

**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**

**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**

# 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

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

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**

## 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

## Project I Approval for Bachelor of Engineering

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

Examiners

1. ....

2. ....

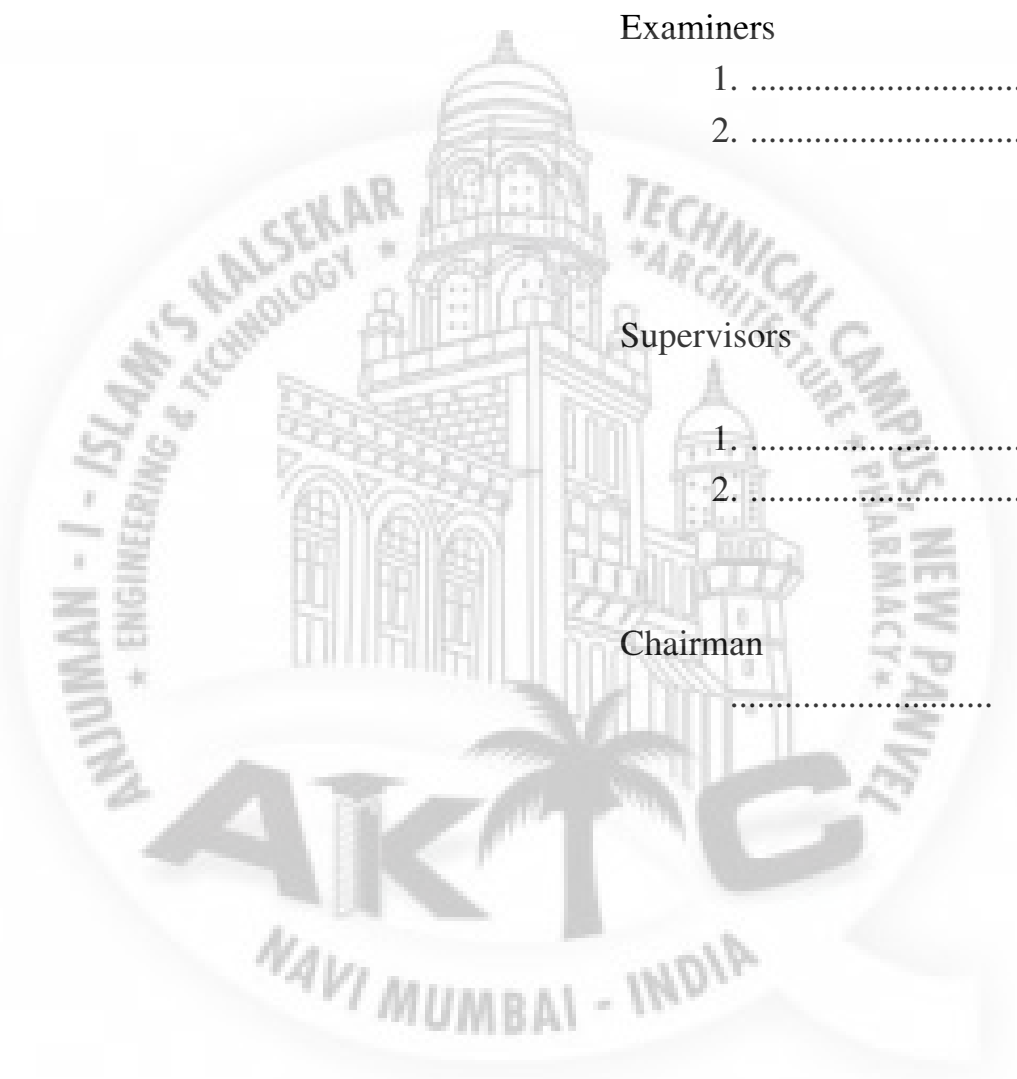
Supervisors

1. ....

2. ....

Chairman

.....



