IR@AIKTC-KRRC

A PROJECT REPORT

ON

"COURIO DELIVERY SERVICE"

Submitted to

UNIVERSITY OF MUMBAI

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER ENGINEERING

BY

MOHAMMED SOHAIL SIDDIQUI	16CO59
MANSOOR ALI MALIK BASHA	16CO35
KHAN GULAM HAIDER	16CO26
ISMAIL MEHMOOD MULLA	16DCO63

UNDER THE GUIDANCE OF
PROF. IRFAN JAMKHANDIKAR

DEPARTMENT OF COMPUTER ENGINEERING
Anjuman-I-Islam's Kalsekar Technical Campus
SCHOOL OF ENGINEERING & TECHNOLOGY
Plot No. 2 3, Sector - 16, Near Thana Naka,
Khandagaon, New Panvel - 410206

2020-2021

AFFILIATED TO

UNIVERSITY OF MUMBAI

ir.aiktclibrary.org

IR@AIKTC-KRRC

A PROJECT II REPORT
ON

"COURIO DELIVERY SERVICE"

Submitted to
UNIVERSITY OF MUMBAI

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN COMPUTER ENGINEERING

BY

MOHAMMED SOHAIL SIDDIQUI	16CO59
MANSOOR ALI MALIK BASHA	16CO35
KHAN GULAM HAIDER	16CO26
ISMAIL MEHMOOD MULLA	16DCO63

UNDER THE GUIDANCE OF
PROF. IRFAN JAMKHANDIKAR

DEPARTMENT OF COMPUTER ENGINEERING
Anjuman-I-Islam's Kalsekar Technical Campus
SCHOOL OF ENGINEERING & TECHNOLOGY
Plot No. 2 3, Sector - 16, Near Thana Naka,
Khandagaon, New Panvel - 410206

2020-2021
AFFILIATED TO

UNIVERSITY OF MUMBAI

ir.aiktclibrary.org

IR@AIKTC-KRRC

Anjuman-i-Islam's Kalsekar Technical Campus

Department of Computer Engineering

SCHOOL OF ENGINEERING & TECHNOLOGY

Plot No. 2 3, Sector - 16, Near Thana Naka,

Khandagaon, New Panvel - 410206

CERTIFICATE

This is certify that the project entitled

"COURIO DELIVERY SERVICES"

submitted by

MOHAMMED SOHAIL SIDDIQUI	**16CO59**
MANSOOR ALI MALIK BASHA	**16CO35**
KHAN GULAM HAIDER	**16CO26**
ISMAIL MEHMOOD MULLA	**16DCO63**

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Engineering) at ***Anjuman-I-Islam's Kalsekar Technical Campus, Navi Mumbai*** under the University of MUMBAI. This work is done during year 2020-2021, under our guidance.

Date:26/05/2021

(Prof. IRFAN JAMKHANDIKAR)
Project Guide

(Prof. KALPANA R. BODKE)
Project Coordinator

(Prof. TABREZ KHAN)
HOD, Computer Department

DR. ABDUL RAZAK HONNUTAGI
Director

External Examiner

ir.aiktclibrary.org

IR@AIKTC-KRRC

Acknowledgements

I would like to take the opportunity to express my sincere thanks to my guide **Prof. Irfan Jamkhandikar**, Assistant Professor, Department of Computer Engineering, AIKTC, School of Engineering, Panvel for his invaluable support and guidance throughout my project research work. Without his kind guidance & support this was not possible.

I am grateful to him/her for his timely feedback which helped me track and schedule the process effectively. His/her time, ideas and encouragement that he gave is help me to complete my project efficiently.

We would like to express deepest appreciation towards **DR. ABDUL RAZAK HONNUTAGI**, Director, AIKTC, Navi Mumbai, **Prof. Tabrez Khan**, Head of Department of Computer Engineering and **Prof. Kalpana R. Bodke**, Project Coordinator whose invaluable guidance supported us in completing this project.

At last we must express our sincere heartfelt gratitude to all the staff members of Computer Engineering Department who helped me directly or indirectly during this course of work.

MOHAMMED SOHAIL SIDDIQUI
MANSOOR ALI MALIK BASHA
KHAN GULAM HAIDER
MULLA MEHMOOD ISMAIL

ir.aiktclibrary.org

IR@AIKTC-KRRC

Project I Approval for Bachelor of Engineering

This project entitled ***Ïroject Title¨*** by ***Students Name*** is approved for the degree of ***Bachelor of Engineering in Department of Computer Engineering.***

Examiners

1. .....
2. .....

Supervisors

1. .....
2. .....

Chairman

.....

ir.aiktclibrary.org

IR@AIKTC-KRRC

Declaration

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Student Name: MOHAMMED SOHAIL SIDDIQUI
Roll Number: 16CO59
Student Name: MANSOOR ALI MALIK BASHA
Roll Number: 16CO35
Student Name: KHAN GULAM HAIDER
Roll Number: 16CO26
Student Name: MULLA MEHMOOD ISMAIL
Roll Number: 16DCO63

ir.aiktclibrary.org

IR@AIKTC-KRRC

ABSTRACT

The global same day delivery market size was valued at USD 4.6 billion in 2019 and is expected to grow at a compound annual growth rate (CAGR) of 20.3 from 2020 to 2027. The market is driven by increasing urbanization, rapid e-commerce adoption, and changing customer expectations towards delivery services.

Moreover, the ongoing substitution of stationary retail sales by online sales through e-commerce platforms has led to a significant increase in B2C shipments. Thus, the market is expected to witness substantial growth over the forecast period due to the rise in shipments, coupled with consumer demand for faster delivery service.

Long delivery time is one of the significant reasons owing to which customers shop in brick and mortar stores than on online platforms. However, same day delivery services offer products in less than 24 hours i.e., preferably within the same day of the order placement. Thus, it integrates the convenience of online retail shopping with the immediacy of physical retail stores. Moreover, ease of ordering through online platforms, coupled with reduced shipping time, is creating growth opportunities for the market.

Additionally, the availability of same day delivery services is further expected to support e-commerce adoption among consumers. Thus, online retailers are expected to be benefit ted from the adoption of same day services as reduced delivery time of the product and higher convenience improves their position versus stationary retailers. Moreover, same day delivery services combine the convenience of online shopping with the immediate product availability of retail. The aforementioned benefits, coupled with the rising adoption of same day delivery services among e-commerce platforms, are anticipated to bolster the market growth over the forecast period.
Keywords: B2C, e-commerce

ir.aiktclibrary.org

IR@AIKTC-KRRC

Contents

Acknowledgement iii
Project I Approval for Bachelor of Engineering iv
Declaration v
Abstract vi
Table of Contents ix

1 Introduction **2**
1.1 Purpose 2
1.2 Project Scope 2
1.3 Project Goals and Objectives 3
1.3.1 Goals 3
1.3.2 Objectives 3

2 Literature Survey **4**
2.1 Design and Operation of an Urban Electric Courier Cargo Bike System 4
2.1.1 Advantages of Paper 4
2.1.2 Disadvantages of Paper 4
2.1.3 How to overcome the problems mentioned in Paper 4
2.2 An Integrated Courier Services Application: A New User Experience 4
2.2.1 Advantages of Paper 5
2.2.2 Disadvantages of Paper 5
2.2.3 How to overcome the problems mentioned in Paper 5
2.3 A New Express Management System Based on Encrypted QR Code 5
2.3.1 Advantages of Paper 6
2.3.2 Disadvantages of Paper 6
2.3.3 How to overcome the problems mentioned in Paper 6
2.4 Technical Review 6
2.4.1 Advantages of Technology 6
2.4.2 Reasons to use this Technology 6

3 Project Planning **7**
3.1 Members and Capabilities 7
3.2 Roles and Responsibilities 7
3.3 Assumptions and Constraints 7

ir.aiktclibrary.org

IR@AIKTC-KRRC

3.4 Project Management Approach 8
3.5 Ground Rules for the Project 8
3.6 Project Budget 8
3.7 Project Timeline 9

4 Software Requirements Specification 10
4.1 Overall Description 10
4.1.1 Product Perspective 10
4.1.2 Product Features 10
4.1.3 User Classes and Characteristics 10
4.1.4 Operating Environment 11
4.1.5 Design and Implementation Constraints 11
4.2 System Features 11
4.2.1 System Feature 11
4.3 External Interface Requirements 12
4.3.1 User Interfaces 12
4.3.2 Hardware Interfaces 12
4.3.3 Software Interfaces 12
4.4 Nonfunctional Requirements 13
4.4.1 Performance Requirements 13
4.4.2 Safety Requirements 13
4.4.3 Security Requirements 13

5 System Design 14
5.1 System Requirements Definition 14
5.1.1 Functional requirements 14
5.1.2 System requirements (non-functional requirements) 16
5.2 System Architecture Design 16
5.3 Sub-system Development 17
5.3.1 Orders 17
5.3.2 Shipments 18
5.3.3 Billing 18
5.3.4 Tools 19
5.3.5 Support 20
5.3.6 Returns 20
5.3.7 Account Management 21
5.3.8 Inventory 21
5.4 Systems Integration 21
5.4.1 Class Diagram 21
5.4.2 Sequence Diagram 22

ir.aiktclibrary.org

IR@AIKTC-KRRC

6 Implementation **24**
6.1 Dashboard 24
6.2 Order Detail 39
6.3 Schedule Order 45
6.4 Your Order 53
6.5 Admin Navigator 55
6.6 Sign Up Screen 57
6.7 Swipe Screen 63
6.8 Welcome Screen 64

7 System Testing **67**
7.1 Test Cases and Test Results 67
7.2 Test Case 67
7.2.1 Software Quality Attributes 69

8 Screenshots of Project **70**
8.1 Map - Selecting Pickup 70
8.2 Search Destination 71
8.3 Add Multiple Delivery Points 72
8.4 Map Route 73
8.5 Add Address Details 74
8.6 Add Package Details 75
8.7 Driver - List of Assigned Order 76
8.8 Driver - Order Route in Map with Order Details 77

9 Conclusion and Future Scope **78**
9.1 Conclusion 78
9.2 Future Scope 78

References **78**

ir.aiktclibrary.org

IR@AIKTC-KRRC

List of Figures

2.1 **image** 5

5.1 **Sender** 15
5.2 **Receiver** 15
5.3 **Driver** 16
5.4 **Driver** 17
5.5 **order** 18
5.6 **order** 18
5.7 **Shipment** 18
5.8 **Billing** 19
5.9 **Billing** 19
5.10 **Tools** 19
5.11 **Tools** 19
5.12 **Tools** 19
5.13 **Support** 20
5.14 **Support** 20
5.15 **Return** 20
5.16 **Account Management** 21
5.17 **Inventory** 21
5.18 **UML** 22
5.19 **Sequence Diagram** 23

8.1 **Select Pickup** 70
8.2 **Search Destination** 71
8.3 **Add Multiple Delivery Points** 72
8.4 **Map route** 73
8.5 **Address Details** 74
8.6 **Package Details** 75
8.7 **Assigned Orders** 76
8.8 **Order Route** 77

ir.aiktclibrary.org

IR@AIKTC-KRRC

List of Tables

3.1 Table of Capabilities 7
3.2 Table of Responsibilities 7

ir.aiktclibrary.org

IR@AIKTC-KRRC

Chapter 1

Introduction

The largest share of the consumer goods market is contributed by BRICS nations, followed by North America and Europe. The courier, express, and parcel market is projected to be driven by the revival of the manufacturing sector and expected economic expansion. Asian countries such as China, Taiwan, Korea, and Thailand are witnessing increased penetration of electronic devices. This, coupled with the availability of multiple online payment methods and the growth in electronic goods, automobiles, and food and beverage shipments, will benefit the courier, express, and parcel industry. The rise in contract manufacturing of FMCGs in emerging markets will also significantly influence courier, express, and parcel market growth over the forecast period.

1.1 Purpose

The global Courier Services Market is experiencing a sizeable growth, and will grow considerably in the next few years. Key players of the Courier service market analyzed in the research include FedEx Corporation, SF Express (Group) Co. Ltd., Deutsche Post DHL Group, United Parcel Service Inc., These companies have adopted several strategies such as product launches, partnerships, collaborations, mergers acquisitions, and joint ventures to maintain their foothold in the market.

1.2 Project Scope

In this current economic turmoil in the country, it is really essential to make every Ringgit's and time counts. Any sorts of waste, either an internal or external occurrence, they are a large amount of expenses of time and money. It can be improve through more efficient supply chain responsiveness.

Customer to customer (C2C) is a business model whereby customers can trade with each other, typically in an online environment, So need to go anywhere. [2]

ir.aiktclibrary.org

1.3 Project Goals and Objectives

1.3.1 Goals

1) Build a Quality Offering
2) Organizing Freight Contacts
3) Building Standardized Contracts
4) Customer Service Goals

1.3.2 Objectives

Making the process of courier digitalized and convenient for any one to ship their parcel from any place without leaving their couch on same day, by providing the best optimized routes.

The goal of a C2C is to enable these relationships, helping buyers and sellers locate each other. Customers can benefit from the competition for products and easily find products that may otherwise be difficult to locate.

Thanks to the internet, bigger intermediary companies have fostered more C2C interaction. The most prominent examples of C2C include eBay, an online auction site, and Amazon, which acts as both a B2C and a C2C marketplace. eBay has been successful since its launch in 1995, and it has always been a C2C. Anyone can sign up and begin selling or buying, giving an early voice to consumers in the e-commerce revolution.

IR@AIKTC-KRRC

Chapter 2

Literature Survey

2.1 Design and Operation of an Urban Electric Courier Cargo Bike System

The main goal of this paper is to present the recent developments of a project in Munich where package delivery is carried out by a Courier, Express and Parcel (CEP) company by cargo bikes and eBikes. For the project, two containers and one truck trailer are placed in a designated project area in the city center. These containers function as depots for the parcels to be delivered nearby.

2.1.1 Advantages of Paper

a. No inventory is required every where

b. Dynamic routing

2.1.2 Disadvantages of Paper

a. Heavy containers aren't suitable to go every where

2.1.3 How to overcome the problems mentioned in Paper

a. Instead only small trucks can be considered

2.2 An Integrated Courier Services Application: A New User Experience

This paper presents a study that integrate the selected courier service providers in one stop to save user's time in terms of price comparison while develop a low-cost peer-to-peer courier service for the users. The scope of the study does not only limit itself to the concept of integrating the quotation of the courier services but also

ir.aiktclibrary.org

matching up people who wants to deliver their parcels and people willing to help them and earn while travelling.

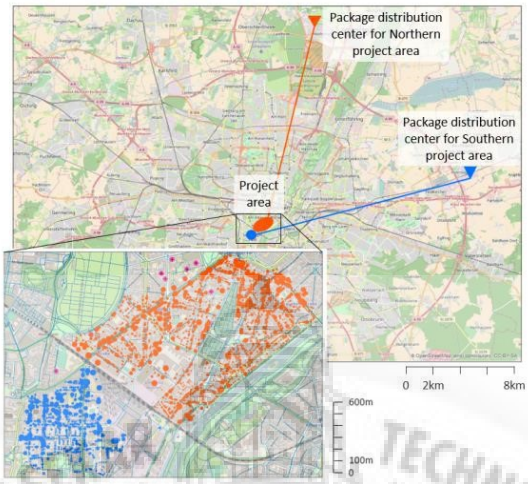


Figure 2.1: image

2.2.1 Advantages of Paper

a. Gives blueprint for delivery system

b. Compares all the quotation from different available courier

2.2.2 Disadvantages of Paper

a. Doesn't considers real life return scenario

2.2.3 How to overcome the problems mentioned in Paper

a. Real Life return scenario is considered by introducing the inventory where all the return will be stored and can be rescheduled

2.3 A New Express Management System Based on Encrypted QR Code

In this paper they have proposed a courier manage system based on encrypted QR code. To solve drawbacks like high risk of information leak, automatic updating and low efficiency of freight. By upgrading to QR code based system the results where quite interesting, the total time taken was decreased by 14 times the initial ones and precision rate was 100

2.3.1 Advantages of Paper

a. High Privacy Level

b. High precision level

2.3.2 Disadvantages of Paper

a. Only considers ideal situation

2.3.3 How to overcome the problems mentioned in Paper

a. Realistic parameters are taken in account by considering the traffic, whether, construction, etc.

