future Scope

This project can be made more sophisticated and we can add more functionalities to this application. The list of the functionalities which can be added to make our application more efficient and user friendly are given below:

- **More Payment Options :** As of now I have only added card payment methods (credit card/ debit card) in my application. But further, we can add google play, paytm and other UPI options for making payment.

- **Tutorial Supports :** We can add an animation-based intro for the first time user in the application. This service is also available on any google application to new users.

- **More login options :** The user of the application can only sign-in only using a phone number, but in future we will integrate email login with phone number, so that the user can have more options to sign-in.

- **More transport options :** we can add more vehicle options based on the kind of product we want to deliver, for more heavy goods and items the user will have an option of booking a bike or mini-truck.

REFERENCES

[1] *Setting up the development environment · REACT NATIVE* (2023) *React Native RSS*. Available at: https://reactnative.dev/docs/environment-setup.

[2] *Running on device · REACT NATIVE* (2023) *React Native RSS*. Available at: https://reactnative.dev/docs/running-on-device.

[3] Masiello, Eric, and Jacob Friedmann. *Mastering React Native*. Packt Publishing Ltd, 2017.

[4] Kiano, J.G., 2018. *A Mobile application to improve tracking and verification of products in supply chain logistics using blockchain technology* (Doctoral dissertation, Strathmore University).

[5] Wu, Wenhao. "React Native vs Flutter, Cross-platforms mobile application frameworks." (2018).

[6] Boduch, A. and Derks, R., 2020. *React and React Native: A complete hands-on guide to modern web and mobile development with React. js*. Packt Publishing Ltd.

[7] Kaushik, V., Gupta, K. and Gupta, D., 2019. React native application development. *International Journal of Advanced Studies of Scientific Research*, *4*(1).

[8] Hansson, N. and Vidhall, T., 2016. Effects on performance and usability for cross-platform application development using React Native.

[9] *Create a custom website: No-code website builder* (no date) *Webflow*. Available at: https://webflow.com/ .

[10] Svennerberg, G., 2010. *Beginning google maps API 3*. Apress.

[11] Mueller, J.P., 2006. *Mining Google web services: building applications with the Google API*. John Wiley & Sons.

[12] Markovich, S., Achwal, N. and Queathem, E., 2017. Stripe: Helping money move on the internet. *Kellogg school of management cases*, pp.1-12.

[13] Sullivan, R.J., 2013. The US adoption of computer-chip payment cards: Implications for payment fraud. *Economic Review-Federal Reserve Bank of Kansas City*, p.59.

[14] Soininen, V., 2021. Jetpack Compose vs React Native–Differences in UI Development.

[15] Tilkov, S. and Vinoski, S., 2010. Node. js: Using JavaScript to build high-performance network programs. *IEEE Internet Computing*, *14*(6), pp.80-83.