## 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

## 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 on-line 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 benefited 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

# Contents

Acknowledgement . . . . .	iii
Project I Approval for Bachelor of Engineering . . . . .	iv
Declaration . . . . .	v
Abstract . . . . .	vi
Table of Contents . . . . .	ix
<b>1 Introduction</b>	<b>2</b>
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
<b>2 Literature Survey</b>	<b>4</b>
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
<b>3 Project Planning</b>	<b>7</b>
3.1 Members and Capabilities . . . . .	7
3.2 Roles and Responsibilities . . . . .	7
3.3 Assumptions and Constraints . . . . .	7



3.4	Project Management Approach . . . . .	8
3.5	Ground Rules for the Project . . . . .	8
3.6	Project Budget . . . . .	8
3.7	Project Timeline . . . . .	9
<b>4</b>	<b>Software Requirements Specification</b>	<b>10</b>
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
<b>5</b>	<b>System Design</b>	<b>14</b>
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

<b>6</b>	<b>Implementation</b>	<b>24</b>
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
<b>7</b>	<b>System Testing</b>	<b>67</b>
7.1	Test Cases and Test Results . . . . .	67
7.2	Test Case . . . . .	67
7.2.1	Software Quality Attributes . . . . .	69
<b>8</b>	<b>Screenshots of Project</b>	<b>70</b>
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
<b>9</b>	<b>Conclusion and Future Scope</b>	<b>78</b>
9.1	Conclusion . . . . .	78
9.2	Future Scope . . . . .	78
	<b>References</b>	<b>78</b>

## List of Figures

2.1	<b>image</b>	5
5.1	<b>Sender</b>	15
5.2	<b>Receiver</b>	15
5.3	<b>Driver</b>	16
5.4	<b>Driver</b>	17
5.5	<b>order</b>	18
5.6	<b>order</b>	18
5.7	<b>Shipment</b>	18
5.8	<b>Billing</b>	19
5.9	<b>Billing</b>	19
5.10	<b>Tools</b>	19
5.11	<b>Tools</b>	19
5.12	<b>Tools</b>	19
5.13	<b>Support</b>	20
5.14	<b>Support</b>	20
5.15	<b>Return</b>	20
5.16	<b>Account Management</b>	21
5.17	<b>Inventory</b>	21
5.18	<b>UML</b>	22
5.19	<b>Sequence Diagram</b>	23
8.1	<b>Select Pickup</b>	70
8.2	<b>Search Destination</b>	71
8.3	<b>Add Multiple Delivery Points</b>	72
8.4	<b>Map route</b>	73
8.5	<b>Address Details</b>	74
8.6	<b>Package Details</b>	75
8.7	<b>Assigned Orders</b>	76
8.8	<b>Order Route</b>	77

## List of Tables

3.1	Table of Capabilities . . . . .	7
3.2	Table of Responsibilities . . . . .	7



# 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]

## 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.

## 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

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

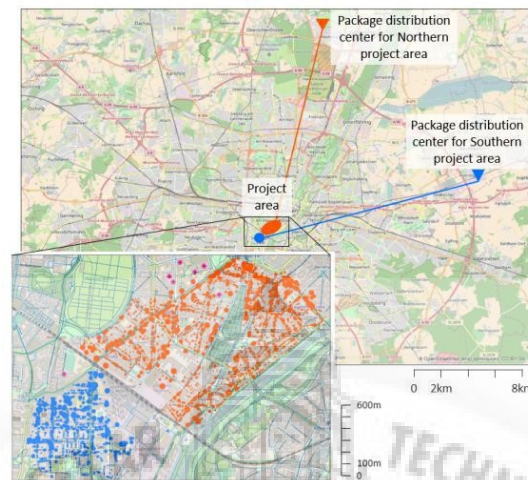


Figure 2.1: image

### 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 were 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

- All of the members are equally important in developing the project.
- We work on a different part of the project based on one's capability.
- Firstly we came up with documentation, And based on the documentation we set our goal and created a blueprint.
- 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

- User using this app need their work done easily.
- User of this app expects 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.