2.4 Technical Review

The impact evaluation uses a client survey and other data to analyze the experiences and outcomes of participants at 13 of the 15 CBOs engaged in the C2C study and compares them with participants at ten CBOs offering similar social services but without the mental health support integration.

2.4.1 Advantages of Technology

a. Consumer Requirement

b. Live Tracking

c. Billing

2.4.2 Reasons to use this Technology

a. Optimization

b. API

Chapter 3

Project Planning

3.1 Members and Capabilities

Table 3.1: Table of Capabilities

SR. No	Name of Member	Capabilities
1	Mohammed Sohail Siddiqui	Model Preprocessing, Integration
2	Mansoor Ali Malik Basha	App UI Design, Frontend Development
3	Khan Gulam Haider	Market research, Literature Survey
4	Mulla Mehmood Ismail	Literature Survey

Work Breakdown Structure
a. All of the members are equally important in developing the project.
b. We work on a different part of the project based on one's capability.
c. Firstly we came up with documentation, And based on the documentation we set our goal and created a blueprint.
d. We then started going hands-on with the project to develop it according to the flow as decided earlier.

3.2 Roles and Responsibilities

Table 3.2: Table of Responsibilities

SR. No	Name of Member	Role	Responsibilities
1	Mohammed Sohail Siddiqui	Team Leader	Integration, Development
2	Mansoor Ali Malik Basha	Embedding System	UI, Frontend
3	Khan Gulam Haider	Project Timeline Manager	Market research
4	Mulla Mehmood Ismail	Project Timeline Manager	Literature Survey

3.3 Assumptions and Constraints

a) User using this app need their work done easily.
b) User of this app excepts cheaper rate with best Quality

c) User will use this App by 24/7.
d) The App promises same day delivery with proper planning.
e) The Delivery executive may not available at some time.

3.4 Project Management Approach

a. Planning of project.
b. Defining the scope of the project.
c. Estimation of time and It's management.
d. Creating Gantt Charts and properly assigning tasks to members.
e. Reporting the progress of project with the guide.

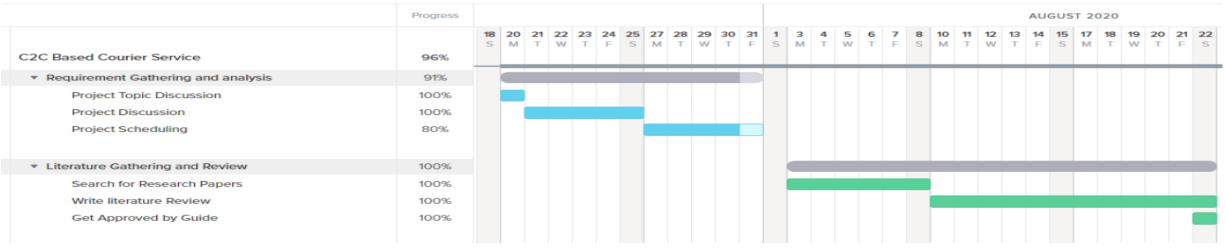
3.5 Ground Rules for the Project

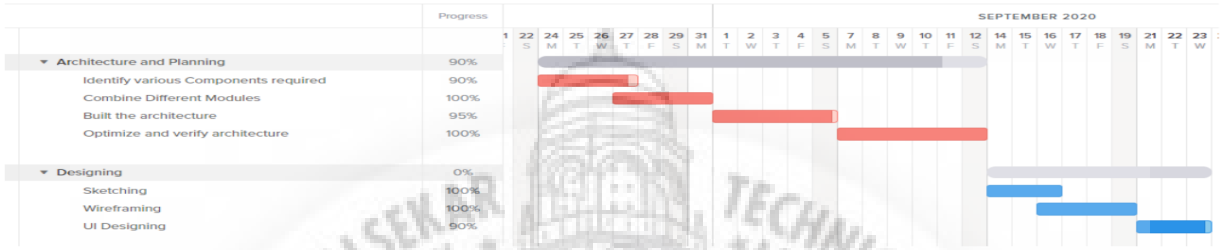
a. Properly planning and gathering relevant information is very important.
b. Developing a Blueprint of the project and work accordingly.
c. All the members should report to the guide whenever required
d. Setting up small goals every week.
e. Achieving the small goal within that span of time.
f. Keeping tracks of the progress towards project.

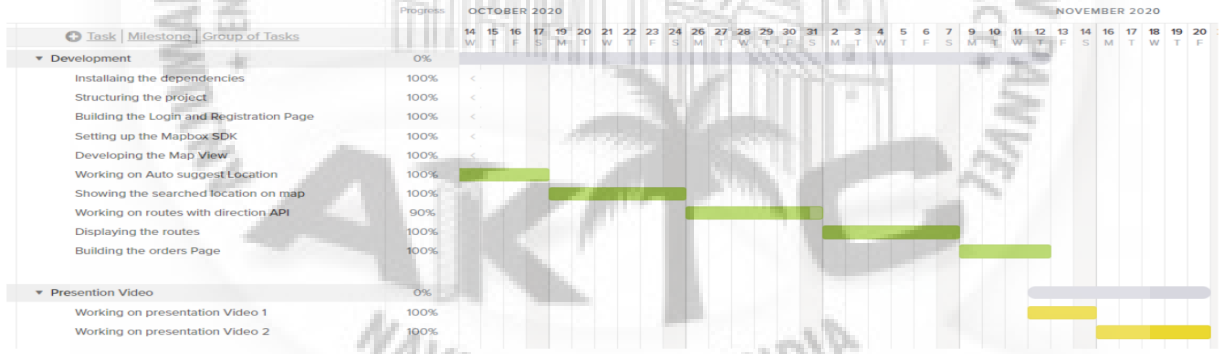
3.6 Project Budget

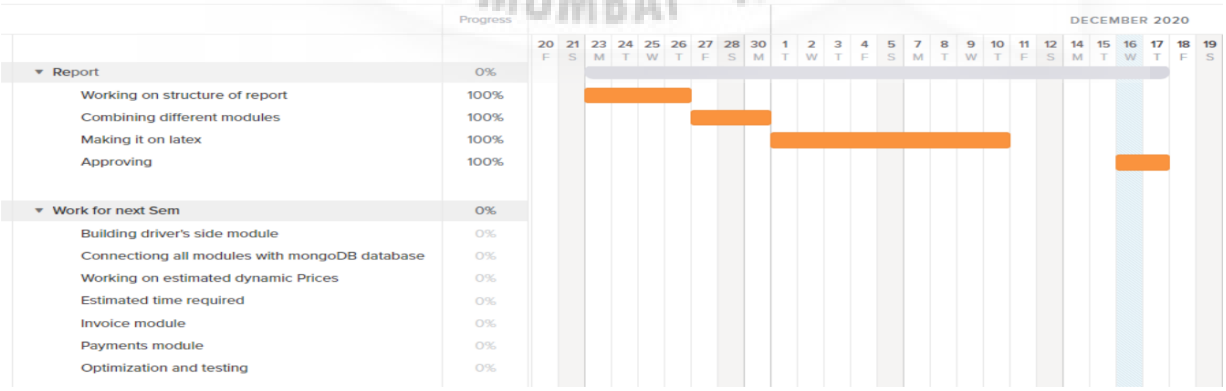
a. It is a light project.
b. Cost of the project is very low and efficient.

3.7 Project Timeline









IR@AIKTC-KRRC

Chapter 4

Software Requirements Specification

4.1 Overall Description

4.1.1 Product Perspective

E-commerce and the digital revolution has resulted in new business models affecting supply chain configuration. These models integrate further inter-company and intra-company functions (related to physical, financial and information flows) and make extensive use of technological solutions to improve the retail experience. In view of this, infrastructure development (from the perspective of improving distribution networks) and (adding value to) transport services could be important elements in the path to develop transport systems that are conducive to promote e-commerce in developing countries.[1].

4.1.2 Product Features

There are many feature available in the application,
1) Orders - Place an Order / Process Order / Generate Pickup
2) Shipments - Track Order / Weight Discrepancy
3) Billing - Shipping Charges / Invoices / COD Remittance
4) Tools - Rate Calculator / Activity Log / Report
5) Support - Help Center / Training / Tickets
6) Returns - Manage the return Orders

4.1.3 User Classes and Characteristics

This project is designed all the people who value their time and except the better and innovative service which saves their time and work efficiently, and the people who expects cheaper value and Best Service.

ir.aiktclibrary.org

4.1.4 Operating Environment

a) React Native
b) MongoDB
c) Android Studio
d) Map Box API and SDK

4.1.5 Design and Implementation Constraints

This system focuses one of the features at time. It is not able to provide two or more services at a time.At any instant only one of the services is accessible. Suppose, While using Order placing feature one cannot search for shipments or vice-versa.

4.2 System Features

The C2C model involves transactions between consumers. Here a consumer sells directly to another consumer. eBay and craigslist are common examples of online auction websites that provide a consumer to advertise and sell their product online to another consumer.
The most common C2C e-commerce models are free online classifieds, auctions, forums, and individual pages for start-up entrepreneurs. Personal items and used items are mainly transacted in this market.

4.2.1 System Feature

1) Low transaction cost
2) No intermediary
3) Wide reach
4) Round the clock availability

Description and Priority

Buyers benefit greatly from using C2C websites mostly because of the reduced price. What is more, they can deal with different sellers. Besides this, searches using criteria are available. For example, it is possible to select the best sellers, most popular products or offers from your area and much more. One more important thing is that users may choose the best proposal, contacting you directly without intermediary assistance.

Stimulus/Response Sequences

The main pros of C2C for vendors are high profitability due to direct sales. Sellers avail themselves mostly through overhead cost reduction. For entrepreneurs, this means that there is no need to spend money on facilities like rent, office supplies or salaries. Furthermore, this type of e-commerce broadens the range of potential clients as it covers not only national but also the international market. That the transaction cost is not high is definitely a plus. Last, but not least, is efficiency in selling personal or unique goods, including handmade products.

Functional Requirements

Potential users of such a C2C E-Commerce platform, such as the customers of an ISP, will normally not search for individual services (e.g. a specific credit card approval service), but for complete solutions to address their business needs (e.g. services to provide a certain kind of payment functionality for the Web shop). That is why the platform must be capable to support very different application scenarios, while at the same time it should remain generic enough so that the same set of common components can be (re-)used to run each of such scenarios. REQ-1: Optimization of Routing

4.3 External Interface Requirements

4.3.1 User Interfaces

It is very light app, so the GUI is very simple.Home pages provides two button to provide two different services to the users. The first button is of text recognition and second button is of object recognition.This button connects to a new window with respective services. Those windows consist of field taking image as an input and performing task on that and giving result.

4.3.2 Hardware Interfaces

This app requires permission of some of the hardware commodities. One need to give camera access in-order to capture the pic. One should also give storage access to save the images. Location access should also given to delivery purpose with GPS.

4.3.3 Software Interfaces

This software uses different libraries.Numpy library is use to compute and evaluate the images. Agile model is used to deal with the Map live tracking part. MongoDB

database to interact with data. GPS is used to interact with Location. And also use it to match while performing the task.OS support also needed to manage the images.

4.4 Nonfunctional Requirements

4.4.1 Performance Requirements

Performance of overall system is very efficient and well optimize. From the time taken to capture photo of package and process it everything is well organized.While placing an order it take same time for other operations.Mapping route take only seconds.

4.4.2 Safety Requirements

This system does not contain any critical data. Still it provide. The databases that are accessed are locally executed.In case of any updates in libraries used can lead to the failure in systems.

4.4.3 Security Requirements

All the libraries used are certified and standard.Also camera access is until the process is done completely.After that is released.

Chapter 5

System Design

5.1 System Requirements Definition

System requirement definitions specify [1] what the system should do, its functionality and its essential and desirable system properties. The techniques applied to elicit and collect information in order to create system specifications and requirement definitions involve consultations, interviews, requirements workshop with customers and end users. The objective of the requirements definition phase is to derive the two types of requirement:

5.1.1 Functional requirements

They define the basic functions that the system must provide and focus on the needs and goals of the end users.

Use-case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.In our system User will interact with use cases like Capture Image,Audio Input,Save text,Retrieve Text,Audio output.

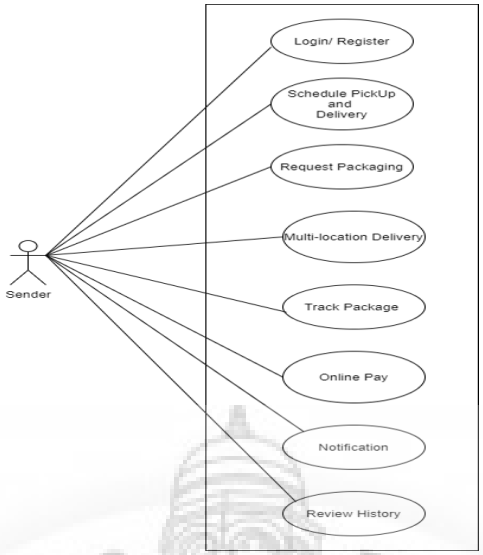


Figure 5.1: Sender

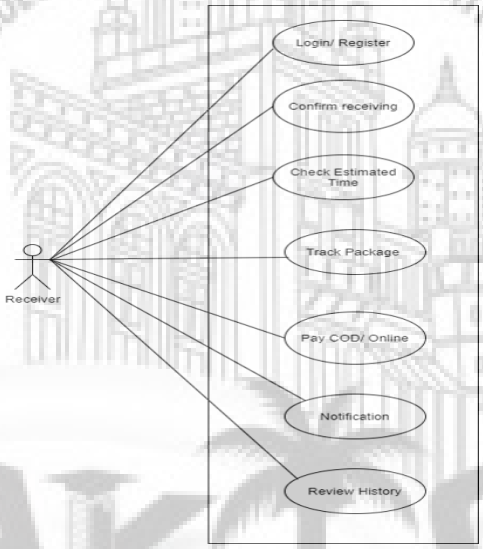


Figure 5.2: Receiver

ir.aiktclibrary.org

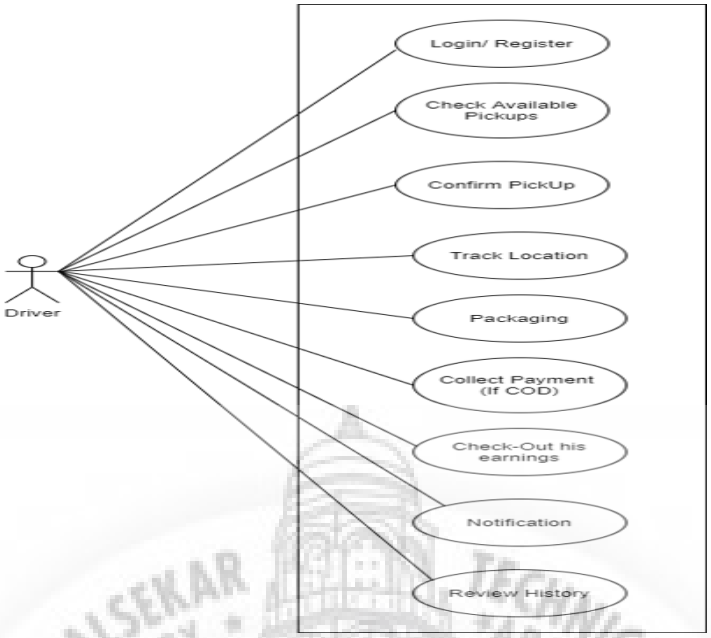


Figure 5.3: Driver

5.1.2 System requirements (non-functional requirements)

These are non-functional system properties such as availability, performance and safety etc. They define functions of a system, services and operational constraints in detail.
a. Usability - Application implementation is feasible using technologies that are accessible to the end-users.
b. Portability - The interfaces are compatible with Android.
c. Performance Efficiency -Application is able to perform well in a proper time constraint.
d. Multi User System -Application is able to consider the presence of more than one user in the same environment. All the features of the system operates properly for all users and provides proper transparency.
e. Time Efficiency - Time taken for the executing of system is less.