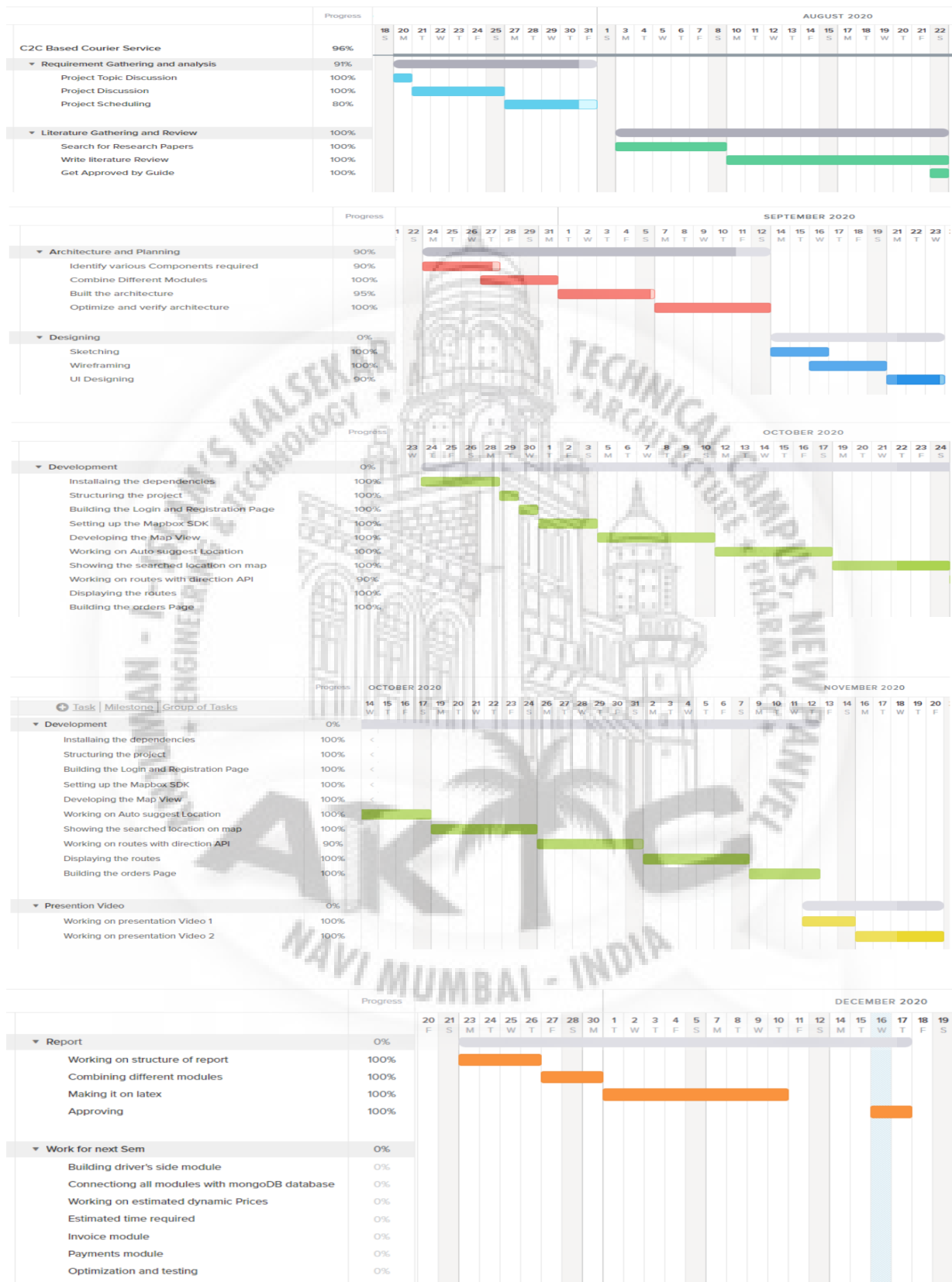
### **3.5 Ground Rules for the Project**

- 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



## 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 expect the better and innovative service which saves their time and work efficiently, and the people who expect cheaper value and Best Service.