5.2 System Architecture Design

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

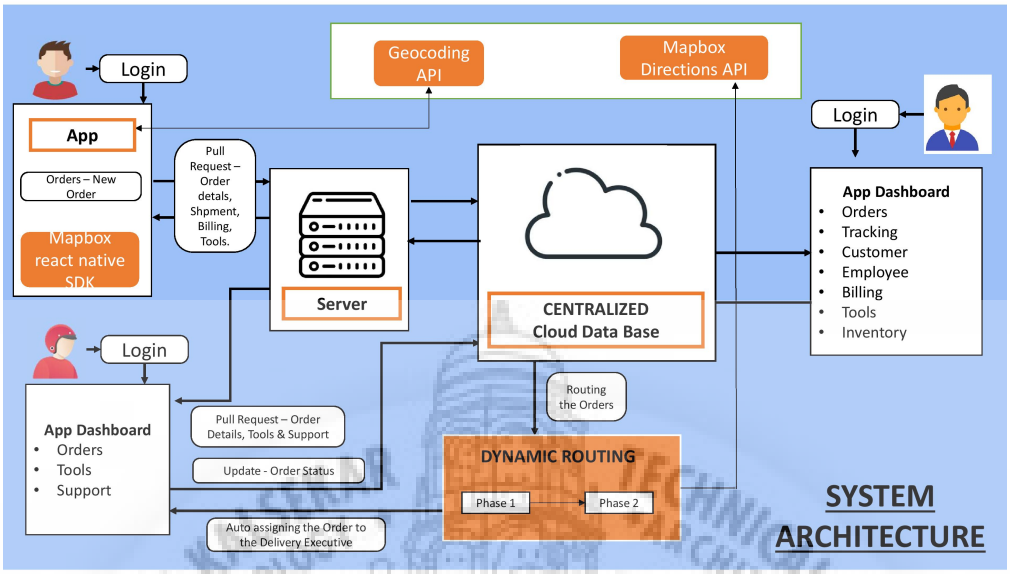


Figure 5.4: Driver

5.3 Sub-system Development

In this module the Customer / Delivery executive / Admin have a different Dashboard respectively, for Customer it shows the statistics of his/her orders in category as Completed order, pending order, pending for pickup, pending for delivery etc.
For Delivery executive the Dashboard shows statistical information of his assigned orders and the list of orders assigned to him as a Quick view for his assigned orders from where he can accept the order, check the order and pick and drop respectively.
And for Admin the Dashboard shows the statistical details of orders place by customers and the processing orders of Delivery executive and pending orders so that he can trace the orders and assist the customer Delivery executive wherever required.

5.3.1 Orders

This module 'Orders' is mainly used to place order (Add/Process Order) and can do the process related to completion of the Order which will be placed.
As same as the Dashboard in this Order module there are different features available to different users like Customer / Delivery executive / Admin. For Customer they can Add / Process the Order, can Generate the pick-up as per their convenient time, can download the Order Manifest and check their Order History.
For Delivery executive he will be able to see the list of assigned order to him, he

need to accept the order, pick-up and Drop the conveniently as per the scheduled time.
For the Admin he will get all the details of the order placed by the customer and their processing details like can able to see in the Category like Active / Pending / Picked-up / Completed / Cancelled Orders.

Module 1 Flow Diagram or Modular Diagram

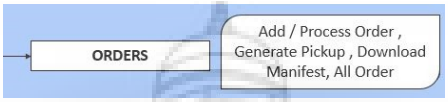


Figure 5.5: order

Figure 5.6: order

5.3.2 Shipments

This module 'Shipments' is only available for the Customer where the Customer can track the placed order and check the status of their order and check the weight Discrepancy so that they can place the order as per their packet.

Module 2 Flow Diagram or Modular Diagram

Figure 5.7: Shipment

5.3.3 Billing

This module 'Billing' is only for Customer Admin where Customer can check the Billing details of their order and check the shipment charges and can download all the completed orders Invoice for the record.
For Admin he can check all the Invoices of the customer and will maintain the record of the Cash On Delivery (COD) Remittance.
When the customers place the order on COD Option, they agree to pay the cash to the delivery guy when the order is picked-up. The courier guy takes cash from the customer, deposits it to the respective Admin. This is called as COD Remittance simply we can define as to remit the cash from Delivery Executive is called as COD Remittance, which will be managed by Admin under this feature.

Module 3 Flow Diagram or Modular Diagram

Figure 5.8: Billing

Figure 5.9: Billing

5.3.4 Tools

This module 'Tools' mainly consist of the feature namely Rate Calculator, Activity log Report and this is available for all three types of user Customer / Delivery Executive / Admin.

For Customer, available feature is Rate Calculator. The Rate Calculator is the Calculator used to calculate the price for the packet respective to the size of the packet and distance between pick-up and delivery location, it will be useful for the customer to check the rate for this order.

For Delivery Executive, available features are Activity log Report. The Activity log is the log (Record) of the assigned order to the Delivery Executive or the Activity of the Delivery Executive to the Assigned order and Report are the formal details of the Orders assigned to him and his contribution to the orders.

For Admin, available features are Rate Calculator, Activity log Report. Admin can check the Activity of all the Customer Delivery executive and their respective Report.

Module 4 Flow Diagram or Modular Diagram

Figure 5.10: Tools

Figure 5.11: Tools

Figure 5.12: Tools

5.3.5 Support

This module 'Support' is mainly used for Help Support, Training videos and Tickets. And it is available for all users namely Customer, Delivery Executive Admins.

For Customer, he will get the feature Help Support where he will get the contact details of the customer care and Frequently Asked Questions (FAQ) so that he can clear the concern easily, and he will get Training videos of How to like how to place order, how to download manifest etc. so that he can use the app easily and he get option of tickets he can raise the ticket of his concern/ complaint/ feedback etc. to the customer care team and it will be resolved respectively.

For Delivery Executive also he will get all the feature provided to Customer namely help Support, Training Tickets. Where he will use it as per his requirement.

For Admin he will have option of Tickets where he will manage and resolve the tickets raised by Customer Delivery Executive.

Module 5 Flow Diagram or Modular Diagram

Figure 5.13: Support

Figure 5.14: Support

5.3.6 Returns

This module 'Returns' is only for Admin where the Admin will manage the return orders, by chance if the packet is not delivered to the desired destination, the order will be updated for the return which will be returned to the Sender.

Module 6 Flow Diagram or Modular Diagram

Figure 5.15: Return

5.3.7 Account Management

This module is only for the Admin where he will manage all the registered accounts of Customer Delivery Executive.

Module 7 Flow Diagram or Modular Diagram

Figure 5.16: Account Management

5.3.8 Inventory

This module 'Inventory' is only for the Admin and it is designed in the way that it will be used less in the last stage. normally the order will be picked-up Dropped respectively when the receiver is not in the place to receive the packet the Delivery executive will try to return it back to the sender else coordinate with sender receiver and deliver accordingly if the receiver as well as sender is not responding then the packet will be kept in Inventory. The Inventory will be the van in the city.

Module 8 Flow Diagram or Modular Diagram

Figure 5.17: Inventory

5.4 Systems Integration

System integration (SI) is an engineering process or phase concerned with joining different subsystems or components as one large system. It ensures that each integrated subsystem functions as required. SI is also used to add value to a system through new functionalities provided by connecting functions of different systems.

5.4.1 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.Our System consist of four Classes Image,object,Text and Text to Speech.