#### 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 a 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.

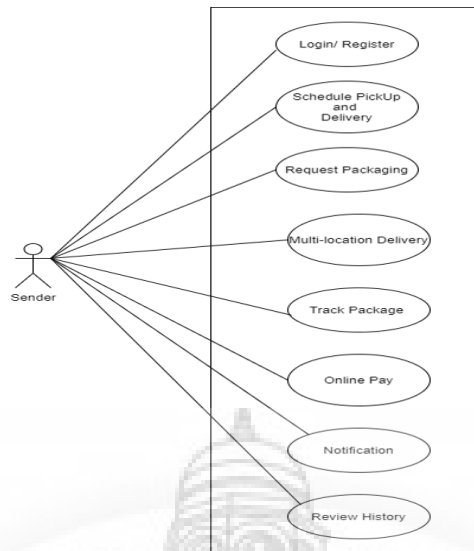


Figure 5.1: Sender

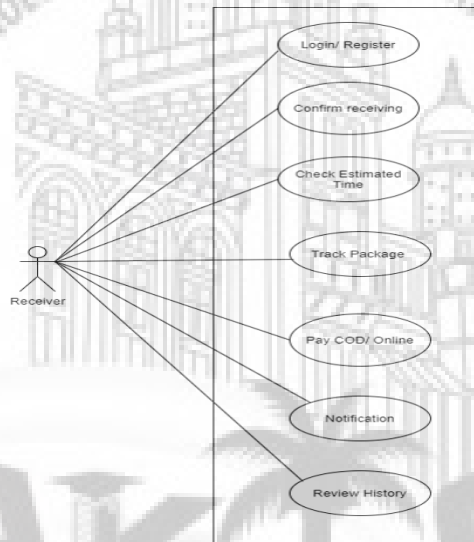


Figure 5.2: Receiver

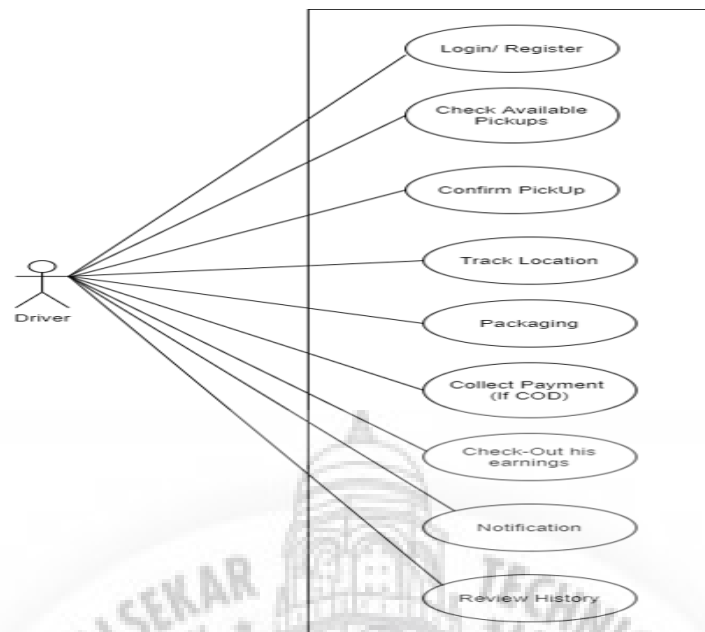


Figure 5.3: Driver

### 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.

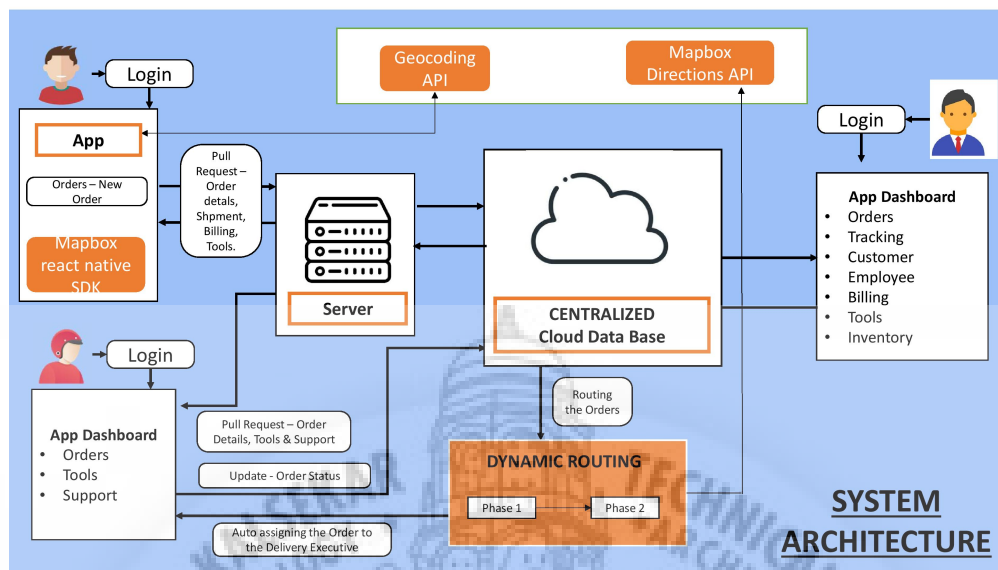


Figure 5.4: Driver

## 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.



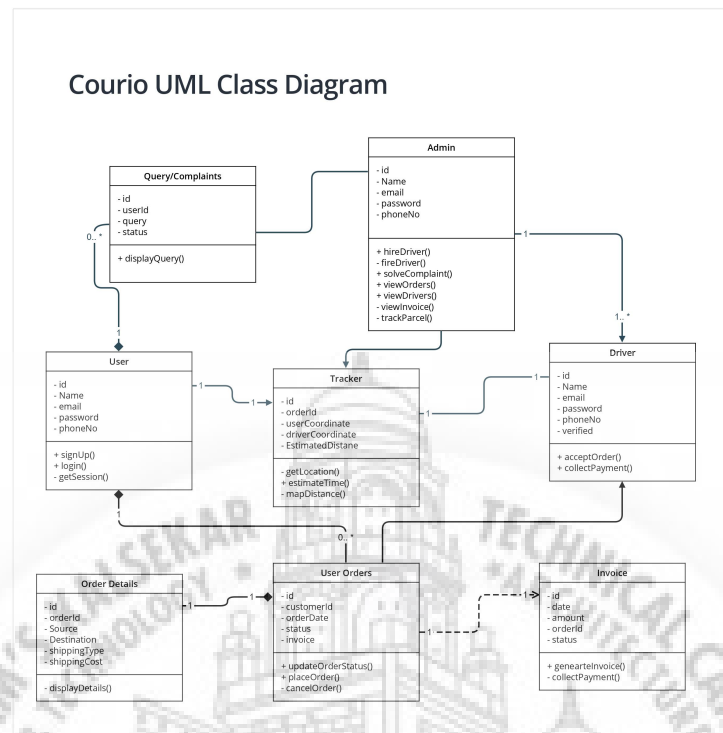


Figure 5.18: UML

### 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.

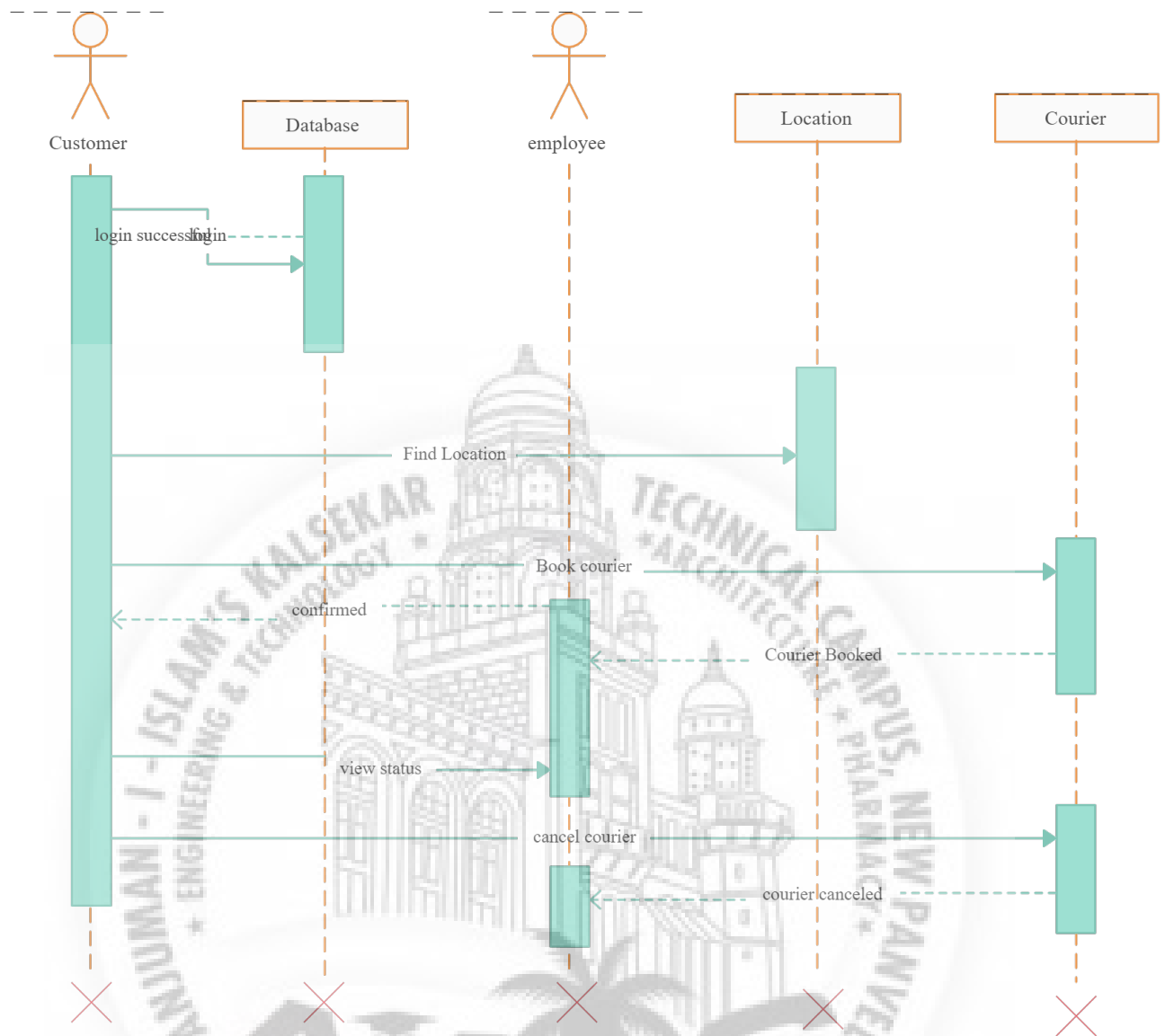


Figure 5.19: Sequence Diagram

# Chapter 6

## Implementation

### 6.1 Dashboard

```
1 import React, {Component} from 'react';
2 import axios from 'axios';
3
4 import {
5   View,
6   Text,
7   TouchableHighlight,
8   TextInput,
9   Button,
10  FlatList,
11  ScrollView,
12  Keyboard,
13  StatusBar,
14  Dimensions,
15  KeyboardAvoidingView,
16  StyleSheet,
17  TouchableOpacity,
18  PermissionsAndroid,
19 } from 'react-native';
20