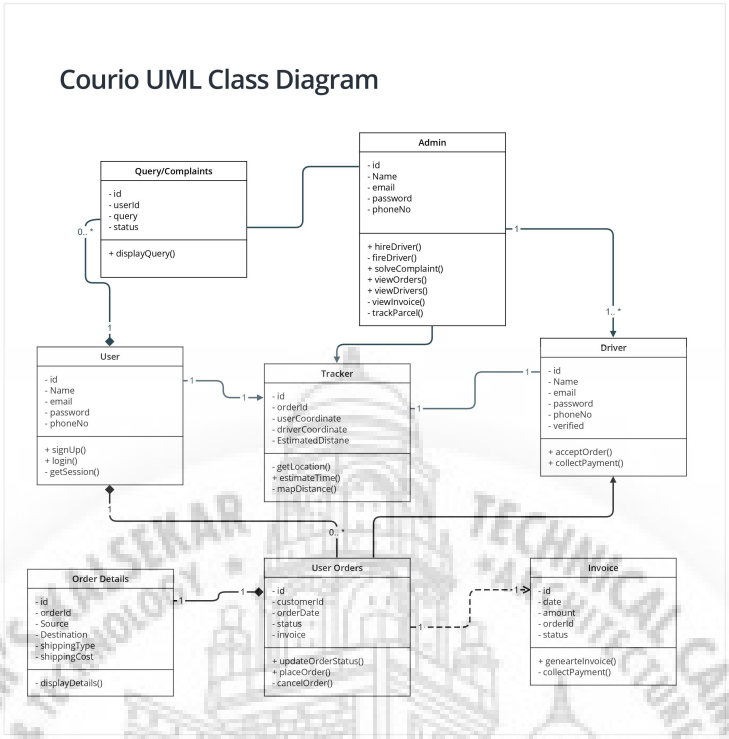


Figure 5.18: UML

5.4.2 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

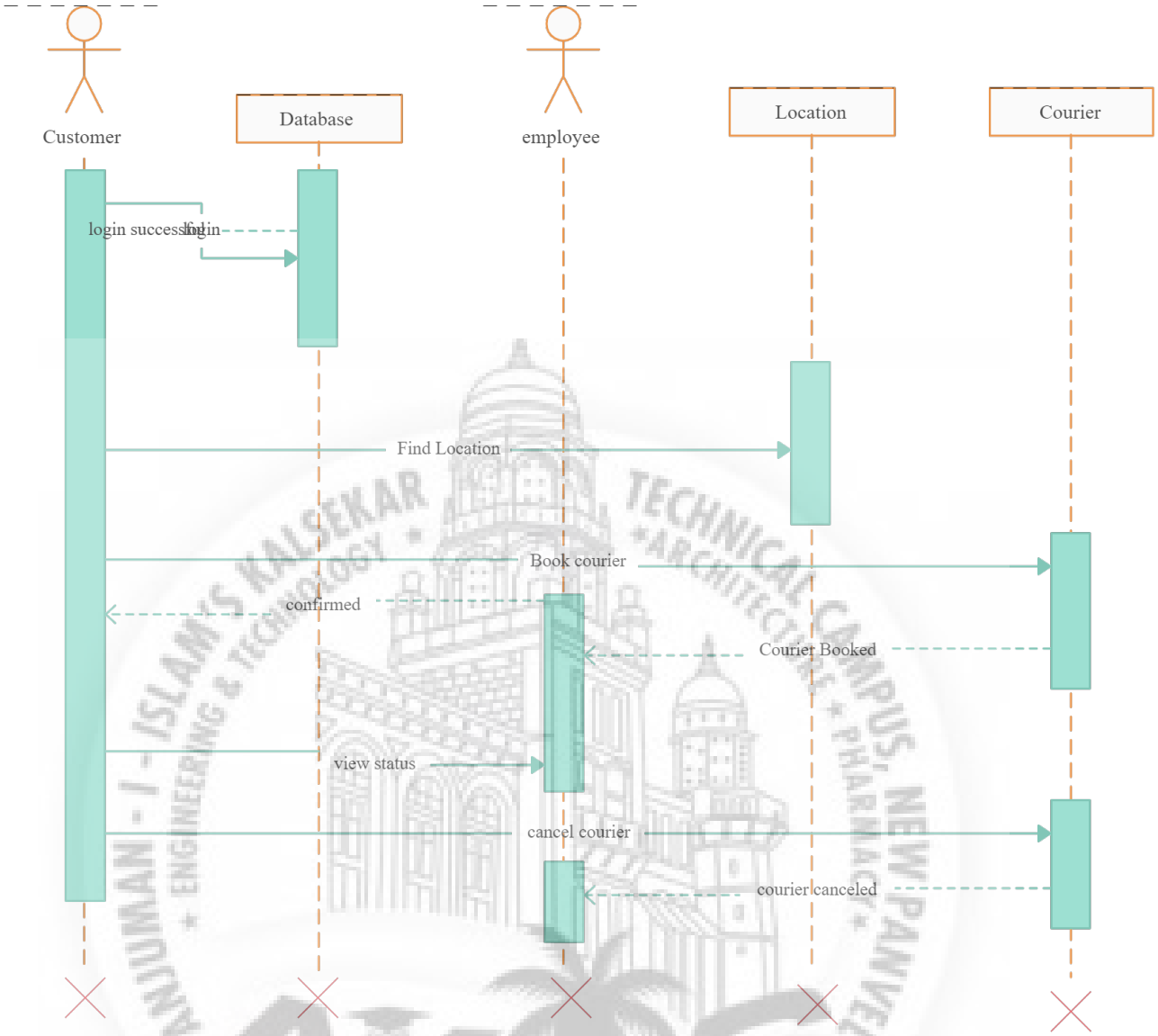


Figure 5.19: Sequence Diagram

Chapter 6

Implementation

6.1 Dashboard

```
import React, {Component} from 'react';
import axios from 'axios';

import {
  View,
  Text,
  TouchableHighlight,
  TextInput,
  Button,
  FlatList,
  ScrollView,
  Keyboard,
  StatusBar,
  Dimensions,
  KeyboardAvoidingView,
  StyleSheet,
  TouchableOpacity,
  PermissionsAndroid,
} from 'react-native';

import MapboxGL from '@react-native-mapbox-gl/maps';
import BottomSheet from 'reanimated-bottom-sheet';
import {SearchBar} from 'react-native-elements';
import Icon from 'react-native-vector-icons/Ionicons';
import SearchBarCustom from '../components/searchBar';
import Geolocation from 'react-native-geolocation-service';
//import Geolocation from '@react-native-community/geolocation';
//Geolocation.setRNConfiguration(config);
const height = Dimensions.get('window').height;

MapboxGL.setAccessToken(
  'pk.eyJ1IjoiZXplY2EiLCJhIjoiY2tndW44MXk1MG1jODJ0cGViZ2syamNhMyJ9.
     DxN05YR2CCkIGBq8cU0ZJg',
);

class App extends Component {
  constructor(props) {
    super(props);

    this.state = {
      toggleInput: false,
      focusedSourceInput: false,
      focusedInput: false,
```

```
      behavior: 'position',
      search: '',
      noStops: 2,
      upperLayoutHeight: 0,
      mapRegion: {
        latitude: 72.8331,
        longitude: 18.941,
        latitudeDelta: 0.00922 * 1.5,
        longitudeDelta: 0.00421 * 1.5,
      },
      lastLat: null,
      lastLong: null,
      location: 4.57901,
      fromInput: '',
      sourceFlag: 0,
      sourceName: 'Your Location',
      destName: '',
      dest2Name: '',
      dest3Name: '',
      source: {lat: 72.8331, lon: 18.941},
      destination: {lat: 72.831, lon: 18.9293},
      destination2: {lat: null, lon: null},
      destination3: {lat: null, lon: null},
      lat: 72.8331,
      lon: 18.941,
      placeResult: '',
      route: {
        type: 'FeatureCollection',
        features: [
          {
            type: 'Feature',
            properties: {},
            geometry: {
              type: 'LineString',
              coordinates: [],
            },
            style: {
              fill: 'blue',
              strokeWidth: '6',
              fillOpacity: 0.6,
            },
            paint: {
              'fill-color': '#088',
              'fill-opacity': 0.8,
            },
          },
        ],
      },
    };
  }

  //    {
  //      coordinates: [
  //        [72.8331, 18.941],
  //        [72.831, 18.9293],
  //      ],
  //    },

  getCoord(keyVal) {
    console.log(keyVal);
    item = this.state.placeResult[keyVal].coordinates;
```

```
104     placeName = this.state.placeResult[keyVal].name;
105     console.log(item);
106     console.log(this.state.source);
107     console.log(this.state.destination);
108     var sFlag;
109     if (this.state.fromInput == 'source') {
110       var source = this.state.source;
111       source.lat = item[0];
112       source.lon = item[1];
113       this.setState({source});
114       sFlag = this.state.sourceFlag + 1;
115       this.setState({sourceFlag: sFlag});
116       this.setState({sourceName: placeName});
117     } else if (this.state.fromInput == 'dest') {
118       var dest = this.state.destination;
119       dest.lat = item[0];
120       dest.lon = item[1];
121       this.setState({dest});
122       sFlag = this.state.sourceFlag + 1;
123       this.setState({sourceFlag: sFlag});
124       this.setState({destName: placeName});
125     } else if (this.state.fromInput == 'stop2') {
126       var dest = this.state.destination2;
127       dest.lat = item[0];
128       dest.lon = item[1];
129       this.setState({dest});
130       sFlag = this.state.sourceFlag + 1;
131       this.setState({sourceFlag: sFlag});
132       this.setState({dest2Name: placeName});
133     } else if (this.state.fromInput == 'stop3') {
134       var dest = this.state.destination3;
135       dest.lat = item[0];
136       dest.lon = item[1];
137       this.setState({dest});
138       sFlag = this.state.sourceFlag + 1;
139       this.setState({sourceFlag: sFlag});
140       this.setState({dest3Name: placeName});
141     }
142     this.setState({lat: item[0]});
143     this.setState({lon: item[1]});
144     //6.57901, 6.57901
145     this.setState({placeResult: ''});
146     console.log(']]]]]]]]]]]]]]]]]]]]');
147
148     if (this.state.sourceFlag >= 1) {
149       console.log('77777777777777777777');
150       this.direction();
151     }
152   }
153   // https://api.mapbox.com/optimized-trips/v1/mapbox/driving
        /13.388860,52.517037;13.397634,52.529407;13.428555,52.523219;13.418555,52.523215?
        source=first&destination=last&roundtrip=true&access_token=
        YOUR_MAPBOX_ACCESS_TOKEN
154   direction() {
155     if (this.state.sourceName == 'Your Location') {
156       console.log(this.state.mapRegion.latitude);
157       console.log(this.state.mapRegion.longitude);
158       // var routeUrl =
159       //   'https://api.mapbox.com/optimized-trips/v1/mapbox/driving/' +
160       //   this.state.mapRegion.longitude +
161       //   ',' +
```

```
162         //    this.state.mapRegion.latitude +
163         //    ';' +
164         //    this.state.destination.lat +
165         //    ',' +
166         //    this.state.destination.lon +
167         //    +(this.state.destination2.lat != null
168         //      ? ';' +
169         //        this.state.destination2.lat +
170         //        ',' +
171         //        this.state.destination2.lon +
172         //        ';'
173         //      : '') +
174         //    (this.state.destination3.lat != null
175         //      ? this.state.destination3.lat + ',' + this.state.destination3.lon
176         //      : '') +
177         //    +'?source=first&destination=last&roundtrip=false&geometries=geojson&&
                access_token=pk.
                eyJ1IjoiZXplY2EiLCJhIjoiY2tndW44MXk1MG1jODJ0cGViZ2syamNhMyJ9.
                DxN05YR2CCkIGBq8cU0ZJg';
178
179         var routeUrl =
180           'https://api.mapbox.com/optimized-trips/v1/mapbox/driving/' +
181           this.state.mapRegion.longitude +
182           ',' +
183           this.state.mapRegion.latitude +
184           ';' +
185           this.state.destination.lat +
186           ',' +
187           this.state.destination.lon +
188           ';' +
189           this.state.destination2.lat +
190           ',' +
191           this.state.destination2.lon +
192           ';' +
193           this.state.destination3.lat +
194           ',' +
195           this.state.destination3.lon +
196           '?source=first&destination=last&roundtrip=false&geometries=geojson&&
                access_token=pk.
                eyJ1IjoiZXplY2EiLCJhIjoiY2tndW44MXk1MG1jODJ0cGViZ2syamNhMyJ9.
                DxN05YR2CCkIGBq8cU0ZJg';
197       } else {
198         // var routeUrl =
199         //    'https://api.mapbox.com/optimized-trips/v1/mapbox/driving/' +
200         //    this.state.source.lat +
201         //    ',' +
202         //    this.state.source.lon +
203         //    ';' +
204         //    this.state.destination.lat +
205         //    ',' +
206         //    this.state.destination.lon +
207         //    +(this.state.destination2.lat != null
208         //      ? ';' +
209         //        this.state.destination2.lat +
210         //        ',' +
211         //        this.state.destination2.lon +
212         //        ';'
213         //      : '') +
214         //    (this.state.destination3.lat != null
215         //      ? this.state.destination3.lat + ',' + this.state.destination3.lon
216         //      : '') +
```

```
217       //    '?source=first&destination=last&roundtrip=false&geometries=geojson&&
              access_token=pk.
              eyJ1IjoiZXplY2EiLCJhIjoiY2tndW44MXk1MG1jODJ0cGViZ2syamNhMyJ9.
              DxN05YR2CCkIGBq8cU0ZJg';
218
219       var routeUrl =
220         'https://api.mapbox.com/optimized-trips/v1/mapbox/driving/' +
221         this.state.source.lat +
222         ',' +
223         this.state.source.lon +
224         ';' +
225         this.state.destination.lat +
226         ',' +
227         this.state.destination.lon +
228         ';' +
229         this.state.destination2.lat +
230         ',' +
231         this.state.destination2.lon +
232         ';' +
233         this.state.destination3.lat +
234         ',' +
235         this.state.destination3.lon +
236         '?source=first&destination=last&roundtrip=false&geometries=geojson&&
              access_token=pk.
              eyJ1IjoiZXplY2EiLCJhIjoiY2tndW44MXk1MG1jODJ0cGViZ2syamNhMyJ9.
              DxN05YR2CCkIGBq8cU0ZJg';
237     }
238
239     // const routeUrl =
240     //   'https://api.mapbox.com/directions/v5/mapbox/driving/' +
241     //   this.state.source.lat +
242     //   ',' +
243     //   this.state.source.lon +
244     //   ';' +
245     //   this.state.destination.lat +
246     //   ',' +
247     //   this.state.destination.lon +
248     //   ';' +
249     //   this.state.destination2.lat +
250     //   ',' +
251     //   this.state.destination2.lon +
252     //   ';' +
253     //   this.state.destination3.lat +
254     //   ',' +
255     //   this.state.destination3.lon +
256     //   '?geometries=geojson&access_token=pk.
          eyJ1IjoiZXplY2EiLCJhIjoiY2tndW44MXk1MG1jODJ0cGViZ2syamNhMyJ9.
          DxN05YR2CCkIGBq8cU0ZJg';
257     console.log('++++++++++++++++++++++++', routeUrl);
258     axios
259       .get(routeUrl)
260       .then((response) => {
261         console.log(response.request._response);
262         console.log('----------------------------');
263         console.log('----------------------------');
264         var obj = JSON.parse(response.request._response);
265         console.log(obj.trips[0].geometry.coordinates);
266         //console.log(obj.routes[0].geometry.coordinates);
267
268         var coordin = obj.trips[0].geometry.coordinates;
269         // var coordin = obj.routes[0].geometry.coordinates;
```

```
        return coordin;
      })
      .then((coordin) => {
        var geometry = this.state.route.features[0].geometry;
        console.log('routes:', coordin);
        console.log('cord:', geometry);
        geometry.coordinates = coordin;

        this.setState({geometry});
        console.log(cord);
        console.log(
          'state route',
          this.state.route.features[0].geometry.coordinates,
        );
      })
      .catch((error) => {
        console.log(error);
      });
  }
  _handlePress() {
    console.log(this.state.location);
    const searchUrl =
      'https://api.mapbox.com/geocoding/v5/mapbox.places/' +
      this.state.location +
      '.json?bbox
        =72.64472835639785,18.777422428792846,73.31946817024394,19.581105982221317&
        access_token=pk.
        eyJ1IjoiZXplY2EiLCJhIjoiY2tndW44MXk1MG1jODJ0cGViZ2syamNhMyJ9.
        DxN05YR2CCkIGBq8cU0ZJg';
    console.log(searchUrl);
    axios
      .get(searchUrl)
      .then((response) => {
        //console.log(response.request._response);
        resData = response.request._response;
        //var finalData = resData.replace(/\\/g, '');
        var obj = JSON.parse(resData);
        console.log(obj.features[0].center);
        var featuresLen = obj.features.length;
        console.log('len' + featuresLen);
        var i;
        var places = [];

        for (i = 0; i < 5; i++) {
          console.log(obj.features[i].place_name);
          placeName = obj.features[i].place_name;
          coordinate = obj.features[i].center;

          places.push({
            key: i,
            name: placeName,
            coordinates: coordinate,
          });
        }
        console.log(places);
        return places;

        //console.log(finalData.features);
      })
      .then((places) => {
        console.log(places);
```

```
        this.setState({placeResult: places});
        console.log('----------------');
        console.log(this.state.placeResult);
      })
      .catch((error) => {
        console.log(error);
      });
  }

  // openPopUp() {
  //   this.bottomSheet.open();
  // }
  find_dimesions(layout) {
    const {x, y, width, height} = layout;
    console.warn(x);
    console.warn(y);
    console.warn(width);
    console.warn(height);
    this.setState({upperLayoutHeight: height});
  }
  updateSearch = (search) => {
    this.setState({search});
  };
  getLocation = async () => {
    try {
      const granted = await PermissionsAndroid.request(
        PermissionsAndroid.PERMISSIONS.ACCESS_FINE_LOCATION,
      );
      if (granted === PermissionsAndroid.RESULTS.GRANTED) {
        Geolocation.getCurrentPosition(
          (position) => {
            console.log('pos', position);
            let region = {
              latitude: position.coords.latitude,
              longitude: position.coords.longitude,
              latitudeDelta: 0.00922 * 1.5,
              longitudeDelta: 0.00421 * 1.5,
            };
            this.setState({mapRegion: region});
            // var source = this.state.source;
            // source.lat = position.coords.latitude;
            // source.lon = position.coords.longitude;
            // this.setState({source});
          },
          (error) => {
            // See error code charts below.
            console.log(error.code, error.message);
          },
          {enableHighAccuracy: true, timeout: 15000, maximumAge: 10000},
        );
      } else {
        alert('Location permission denied');
      }
    } catch (err) {
      console.warn(err);
    }
  };
  onRegionChange = async (region, lastLat, lastLong) => {
    await this.setState({
      mapRegion: region,
      // If there are no new values set the current ones
```

```
      lastLat: lastLat || this.state.lastLat,
      lastLong: lastLong || this.state.lastLong,
    });
    console.log(region.latitude, region.longitude);
  };

  componentDidMount = async () => {
    {
      this.getLocation();
    }
    {
      this.watchID = Geolocation.watchPosition(
        (position) => {
          // Create the object to update this.state.mapRegion through the
             onRegionChange function
          let region = {
            latitude: position.coords.latitude,
            longitude: position.coords.longitude,
            latitudeDelta: 0.00922 * 1.5,
            longitudeDelta: 0.00421 * 1.5,
          };

          this.onRegionChange(region, region.latitude, region.longitude);
        },
        (error) => console.log(error),
      );
    }
  };
  componentWillUnmount() {
    Geolocation.clearWatch(this.watchID);
  }
  renderContent = () => (
    <View
      style={{
        backgroundColor: 'white',
        padding: 16,
        height: 300,
      }}>
      <View>
        <View style={styles.panelHandle} />
      </View>

      <View style={styles.searchContainer}>
        <TextInput
          placeholder="Pickup"
          name="source"
          style={styles.searchBox}
          value={this.state.sourceName}
          onChangeText={(text) => {
            this.setState({location: text});
            this.setState({fromInput: 'source'});
            this.setState({sourceName: text});
            this._handlePress();
          }}></TextInput>
        <TextInput
          placeholder="Search Destination"
          name="dest"
          style={styles.searchBox}
          value={this.state.destName}
          onChangeText={(text) => {
            this.setState({location: text});
```

```
            this.setState({fromInput: 'dest'});
            this.setState({destName: text});
            this._handlePress();
          }}></TextInput>
      </View>
      <View>
        <FlatList
          data={this.state.placeResult}
          renderItem={({item}) => (
            <TouchableHighlight onPress={() => this.getCoord(item.key)}>
              <Text key={item.key.toString()} bottomDivider>
                {item.name}
              </Text>
            </TouchableHighlight>
          )}
        />
      </View>
    </View>
  );

  render() {
    return (
      <View style={{flex: 1, height: '100%', width: '100%'}}>
        <View>
          <View
            style={styles.searchContainer}
            onLayout={(event) => {
              this.find_dimesions(event.nativeEvent.layout);
            }}>
            {this.state.toggleInput && (
              <View style={{marginBottom: 8, marginTop: 8}}>
                {/* <TextInput
                //placeholder="Pickup"
                name="source"
                style={styles.searchBox}
                value={this.state.sourceName}
                onFocus={() => {
                  if (this.state.sourceName == 'Your Location') {
                    this.setState({sourceName: ''});
                  }
                }}
                onChangeText={(text) => {
                  this.setState({location: text});
                  this.setState({fromInput: 'source'});
                  this.setState({sourceName: text});
                  this._handlePress();
                }}></TextInput> */}
                <SearchBarCustom
                  textValue={this.state.sourceName}
                  onFocusBox={() => {
                    if (this.state.sourceName == 'Your Location') {
                      this.setState({sourceName: ''});
                      this.setState({focusedInput: true});
                      this.setState({focusedSourceInput: true});
                    }
                  }}
                  valueChange={async (text) => {
                    this.setState({location: text});
                    this.setState({fromInput: 'source'});
                    this.setState({sourceName: text});
                    this._handlePress();
```

```
            }}></SearchBarCustom>
          </View>
        )}
        {/* <TextInput
        placeholder="Search Destination"
        name="dest"
        style={styles.searchBox}
        value={this.state.destName}
        onFocus={() => this.setState({toggleInput: true})}
        onChangeText={(text) => {
          this.setState({location: text});
          this.setState({fromInput: 'dest'});
          this.setState({destName: text});
          this._handlePress();
        }}></TextInput> */}
        <View style={{marginBottom: 8, marginTop: 8}}>
          <SearchBarCustom
            textValue={this.state.destName}
            onFocusBox={() => {
              this.setState({toggleInput: true});
              this.setState({focusedInput: true});
            }}
            valueChange={async (text) => {
              this.setState({location: text});
              this.setState({fromInput: 'dest'});
              this.setState({destName: text});
              this._handlePress();
            }}></SearchBarCustom>
        </View>
        {this.state.toggleInput && this.state.noStops >= 3 && (
          <View style={{flexDirection: 'row'}}>
            <View style={{flex: 1, marginBottom: 6}}>
              <SearchBarCustom
                onFocusBox={() => {}}
                textValue={this.state.dest2Name}
                valueChange={async (text) => {
                  this.setState({location: text});
                  this.setState({fromInput: 'stop2'});
                  await this.setState({dest2Name: text});
                  this._handlePress();
                  console.log(text);
                  console.log(this.state.dest2Name);
                }}></SearchBarCustom>
            </View>
            <TouchableOpacity>
              <Icon
                name="close-outline"
                style={{
                  marginRight: 15,
                  marginTop: 8,
                  marginLeft: 5,
                  flex: 1,
                }}
                onPress={() => {
                  this.setState({noStops: this.state.noStops - 1});
                }}
                size={25}></Icon>
            </TouchableOpacity>
          </View>
        )}
        {this.state.toggleInput && this.state.noStops >= 4 && (
```

```
            <View style={{flexDirection: 'row'}}>
              <View style={{flex: 1, marginBottom: 6}}>
                <SearchBarCustom
                  onFocusBox={() => {}}
                  textValue={this.state.dest3Name}
                  valueChange={async (text) => {
                    this.setState({location: text});
                    this.setState({fromInput: 'stop3'});
                    await this.setState({dest3Name: text});
                    this._handlePress();
                    console.log(text);
                    console.log(this.state.dest3Name);
                    console.log(this.state.dest2Name);
                  }}></SearchBarCustom>
              </View>
              <TouchableOpacity>
                <Icon
                  name="close-outline"
                  style={{
                    marginRight: 15,
                    marginTop: 8,
                    marginLeft: 5,
                    flex: 1,
                  }}
                  onPress={() => {
                    this.setState({noStops: this.state.noStops - 1});
                  }}
                  size={25}></Icon>
              </TouchableOpacity>
            </View>
          )}
          {this.state.toggleInput && this.state.noStops <= 3 && (
            <Button
              title="Add Stop"
              onPress={() => {
                this.setState({noStops: this.state.noStops + 1});
              }}></Button>
          )}
          {this.state.toggleInput && (
            <TouchableOpacity
              style={{margin: 10}}
              onPress={() => {
                this.setState({toggleInput: false});
                Keyboard.dismiss();
                this.setState({focusedInput: false});
              }}>
              <Text style={{textAlign: 'right'}}>Done</Text>
            </TouchableOpacity>
          )}
        </View>
      </View>

      {this.state.focusedInput && (
        <TouchableOpacity
          onPress={() => {
            this.setState({focusedInput: !this.state.focusedInput});
          }}>
          <View style={{height: 700, backgroundColor: 'transparent'}}>
            <ScrollView>
              {this.state.focusedSourceInput && (
                <TouchableOpacity
```

```
                    onPress={() => {
                      this.setState({location: 'Your Location'});
                      this.setState({fromInput: 'source'});
                      this.setState({sourceName: 'Your Location'});
                      this.getLocation();
                      this.setState({focusedSourceInput: false});
                    }}>
                    <Text>Your Location</Text>
                  </TouchableOpacity>
                )}

                <FlatList
                  data={this.state.placeResult}
                  renderItem={({item}) => (
                    <TouchableHighlight onPress={() => this.getCoord(item.key)}>
                      <Text key={item.key.toString()} bottomDivider>
                        {item.name}
                      </Text>
                    </TouchableHighlight>
                  )}
                />
              </ScrollView>
            </View>
          </TouchableOpacity>
        )}

        <MapboxGL.MapView
          styleURL={MapboxGL.StyleURL.Street}
          zoomLevel={16}
          centerCoordinate={[this.state.lat, this.state.lon]}
          style={{
            flex: 1,
          }}
          onPress={(feature) =>
            console.log('Coords:', feature.geometry.coordinates)
          }>
          <MapboxGL.Camera
            zoomLevel={16}
            centerCoordinate={[this.state.lat, this.state.lon]}
            // centerCoordinate={[
            //   this.state.mapRegion.latitude,
            //   this.state.mapRegion.longitude,
            // ]}
            animationMode={'flyTo'}
            animationDuration={20}></MapboxGL.Camera>
          {this.state.sourceName == 'Your Location' && (
            <MapboxGL.PointAnnotation
              key="key1"
              id="id1"
              title="sourcePointer"
              coordinate={[
                this.state.mapRegion.latitude,
                this.state.mapRegion.longitude,
              ]}></MapboxGL.PointAnnotation>
          )}
          {this.state.sourceName != 'Your Location' && (
            <MapboxGL.PointAnnotation
              key="key1"
              id="id1"
              title="sourcePointer"
              coordinate={[
```

```
            this.state.source.lat,
            this.state.source.lon,
          ]}></MapboxGL.PointAnnotation>
        )}
        <MapboxGL.PointAnnotation
          key="key2"
          id="id2"
          title="destPointer"
          coordinate={[
            this.state.destination.lat,
            this.state.destination.lon,
          ]}></MapboxGL.PointAnnotation>
        {this.state.destination2.lat != null && (
          <MapboxGL.PointAnnotation
            key="key3"
            id="id3"
            title="dest2Pointer"
            coordinate={[
              this.state.destination2.lat,
              this.state.destination2.lon,
            ]}></MapboxGL.PointAnnotation>
        )}
        {this.state.destination3.lat != null && (
          <MapboxGL.PointAnnotation
            key="key4"
            id="id4"
            title="dest3Pointer"
            coordinate={[
              this.state.destination3.lat,
              this.state.destination3.lon,
            ]}></MapboxGL.PointAnnotation>
        )}
        <MapboxGL.ShapeSource id="progressSource" shape={this.state.route}>
          <MapboxGL.LineLayer
            id="progressFill"
            style={{
              lineColor: '#ffcc5c',
              lineWidth: 6,
              lineCap: 'round',
            }}
          />
        </MapboxGL.ShapeSource>
        <MapboxGL.UserLocation visible={true} />
      </MapboxGL.MapView>

      <View
        style={{
          width: 80,
          backgroundColor: 'transparent',
          position: 'absolute',

          top: '85%',
          left: '70%',
          zIndex: 10,
        }}>
        {this.state.destination.lat != '' && (
          <Button
            onPress={() => {
              console.log('$$$$$$$$$$$$$$$$$$.....', this.state.noStops);
              //console.log('$$$$$.....', this.state.noStops);
```

```
                if (this.state.noStops == 2) {
                  console.log('sourCord', this.state.source);
                  this.props.navigation.navigate('Details', {
                    mapRegion: this.state.mapRegion,
                    sourceName: this.state.sourceName,
                    destName: this.state.destName,
                    sourceCord: this.state.source,
                    destCord: this.state.destination,
                    noStops: this.state.noStops,
                  });
                } else if (this.state.noStops == 3) {
                  this.props.navigation.navigate('Details', {
                    mapRegion: this.state.mapRegion,
                    sourceName: this.state.sourceName,
                    destName: this.state.destName,
                    dest2Name: this.state.dest2Name,
                    sourCord: this.state.source,
                    destCord: this.state.destination,
                    dest2Cord: this.state.destination2,
                    noStops: this.state.noStops,
                  });
                } else if (this.state.noStops == 4) {
                  this.props.navigation.navigate('Details', {
                    mapRegion: this.state.mapRegion,
                    sourceName: this.state.sourceName,
                    destName: this.state.destName,
                    dest2Name: this.state.dest2Name,
                    dest3Name: this.state.dest3Name,
                    sourCord: this.state.source,
                    destCord: this.state.destination,
                    dest2Cord: this.state.destination2,
                    dest3Cord: this.state.destination3,
                    noStops: this.state.noStops,
                  });
                }
              }}
              title="GO"></Button>
          )}

          {/* <Button
            onPress={() => this.props.navigation.navigate('Swipe')}
            title="Swipe"></Button> */}
        </View>

        {/* <BottomSheet
          //ref={sheetRef}
          //height - this.state.upperLayoutHeight - 80
          snapPoints={[200, 30]}
          initialSnap={0}
          enabledGestureInteraction={true}
          enabledBottomClamp={true}
          borderRadius={25}
          renderContent={this.renderContent}
        /> */}
      </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
```

```
    flex: 1,
    padding: 24,
    backgroundColor: '#eaeaea',
  },
  buttonStyle: {
    width: 80,
    borderColor: 'gray',
    borderWidth: 1,
    right: 0,
    bottom: 0,
    position: 'absolute',
    backgroundColor: 'transparent',
  },
  crossButtonStyle: {
    width: 80,
    borderColor: 'gray',
    borderWidth: 1,
    margin: 20,
    height: 10,
    flex: 1,
    backgroundColor: 'transparent',
  },
  searchBox: {
    height: 40,
    borderColor: 'gray',
    borderWidth: 2,
    margin: 8,
    backgroundColor: 'white',
  },
  searchBoxStop: {
    height: 40,
    borderColor: 'gray',
    borderWidth: 2,
    margin: 8,
    flex: 1,
    backgroundColor: 'white',
  },
  searchContainer: {
    backgroundColor: 'white',
    // backgroundColor: '#EBF5FB',
  },
  panelHandle: {
    width: 40,
    height: 8,
    borderRadius: 4,
    backgroundColor: '#00000040',
    marginBottom: 10,
    marginLeft: 170,
  },
});

export default App;
```

6.2 Order Detail

```
  import React, {Component} from 'react';
import {
  Text,
  View,
  TextInput,
  StyleSheet,
  Button,
  ScrollView,
} from 'react-native';
import {Card} from 'react-native-elements';

import AsyncStorage from '@react-native-async-storage/async-storage';

class orderDetails extends Component {
  state = {
    sourceMobile: null,
    destMobile: null,
    dest2Mobile: null,
    dest3Mobile: null,

    sourceAddressNote: '',
    destAddressNote: '',
    dest2AddressNote: '',
    dest3AddressNote: '',
  };

  // saveOrder = async () => {
  //   const value = await AsyncStorage.getItem('token');
  //   console.log('$$$$$$$$$$$$$$$$$$.....', value);

  //   const body = JSON.stringify({
  //     packageName: this.state.packageName,
  //     source: this.state.source,
  //     destination: this.state.destination,
  //     dimLength: this.state.dimLength,
  //     dimWidth: this.state.dimWidth,
  //     dimHeight: this.state.dimHeight,
  //     weight: this.state.weight,
  //   });

  //   const headers = {
  //     Accept: 'application/json',
  //     'Content-Type': 'application/json',
  //     Authorization: 'Bearer ' + value,
  //   };

  //   const settings = {
  //     method: 'POST',
  //     headers,
  //     body,
  //   };

  //   try {
  //     const fetchResponse = await fetch(
  //       `http://192.168.0.108:3000/postOrder`,
  //       settings,
  //     );
  //     const data = await fetchResponse.json();
  //     console.log('+++++', data);
```

```
//       if (data.token) {
//         this.onOrderSuccess(data);
//       }

//       return data; // your response data
//     } catch (e) {
//       console.log(e);
//       return e; // handle your error here
//     }
// };
// onOrderSuccess = async (data) => {
//   console.log('Order Success', data);
//   this.props.navigation.navigate('YourOrders', {
//     packageName: this.state.packageName,
//   });

//   const value = await AsyncStorage.getItem('token');
//   console.log('$$$$$$$$$$$$$$$$$$---', value);
// };
navigateToSchedule(
  sourceName,
  destName,
  dest2Name,
  dest3Name,
  sourceCordin,
  destCord,
  dest2Cord,
  dest3Cord,
  noStops,
) {
  console.log('$$$$$$$$$$$$$$$$$$.....', noStops);

  this.props.navigation.navigate('ScheduleOrder', {
    sourceMobile: this.state.sourceMobile,
    destMobile: this.state.destMobile,
    dest2Mobile: this.state.dest2Mobile,
    dest3Mobile: this.state.dest3Mobile,
    sourceAddressNote: this.state.sourceAddressNote,
    destAddressNote: this.state.destAddressNote,
    dest2AddressNote: this.state.dest2AddressNote,
    dest3AddressNote: this.state.dest3AddressNote,
    sourceName: sourceName,
    destName: destName,
    dest2Name: dest2Name,
    dest3Name: dest3Name,
    sourceCord: sourceCordin,
    destCord,
    dest2Cord,
    dest3Cord,
    noStops,
  });
}

render() {
  const sourceName = this.props.navigation.getParam(
    'sourceName',
    'Your Loaction',
  );
  const destName = this.props.navigation.getParam('destName', 'Destination');
  const dest2Name = this.props.navigation.getParam(
    'dest2Name',
```

```
      'Destination2',
    );
    const dest3Name = this.props.navigation.getParam(
      'dest3Name',
      'Destination3',
    );

    const mapRegion = this.props.navigation.getParam('mapRegion', null);
    const sourceCord = this.props.navigation.getParam('sourceCord', null);
    const destCord = this.props.navigation.getParam('destCord', null);
    const dest2Cord = this.props.navigation.getParam('dest2Cord', null);
    const dest3Cord = this.props.navigation.getParam('dest3Cord', null);

    const noStops = this.props.navigation.getParam('noStops', 2);

    return (
      <View style={styles.container}>
        <ScrollView>
          <Text style={styles.headerText}>Address Details</Text>
          <Card style={{borderRadius: 8}}>
            <Text>{sourceName}</Text>
            <TextInput
              keyboardType="numeric"
              maxLength={10}
              style={styles.textBox}
              onChangeText={(num) => {
                this.setState({sourceMobile: num});
              }}
              placeholder="Mobile No"></TextInput>
            <TextInput
              style={styles.multiLineBox}
              multiline={true}
              onChangeText={(text) => {
                this.setState({sourceAddressNote: text});
              }}
              placeholder="Source Address Note (Ex.Flat No., Compound, Street
                ...)"></TextInput>
          </Card>
          <Card>
            <Text>{destName}</Text>
            <TextInput
              keyboardType="numeric"
              maxLength={10}
              style={styles.textBox}
              onChangeText={(num) => {
                this.setState({destMobile: num});
              }}
              placeholder="Receivers Mobile No"></TextInput>
            <TextInput
              style={styles.multiLineBox}
              multiline={true}
              onChangeText={(text) => {
                this.setState({destAddressNote: text});
              }}
              placeholder="Destination Address Note (Ex.Flat No., Compound,
                Street ...)"></TextInput>
          </Card>
          {dest2Name != 'Destination2' && (
            <Card>
              <Text>{dest2Name}</Text>
              <TextInput
```

```
            keyboardType="numeric"
            maxLength={10}
            style={styles.textBox}
            onChangeText={(num) => {
              this.setState({dest2Mobile: num});
            }}
            placeholder="Receivers Mobile No"></TextInput>
          <TextInput
            style={styles.multiLineBox}
            multiline={true}
            onChangeText={(text) => {
              this.setState({dest2AddressNote: text});
            }}
            placeholder="Destination Address Note (Ex.Flat No., Compound,
              Street ...)"></TextInput>
        </Card>
      )}
      {dest3Name != 'Destination3' && (
        <Card>
          <Text>{dest3Name}</Text>
          <TextInput
            keyboardType="numeric"
            maxLength={10}
            style={styles.textBox}
            onChangeText={(num) => {
              this.setState({dest3Mobile: num});
            }}
            placeholder="Receivers Mobile No"></TextInput>
          <TextInput
            style={styles.multiLineBox}
            multiline={true}
            onChangeText={(text) => {
              this.setState({dest3AddressNote: text});
            }}
            placeholder="Destination Address Note (Ex.Flat No., Compound,
              Street ...)"></TextInput>
        </Card>
      )}
      {/* <Text style={styles.headerText}>Package Dimensions</Text> */}
      {/* <TextInput
        style={styles.textBox}
        onChangeText={(text) => {
          this.setState({dimLength: text});
        }}
        placeholder="Length"></TextInput>
      <TextInput
        style={styles.textBox}
        onChangeText={(text) => {
          this.setState({dimWidth: text});
        }}
        placeholder="Width"></TextInput>
      <TextInput
        style={styles.textBox}
        onChangeText={(text) => {
          this.setState({dimHeight: text});
        }}
        placeholder="Height"></TextInput>
      <TextInput
        style={styles.textBox}
        onChangeText={(text) => {
          this.setState({weight: text});
```

```
            }}
            placeholder="Weight Approx"></TextInput> */}
          {/* <TextInput
            style={styles.textBox}
            onChangeText={(text) => {
              this.setState({packageName: text});
            }}
            placeholder="Give a name to your Package"></TextInput> */}
        </ScrollView>
        <View
          style={{
            width: 90,
            backgroundColor: 'transparent',
            position: 'absolute',
            top: '85%',
            left: '70%',
            zIndex: 10,
          }}>
          <Button
            onPress={() => {
              console.log('nostops: ', noStops);
              if (sourceName == 'Your Location') {
                var sourceCordin = {
                  lat: mapRegion.longitude,
                  lon: mapRegion.latitude,
                };

                this.navigateToSchedule(
                  sourceName,
                  destName,
                  dest2Name,
                  dest3Name,

                  sourceCordin,
                  destCord,
                  dest2Cord,
                  dest3Cord,
                  noStops,
                );
              } else {
                var sourceCordin = sourceCord;
                console.log('sourceCord:', sourceCord);
                console.log('sourceCord:', sourceCordin);
                this.navigateToSchedule(
                  sourceName,
                  destName,
                  dest2Name,
                  dest3Name,

                  sourceCordin,
                  destCord,
                  dest2Cord,
                  dest3Cord,
                  noStops,
                );
              }
            }}
            title="Next"></Button>
        </View>
      </View>
    );
```

```
  }
}

const styles = StyleSheet.create({
  container: {
    backgroundColor: '#EBF5FB',
    //backgroundColor: '#E9967A',
    height: '100%',
  },
  textBox: {
    height: 40,
    borderColor: 'gray',
    borderWidth: 1,

    margin: 20,
  },
  multiLineBox: {
    // height: 40,
    borderColor: 'gray',
    borderWidth: 1,

    marginLeft: 20,
    marginRight: 20,
  },
  headerText: {
    textAlign: 'center',
    margin: 10,
  },
  buttonSTyle: {
    width: 40,
    borderColor: 'gray',
    borderWidth: 1,

    margin: 20,
  },
});

export default orderDetails;
```

6.3 Schedule Order

```
  import React, {Component} from 'react';
import {Text, View, TextInput, StyleSheet, ScrollView} from 'react-native';
import {Card, Slider, Button, ButtonGroup} from 'react-native-elements';
import {Animated} from 'react-native';
import Icon from 'react-native-vector-icons/Ionicons';
import DatePicker from 'react-native-date-picker';

import AsyncStorage from '@react-native-async-storage/async-storage';
import {Directions} from 'react-native-gesture-handler';

class scheduleScreen extends Component {
  constructor() {
    super();
    this.state = {
      weightValue: 1,
      sourceMobile: null,
      destMobile: null,
      dest2Mobile: null,
      dest3Mobile: null,

      sourceName: '',
      destName: '',
      dest2Name: '',
      dest3Name: '',

      sourceAddressNote: '',
      destAddressNote: '',
      dest2AddressNote: '',
      dest3AddressNote: '',

      sourceCord: {lat: null, lon: null},
      destCord: {lat: null, lon: null},
      dest2Cord: {lat: null, lon: null},
      dest3Cord: {lat: null, lon: null},

      selectedIndex: 2,
      scheduleDateTime: new Date(),
      currentTime: null,
      packageName: '',
      amount: 50,
    };
  }

  saveOrder = async (
    sourceName,
    destName,
    dest2Name,
    dest3Name,
    noStops,
    sourceCord,
    destCord,
    dest2Cord,
    dest3Cord,
    sourceAddressNote,
    destAddressNote,
    dest2AddressNote,
    dest3AddressNote,
    sourceMobile,
    destMobile,
```

```
    dest2Mobile,
    dest3Mobile,
  ) => {
    const value = await AsyncStorage.getItem('token');
    console.log('$$$$$$$$$$$$$$$$$$.....', value);
    console.log('$$$$$$$$$$$$$$$$$$.....', noStops);

    var details = {
      sourceName,
      destName,
      dest2Name,
      dest3Name,
      noStops,
      sourceCord,
      destCord,
      dest2Cord,
      dest3Cord,
      sourceAddressNote,
      destAddressNote,
      dest2AddressNote,
      dest3AddressNote,
      sourceMobile,
      destMobile,
      dest2Mobile,
      dest3Mobile,
    };

    const body = JSON.stringify({
      packageName: this.state.packageName,
      sourceName,
      destName,
      dest2Name,
      dest3Name,

      sourceCord,
      destCord,
      dest2Cord,
      dest3Cord,

      sourceAddressNote,
      destAddressNote,
      dest2AddressNote,
      dest3AddressNote,

      sourceMobile,
      destMobile,
      dest2Mobile,
      dest3Mobile,

      noStops,

      sceduleType: this.state.selectedIndex,
      scheduleDateTime: this.state.scheduleDateTime,
      amount: this.state.amount,
      weight: this.state.weightValue,
    });

    const headers = {
      Accept: 'application/json',
      'Content-Type': 'application/json',
      Authorization: 'Bearer ' + value,
```

```
    };

    const settings = {
      method: 'POST',
      headers,
      body,
    };

    try {
      const fetchResponse = await fetch(
        `http://192.168.0.107:3000/postOrder`,
        settings,
      );
      const data = await fetchResponse.json();
      console.log('+++++', data);
      if (data.packageName) {
        this.onOrderSuccess(data);
      }

      return data; // your response data
    } catch (e) {
      console.log(e);
      return e; // handle your error here
    }
  };
  onOrderSuccess = async (data) => {
    console.log('Order Success', data);
    this.props.navigation.navigate('YourOrders', {
      packageName: this.state.packageName,
    });
    // if(this.props.selectedIndex == 1){
    //   this.props.navigation.navigate('YourOrders', {
    //     packageName: this.state.packageName,
    //   });
    // }
    // else if(this.props.selectedIndex == 0){
    //   this.props.navigation.navigate('Tracking', {
    //     data: details,
    //   });
    // }

    const value = await AsyncStorage.getItem('token');
    console.log('$$$$$$$$$$$$$$$$$$---', value);
  };

  updateIndex = async (selectedIndex) => {
    await this.setState({selectedIndex});
    console.log(selectedIndex);
    console.log(this.state.selectedIndex);
  };

  navigateToSchedule(
    sourceName,
    destName,
    dest2Name,
    dest3Name,
    sourceCord,
    destCord,
    dest2Cord,
    dest3Cord,
    noStops,
```

```
  ) {
    this.props.navigation.navigate('YourOrders', {
      sourceMobile: this.state.sourceMobile,
      destMobile: this.state.destMobile,
      dest2Mobile: this.state.dest2Mobile,
      dest3Mobile: this.state.dest3Mobile,
      sourceAddressNote: this.state.sourceAddressNote,
      destAddressNote: this.state.destAddressNote,
      dest2AddressNote: this.state.dest2AddressNote,
      dest3AddressNote: this.state.dest3AddressNote,
      sourceName: sourceName,
      destName: destName,
      dest2Name: dest2Name,
      dest3Name: dest3Name,
      sourceCord,
      destCord,
      dest2Cord,
      dest3Cord,
      noStops: noStops,
    });
  }
  componentDidMount = async () => {
    // if (this.state.initialLoad) {
    var date = new Date();
    //var timePostHr = date.getHours();
    await this.setState({currentTime: date.getHours()});
    console.log('---', this.state.currentTime);
    // }
  };

  render() {
    const sourceName = this.props.navigation.getParam(
      'sourceName',
      'Your Loaction',
    );
    const destName = this.props.navigation.getParam('destName', 'Destination');
    const dest2Name = this.props.navigation.getParam(
      'dest2Name',
      'Destination2',
    );
    const dest3Name = this.props.navigation.getParam(
      'dest3Name',
      'Destination3',
    );
    const noStops = this.props.navigation.getParam('noStops', 2);

    const sourceCord = this.props.navigation.getParam('sourceCord', null);
    const destCord = this.props.navigation.getParam('destCord', null);
    const dest2Cord = this.props.navigation.getParam('dest2Cord', null);
    const dest3Cord = this.props.navigation.getParam('dest3Cord', null);

    const sourceAddressNote = this.props.navigation.getParam(
      'sourceAddressNote',
      null,
    );
    const destAddressNote = this.props.navigation.getParam(
      'destAddressNote',
      null,
    );
    const dest2AddressNote = this.props.navigation.getParam(
      'dest2AddressNote',
```

```
      null,
    );
    const dest3AddressNote = this.props.navigation.getParam(
      'dest3AddressNote',
      null,
    );

    const sourceMobile = this.props.navigation.getParam('sourceMobile', null);
    const destMobile = this.props.navigation.getParam('destMobile', null);
    const dest2Mobile = this.props.navigation.getParam('dest2Mobile', null);
    const dest3Mobile = this.props.navigation.getParam('dest3Mobile', null);

    // var date = new Date();
    // var month = new Date().getMonth();
    // var year = new Date().getFullYear();
    // var dateToday = new Date().getDate();
    // var timePostHr = date.getHours() + 1;
    // console.log(typeof date);
    // console.log(date);
    // console.log(date.getHours() + 1);
    // console.log(date.getUTCDate());
    // var dateForm =
    //   year.toString() + '-' + month.toString() + '-' + dateToday.toString();
    //console.log(dateForm);
    //console.log(typeof dateForm);

    return (
      <View style={styles.container}>
        <ScrollView>
          <Text style={styles.headerText}>schedule Details</Text>
          <Card>
            <TextInput
              style={{borderBottomWidth: 1, height: 40}}
              placeholder="Package Name"
              onChangeText={(text) => {
                this.setState({packageName: text});
              }}></TextInput>
          </Card>
          <Card style={{borderRadius: 8}}>
            <Text>Weight: Upto {this.state.weightValue} Kg</Text>
            <Slider
              value={this.state.weightValue}
              //onValueChange={setValue}
              maximumValue={20}
              minimumValue={1}
              step={5}
              onValueChange={(value) => {
                if (value == 6 || value == 11 || value == 16) {
                  this.setState({weightValue: value - 1});
                } else if (value == 1 || value == 20) {
                  this.setState({weightValue: value});
                }
              }}
              thumbStyle={{
                height: 25,
                width: 25,
                backgroundColor: '#F1948A',
              }}
            />
          </Card>

```

```
304          <Card>
305            {/* <View style={{flexDirection: 'row'}}>
306              <Button title="Book Now" type="outline" />
307              <Button title="Schedule" type="solid" />
308            </View> */}
309
310            <ButtonGroup
311              selectedIndex={this.state.selectedIndex}
312              onPress={this.updateIndex}
313              buttons={['Book Now', 'Schedule PickUp']}
314              containerStyle={{height: 50}}
315            />
316            <View>
317              {this.state.selectedIndex == 1 && (
318                <DatePicker
319                  //date={new Date()}
320                  date={this.state.scheduleDateTime}
321                  timeZoneOffsetInMinutes={390}
322                  onDateChange={(date) => {
323                    this.setState({scheduledDateTime: date});
324
325                    console.log(typeof date);
326                  }}
327                />
328              )}
329            </View>
330          </Card>
331
332          <Card>
333            <Text>Total:      50</Text>
334          </Card>
335          {/* <Text style={styles.headerText}>Package Dimensions</Text> */}
336          {/* <TextInput
337            style={styles.textBox}
338            onChangeText={(text) => {
339              this.setState({dimLength: text});
340            }}
341            placeholder="Length"></TextInput>
342          <TextInput
343            style={styles.textBox}
344            onChangeText={(text) => {
345              this.setState({dimWidth: text});
346            }}
347            placeholder="Width"></TextInput>
348          <TextInput
349            style={styles.textBox}
350            onChangeText={(text) => {
351              this.setState({dimHeight: text});
352            }}
353            placeholder="Height"></TextInput>
354          <TextInput
355            style={styles.textBox}
356            onChangeText={(text) => {
357              this.setState({weight: text});
358            }}
359            placeholder="Weight Approx"></TextInput> */}
360          {/* <TextInput
361            style={styles.textBox}
362            onChangeText={(text) => {
363              this.setState({packageName: text});
364            }}
```

```
            placeholder="Give a name to your Package"></TextInput> */}
        </ScrollView>
        <View
          style={{
            width: 370,
            backgroundColor: 'transparent',
            position: 'absolute',
            top: '85%',
            margin: 20,
            // left: '10%',
            //zIndex: 10,
          }}>
          <Button
            onPress={() => {
              this.saveOrder(
                sourceName,
                destName,
                dest2Name,
                dest3Name,
                noStops,
                sourceCord,
                destCord,
                dest2Cord,
                dest3Cord,
                sourceAddressNote,
                destAddressNote,
                dest2AddressNote,
                dest3AddressNote,
                sourceMobile,
                destMobile,
                dest2Mobile,
                dest3Mobile,
              );
              //    this.navigateToSchedule(
              //      sourceName,
              //      destName,
              //      dest2Name,
              //      dest3Name,
              //      noStops,
              //      sourceCord,
              //      destCord,
              //      dest2Cord,
              //      dest3Cord,
              //    );
            }}
            title="Place Order"></Button>
        </View>
      </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    backgroundColor: '#EBF5FB',
    //backgroundColor: '#E9967A',
    height: '100%',
  },
  textBox: {
    height: 40,
    borderColor: 'gray',
```

```
    borderWidth: 1,

    margin: 20,
  },
  multiLineBox: {
    // height: 40,
    borderColor: 'gray',
    borderWidth: 1,

    marginLeft: 20,
    marginRight: 20,
  },
  headerText: {
    textAlign: 'center',
    margin: 10,
  },
  buttonSTyle: {
    width: 40,
    borderColor: 'gray',
    borderWidth: 1,

    margin: 20,
  },
});

export default scheduleScreen;
```

6.4 Your Order

```
  import React, {Component} from 'react';
import {
  Text,
  View,
  TextInput,
  StyleSheet,
  Button,
  FlatList,
} from 'react-native';

class yourOrders extends Component {
  state = {packageName: ''};
  packName = () => {
    var pName = this.props.navigation.getParam('packageName');
    this.setState({packageName: pName});
  };

  render() {
    const pName = this.props.navigation.getParam('packageName', 'Package1');
    console.log(pName);
    var date = new Date().getDate();
    var hours = new Date().getHours(); //To get the Current Hours
    var min = new Date().getMinutes();

    return (
      <View>
        <Text style={styles.headerText}>Your Orders</Text>

        <FlatList
          data={'1'}
          style={styles.container}
          renderItem={({item}) => (
            <View style={styles.rowStyle}>
              <Text style={styles.rightText}>{pName}</Text>
              <Text style={styles.leftText}>{hours}</Text>
              <Text>:{min}</Text>
            </View>
          )}
          //keyExtractor={item => item.id}
        />
        {/* <FlatList
            data={this.state.placeResult}
            renderItem={({item}) => (
              <TouchableHighlight onPress={() => this.getCoord(item.key)}>
                <Text key={item.key.toString()} bottomDivider>
                  {item.name}
                </Text>
              </TouchableHighlight>
            )}
          /> */}
      </View>
    );
  }
}
const styles = StyleSheet.create({
  container: {
    padding: 10,
    marginTop: 3,
    backgroundColor: '#AED6F1',
```

```
    // backgroundColor: '#d9f9b1',

    // alignItems: 'center',
  },
  rightText: {
    marginRight: 'auto',
  },
  leftText: {
    textAlign: 'right',
  },
  headerText: {
    textAlign: 'center',
    margin: 10,
  },
  rowStyle: {
    flexDirection: 'row',
  },
});

export default yourOrders;
```

6.5 Admin Navigator

```
  //import { StatusBar } from "expo-status-bar";
import React from 'react';
import {StyleSheet, Text, View} from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';
import {createAppContainer} from 'react-navigation';
import {createStackNavigator} from 'react-navigation-stack';
import {createSwitchNavigator} from 'react-navigation';
import {createDrawerNavigator} from 'react-navigation-drawer';
import {createBottomTabNavigator} from 'react-navigation-tabs';
import adminDashboard from './src/adminScreens/adminDashboard';
import acceptDriver from './src/adminScreens/acceptDriver';
import viewDriver from './src/adminScreens/viewDriver';
import viewDetailDriver from './src/adminScreens/viewDriverDetails';
import verifyDetailDriver from './src/adminScreens/acceptDriverDetail';
import drawerContentComponents from './src/screens/drawerScreen';
//import orderDetails from './src/screens/orderDetails';
import yourOrders from './src/screens/yourOrders';
import Icon from 'react-native-vector-icons/Ionicons';

const detailsStackNavigator = createStackNavigator({
  adminList: {
    screen: adminDashboard,
    navigationOptions: {
      headerShown: false,
    },
  },
  Accept: {
    screen: acceptDriver,
  },
  ViewDriver: {
    screen: viewDriver,
  },
  ViewDetailDriver: {
    screen: viewDetailDriver,
  },
  VerifyDetail: {
    screen: verifyDetailDriver,
  },
});

const adminTabNavigator = createBottomTabNavigator({
  BookCourier: detailsStackNavigator,

  Schedule: {screen: yourOrders},
});

// const AppDrawerNavigator = createDrawerNavigator({
//   contentComponent: drawerContentComponents,
// });

const adminStackNavigator = createStackNavigator({
  AdminCourio: {
    screen: adminTabNavigator,
    navigationOptions: ({navigation}) => {
      // headerShown: false;
      return {
        headerLeft: (
          <Icon
            style={{paddingLeft: 10}}
```

```
60              onPress={() => navigation.openDrawer()}
61              name="menu-sharp"
62              size={30}
63            />
64          ),
65        };
66      },
67    },
68  });
69
70  const adminDrawer = createDrawerNavigator({
71    adminMain: {
72      screen: adminStackNavigator,
73    },
74
75    Account: drawerContentComponents,
76  });
77
78  export default adminDrawer;
```

6.6 Sign Up Screen

```
import React, {Component} from 'react';
import {
  Text,
  View,
  TextInput,
  StyleSheet,
  Button,
  Picker,
  ScrollView,
} from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';

class SignupView extends Component {
  constructor(props) {
    super(props);

    this.state = {
      name: '',
      email: '',
      password: '',
      confPass: '',
      phoneNo: '',
      userType: '',
      validEmail: true,
      validPassword: true,
      validConfPass: true,
      validPhoneNo: true,

      validInput: true,
    };
  }
  validateEmail(email) {
    var re = /\S+@\S+\.\S+/;
    console.log('Validating', re.test(email));

    if (re.test(email) != true) {
      this.setState({validEmail: false});
    } else {
      this.setState({validEmail: true});
    }

    return re.test(email);
  }
  validatePhoneNo(phoneNo) {
    if (this.state.phoneNo.length != 10) {
      console.log('Invalid PhoneNo.');
      this.setState({validPhoneNo: false});
      return false;
    } else {
      this.setState({validPhoneNo: true});
    }
  }
  // validateUserType(userType) {
  //   if (this.state.userType.length > 0) {
  //     console.log('Invalid userType.');
  //     this.setState({validUserType: false});
  //     return false;
  //   } else {
  //     this.setState({validUserType: true});
```

```
//    }
// }
validatePassword(password) {
  if (this.state.password.length < 8) {
    console.log('Minimum Password length should be 8');
    this.setState({validPassword: false});
    return false;
  } else {
    this.setState({validPassword: true});
  }
}
validateConfPass(confPass) {
  if (this.state.password != this.state.confPass) {
    console.log('Password not Matched');
    this.setState({validConfPass: false});
    return false;
  } else {
    this.setState({validConfPass: true});
  }
}
validateForm() {
  console.log(
    '------------------',
    this.state.name,
    this.state.email,
    this.state.phoneNo,
    this.state.password,
    this.state.confPass,
  );
  var val = this.validateEmail(this.state.email);

  if (
    this.validEmail == false ||
    this.validPhoneNo == false ||
    this.validPassword == false ||
    this.validConfPass == false
  ) {
    console.log('Error');
    return 0;
  } else {
    if (this.state.userType == 'user') {
      this.sendToDB();
    }
  }
}
sendToDB = async () => {
  console.log('------------------');
  const body = JSON.stringify({
    name: this.state.name,
    email: this.state.email,
    password: this.state.password,
    phoneNo: this.state.phoneNo,
  });

  const headers = {
    Accept: 'application/json',
    'Content-Type': 'application/json',
  };

  const settings = {
    method: 'POST',
```

```
      headers,
      body,
    };

    try {
      const fetchResponse = await fetch(
        `http://192.168.0.108:3000/signup`,
        settings,
      );
      const data = await fetchResponse.json();
      console.log('+++++', data);
      if (data.token) {
        this.onSignupSuccess(data);
      }

      return data; // your response data
    } catch (e) {
      console.log(e);
      return e; // handle your error here
    }
  };

  onSignupSuccess = async (data) => {
    console.log('SignUp Success', data);
    await AsyncStorage.setItem('token', data.token);
    this.props.navigation.navigate('Dashboard');

    const value = await AsyncStorage.getItem('token');
    console.log('$$$$$$$$$$$$$$$$$$', value);
  };

  render() {
    return (
      <View>
        <ScrollView>
          <Text style={styles.appName}>Courio</Text>
          <Text style={styles.headerText}>Signup</Text>
          <Text style={styles.pickerHeader}>User Type</Text>
          <View style={styles.pickerStyle}>
            <Picker
              selectedValue={this.state.userType}
              style={styles.pickerStyle}
              mode="dropdown"
              onValueChange={(itemValue, itemIndex) => {
                this.setState({userType: itemValue});
                //this.validateUserType(this.state.userType);
              }}>
              <Picker.Item label="Customer" value="user" />
              <Picker.Item label="Driver" value="driver" />
            </Picker>
          </View>

          <TextInput
            placeholder="Name"
            style={styles.textBox}
            onChangeText={(text) => {
              this.setState({name: text});
            }}></TextInput>

          <TextInput
            placeholder="Email"
```

```
            style={styles.textBox}
            onChangeText={(text) => {
              this.setState({email: text});
            }}
            onEndEditing={(text) => {
              this.validateEmail(this.state.email);
            }}></TextInput>
          <View>
            {this.state.validEmail ? null : <Text>Invalid Email</Text>}
          </View>

          <TextInput
            placeholder="Phone No."
            style={styles.textBox}
            onChangeText={(text) => {
              this.setState({phoneNo: text});
            }}
            onEndEditing={(text) => {
              this.validatePhoneNo(text);
            }}></TextInput>
          <View>
            {this.state.validPhoneNo ? null : <Text>Invalid Phone No</Text>}
          </View>
          <TextInput
            secureTextEntry={true}
            placeholder="Password"
            style={styles.textBox}
            onChangeText={(text) => {
              this.setState({password: text});
            }}
            onEndEditing={(text) => {
              this.validatePassword(text);
            }}></TextInput>
          <View>
            {this.state.validPassword ? null : (
              <Text>Password length must be greater than equal to 8</Text>
            )}
          </View>
          <TextInput
            secureTextEntry={true}
            placeholder="Confirm Password"
            style={styles.textBox}
            onChangeText={(text) => {
              this.setState({confPass: text});
            }}
            onEndEditing={(text) => {
              this.validateConfPass(text);
            }}></TextInput>
          <View>
            {this.state.validConfPass ? null : (
              <Text>Password didn't match</Text>
            )}
          </View>
        </ScrollView>
        <View
          style={{
            width: 90,
            backgroundColor: 'transparent',
            // position: 'absolute',
            top: '10%',
            left: '40%',
```

```
            zIndex: 10,
          }}>
          <Button
            onPress={() => {
              // this.props.navigation.navigate('Dashboard');
              console.log(
                '------------------',
                this.state.name,
                this.state.email,
                this.state.phoneNo,
                this.state.password,
                this.state.confPass,
              );
              this.validateForm();
            }}
            title="Register"></Button>
        </View>
      </View>
    );
  }
}
const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 24,
    backgroundColor: '#eaeaea',
  },
  headerText: {
    textAlign: 'center',
    margin: 10,
  },
  appName: {
    textAlign: 'center',
    margin: 20,
    fontSize: 20,
  },
  textBox: {
    height: 40,
    borderColor: '#e29578',
    backgroundColor: 'white',
    borderWidth: 1,
    margin: 20,
  },
  pickerHeader: {
    // marginLeft: 20,
    // marginRight: 20,
    margin: 16,
  },
  pickerStyle: {
    marginLeft: 20,
    marginRight: 20,

    borderColor: '#e29578',
    borderWidth: 1.5,
    color: '#168aad',
  },
  pickerItem: {
    marginLeft: 20,
    marginRight: 20,
  },
});
```

```
304
305 export default SignupView;
```

6.7 Swipe Screen

```
  import * as React from 'react';
import {StyleSheet, Text, View, Button} from 'react-native';
import Animated from 'react-native-reanimated';
import BottomSheet from 'reanimated-bottom-sheet';

export default function App() {
  const renderContent = () => (
    <View
      style={{
        backgroundColor: 'white',
        padding: 16,
        height: 450,
      }}>
      <Text>Swipe down to close</Text>
    </View>
  );

  const sheetRef = React.useRef(null);

  return (
    <>
      <View
        style={{
          flex: 1,
          backgroundColor: 'papayawhip',
          alignItems: 'center',
          justifyContent: 'center',
        }}>
        <Button
          title="Open Bottom Sheet"
          onPress={() => sheetRef.current.snapTo(0)}
        />
      </View>
      <BottomSheet
        ref={sheetRef}
        snapPoints={[450, 300, 100]}
        initialSnap={0}
        enabledGestureInteraction={true}
        enabledBottomClamp={true}
        borderRadius={50}
        renderContent={renderContent}
      />
    </>
  );
}
```

6.8 Welcome Screen

```
  import React, {Component} from 'react';
import {Text, View, TextInput, StyleSheet, Button} from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';

class Welcome extends Component {
  constructor(props) {
    super(props);

    this.state = {
      username: '',
      password: '',
    };
  }

  validateUser = async () => {
    console.log('**********', this.state.username, this.state.password);
    const body = JSON.stringify({
      email: this.state.username,
      password: this.state.password,
    });

    const headers = {
      Accept: 'application/json',
      'Content-Type': 'application/json',
      Authorization: 'Bearer',
    };

    const settings = {
      method: 'POST',
      headers,
      body,
    };

    try {
      const fetchResponse = await fetch(
        `http://192.168.0.108:3000/signin`,
        settings,
      );
      const data = await fetchResponse.json();
      console.log('+++++', data);
      if (data.token) {
        this.onLoginSuccess(data);
      }

      return data; // your response data
    } catch (e) {
      console.log(e);
      return e; // handle your error here
    }
  };
  onLoginSuccess = async (data) => {
    console.log('Login Success', data);
    await AsyncStorage.setItem('token', data.token);
    this.props.navigation.navigate('Dashboard');

    const value = await AsyncStorage.getItem('token');
    console.log('$$$$$$$$$$$$$$$$$$', value);
  };

```

```
  render() {
    return (
      <View>
        <Text style={styles.appName}>Courio</Text>
        <Text style={styles.headerText}>Email</Text>
        <TextInput
          placeholder="Username"
          style={styles.textBox}
          autoCapitalize="none"
          autoCorrect={false}
          onChangeText={(text) => {
            this.setState({username: text});
          }}></TextInput>
        <Text style={styles.headerText}>Password</Text>
        <TextInput
          secureTextEntry={true}
          placeholder="Password"
          style={styles.textBox}
          autoCapitalize="none"
          autoCorrect={false}
          onChangeText={(text) => {
            this.setState({password: text});
          }}></TextInput>
        <View
          style={{
            width: 90,
            backgroundColor: 'transparent',
            //position: 'absolute',
            top: '10%',
            left: '40%',
            zIndex: 10,
          }}>
          <Button
            onPress={() => {
              console.log(
                '------------------',
                this.state.username,
                this.state.password,
              );
              this.validateUser();
            }}
            title="Login"></Button>
        </View>
        <View>
          <Text
            onPress={() => this.props.navigation.navigate('Signup')}
            style={{color: 'blue', marginTop: 80}}>
            Don't have an account SignUp
          </Text>
        </View>
      </View>
    );
  }
}
const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 24,
    backgroundColor: '#eaeaea',
  },
  textBox: {
```

```
    height: 40,
    borderColor: 'gray',
    borderWidth: 1,

    margin: 20,
  },
  headerText: {
    textAlign: 'center',
    margin: 10,
  },
  appName: {
    textAlign: 'center',
    margin: 20,
    fontSize: 20,
  },
  buttonSTyle: {
    width: 40,
    borderColor: 'gray',
    borderWidth: 1,

    margin: 20,
  },
});

export default Welcome;
```

Chapter 7

System Testing

System testing is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.

7.1 Test Cases and Test Results

Test ID	Test Case Title	Test Condition	System Behavior	Expected Result
T01	Placing Order	data reflect in Driver app	Data reflected successfull	Data Reflected
T02	Live Tracking	Route Show properly	Route showed successfully	Route from sender to Reciever
T03	SMS Notification	SMS on User phone	SMS recieved successfully	SMS sent to User successfully

7.2 Test Case

Title: Placing Order
Description: A registered user should be able to successfully place an order

Precondition: the user must already be registered with app.
Assumption: Database should reflect properly.

Test Steps:

1. Open App
2. Login / Register
3. Place an Order

ir.aiktclibrary.org

4. Fill required details
5. Click 'Place Order'

Expected Result: A page displaying the Order ID and the assigned delivery executive details.

Actual Result:

A page displaying the Order ID and the assigned delivery executive details.

Title: Live Tracking

Description: A registered user should be able to track the package Live in application

Precondition: the user must already be registered with app and need to place the order which will be Active.

Assumption: Mapbox API should reflect.

Test Steps:

1. Open App
2. Login / Register
3. Place an Order
4. Fill required details
5. Click 'Place Order'
6. Click 'Track Order'

Expected Result: A page displaying the Order ID and the assigned delivery executive details with the Map of Live location of Package.

Actual Result:

A page displaying the Order ID and the assigned delivery executive details with the Map of Live location of Package.

7.2.1 Software Quality Attributes

Availability-1 : The system shall be available to users all the time.
Availability-2 : The system shall always have something to function and always pop up error messages in case of component failure.
Efficiency-1 : The system shall generate the correct text audio with an accuracy of 80 above.
Efficiency-2 : The system shall provide the right tools to support all its features.

Chapter 8

Screenshots of Project

8.1 Map - Selecting Pickup

Figure 8.1: Select Pickup

8.2 Search Destination

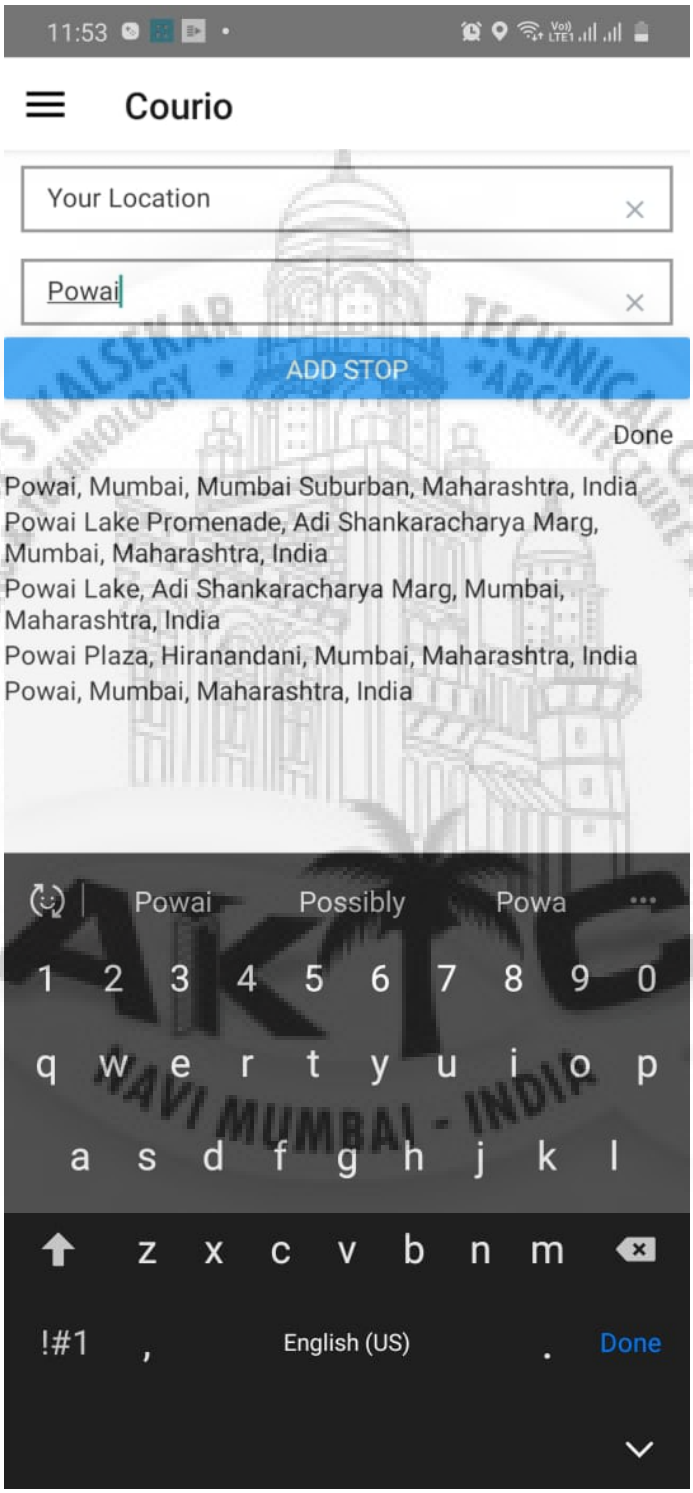


Figure 8.2: Search Destination

8.3 Add Multiple Delivery Points

Figure 8.3: Add Multiple Delivery Points

8.4 Map Route

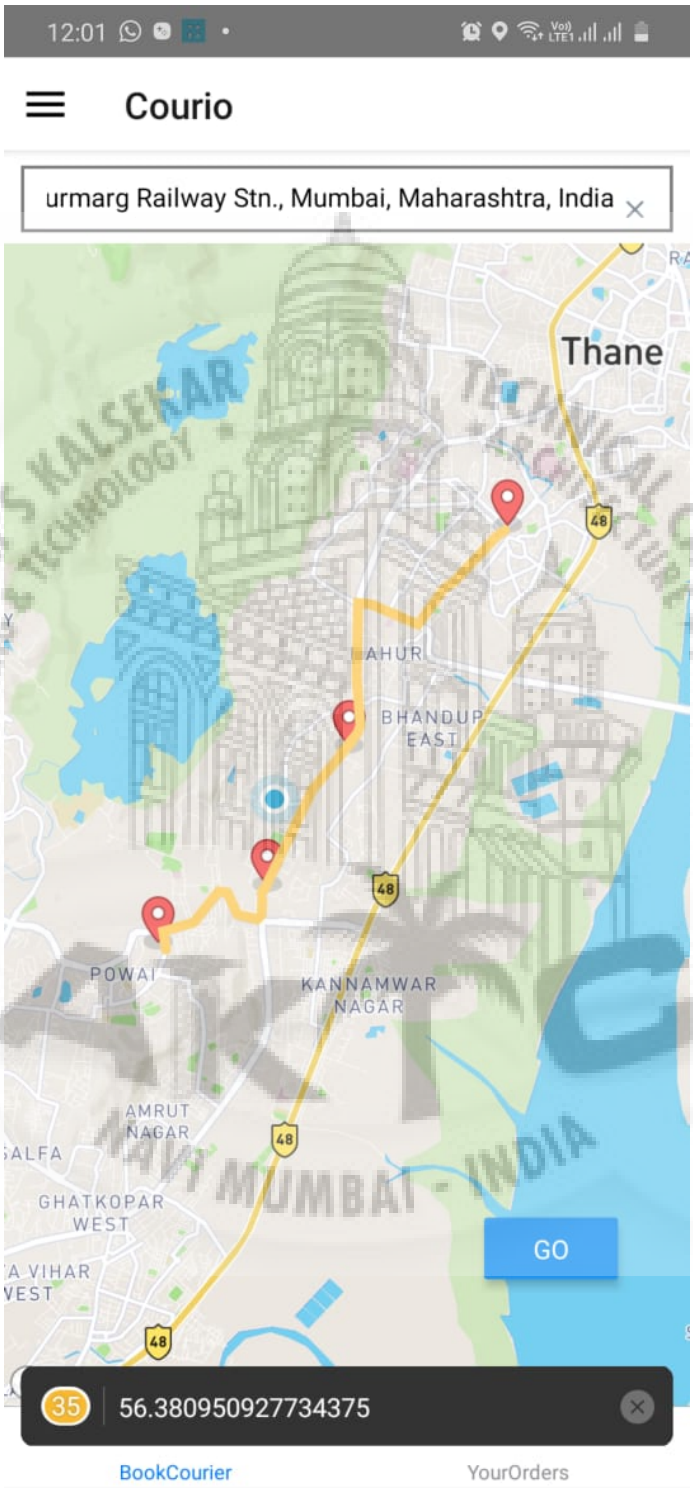


Figure 8.4: Map route

8.5 Add Address Details

Figure 8.5: Address Details

8.6 Add Package Details

Figure 8.6: Package Details

8.7 Driver - List of Assigned Order

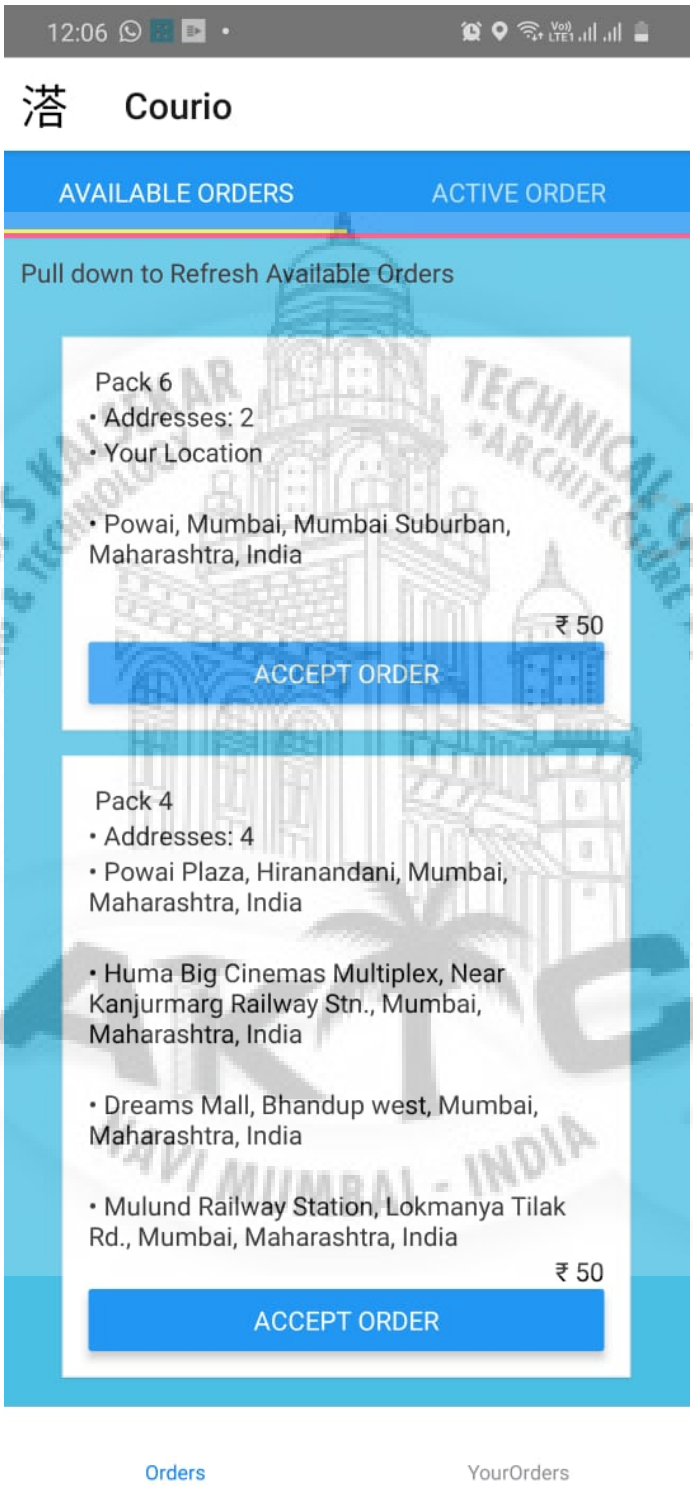


Figure 8.7: Assigned Orders

8.8 Driver - Order Route in Map with Order Details

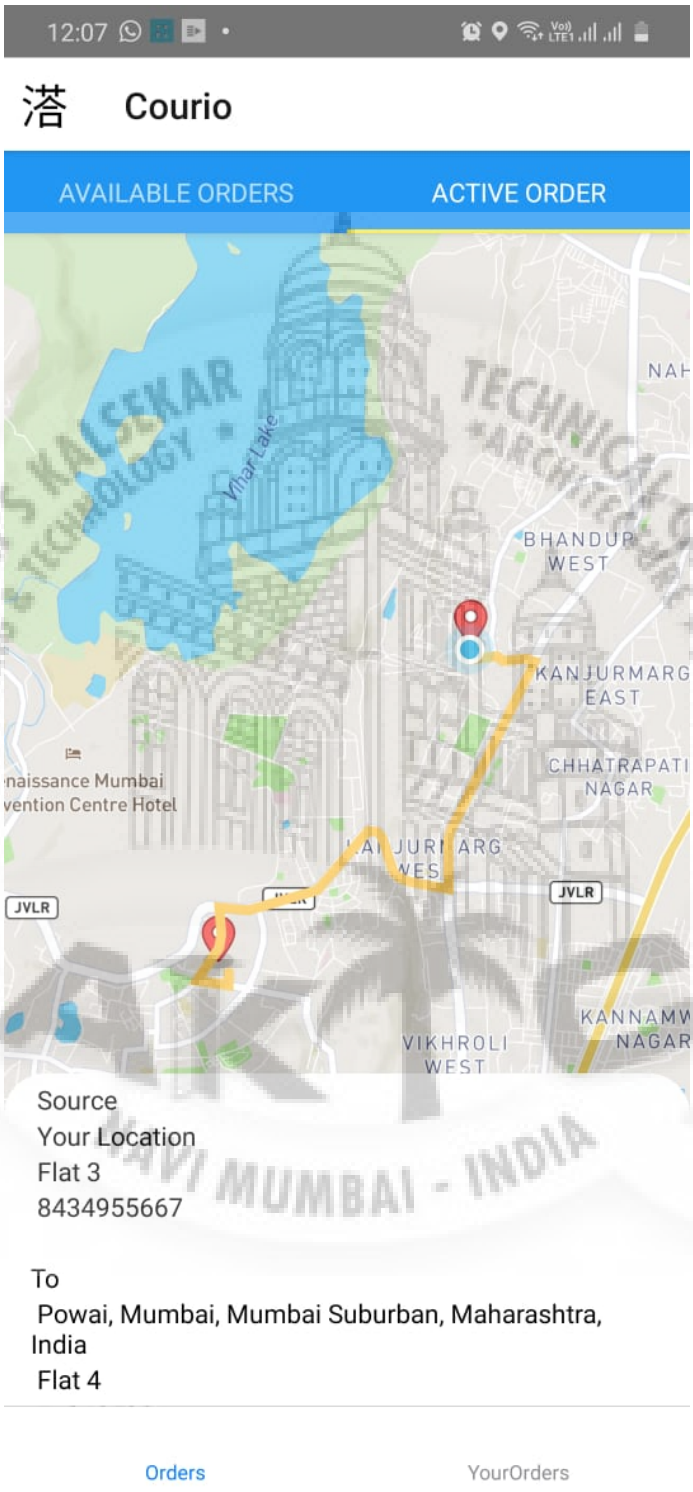


Figure 8.8: Order Route

Chapter 9

Conclusion and Future Scope

9.1 Conclusion

The project titled 'C2C Courier Service' was developed to the courier services and direction and with their help. The Team is working behind the project and all the necessary output will be created and The system will be found to be user-friendly with the help message for the customer.
The system will be implemented successfully. The manpower and working hours needed to operate the system was less and it is seen to be more secure. Thus, the Project will completed successful.

9.2 Future Scope

- Urbanization: Space as the scarcest resource
- E-commerce has a rapidly growing customer base
- New categories move online and new business models emerge

References

[1] *Courier Management System*; • Jomo Kenyatta University of Agriculture and Technology Computer Science https://www.docsity.com/en/courier-management-system/4315730/

[2] Use Mapbox GL JS in a React app Intermediate https://docs.mapbox.com/help/tutorials/use-mapbox-gl-js-with-react/