21 import MapboxGL from '@react-native-mapbox-gl/maps';
22 import BottomSheet from 'reanimated-bottom-sheet';
23 import {SearchBar} from 'react-native-elements';
24 import Icon from 'react-native-vector-icons/Ionicons';
25 import SearchBarCustom from '../components/searchBar';
26 import Geolocation from 'react-native-geolocation-service';
27 //import Geolocation from '@react-native-community/geolocation';
28 // Geolocation.setRNConfiguration(config);
29 const height = Dimensions.get('window').height;
30
31 MapboxGL.setAccessToken(
32   'pk.eyJ1IjoiZXplY2EiLCJhIjoiY2tndW44MXk1MG1jODJ0cGVlZ2syamNhMyJ9.
33   DxN05YR2CCkIGBq8cU0ZJg',
34 );
35
36 class App extends Component {
37   constructor(props) {
38     super(props);
39
40     this.state = {
41       toggleInput: false,
42       focusedSourceInput: false,
43       focusedInput: false,
```

```

43     behavior: 'position',
44     search: '',
45     noStops: 2,
46     upperLayoutHeight: 0,
47     mapRegion: {
48         latitude: 72.8331,
49         longitude: 18.941,
50         latitudeDelta: 0.00922 * 1.5,
51         longitudeDelta: 0.00421 * 1.5,
52     },
53     lastLat: null,
54     lastLong: null,
55     location: 4.57901,
56     fromInput: '',
57     sourceFlag: 0,
58     sourceName: 'Your Location',
59     destName: '',
60     dest2Name: '',
61     dest3Name: '',
62     source: {lat: 72.8331, lon: 18.941},
63     destination: {lat: 72.831, lon: 18.9293},
64     destination2: {lat: null, lon: null},
65     destination3: {lat: null, lon: null},
66     lat: 72.8331,
67     lon: 18.941,
68     placeResult: '',
69     route: {
70         type: 'FeatureCollection',
71         features: [
72             {
73                 type: 'Feature',
74                 properties: {},
75                 geometry: {
76                     type: 'LineString',
77                     coordinates: [],
78                 },
79                 style: {
80                     fill: 'blue',
81                     strokeWidth: '6',
82                     fillOpacity: 0.6,
83                 },
84                 paint: {
85                     'fill-color': '#088',
86                     'fill-opacity': 0.8,
87                 },
88             },
89         ],
90     },
91 };
92 }
93
94 // {
95 //     coordinates: [
96 //         [72.8331, 18.941],
97 //         [72.831, 18.9293],
98 //     ],
99 // },
100
101 getCoord(keyVal) {
102     console.log(keyVal);
103     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
154 // 13.388860,52.517037;13.397634,52.529407;13.428555,52.523219;13.418555,52.523215?
155 // source=first&destination=last&roundtrip=true&access_token=
156 // YOUR_MAPBOX_ACCESS_TOKEN
157 direction() {
158     if (this.state.sourceName == 'Your Location') {
159         console.log(this.state.mapRegion.latitude);
160         console.log(this.state.mapRegion.longitude);
161         // var routeUrl =
162         // 'https://api.mapbox.com/optimized-trips/v1/mapbox/driving/' +
163         // this.state.mapRegion.longitude +
164         // ',' +

```

```

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.
    eyJ1IjoiZXplY2EiLCJhIjoiY2ndW44MXk1MG1jODJ0cGVlZ2syamNhMyJ9.
    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.
    eyJ1IjoiZXplY2EiLCJhIjoiY2ndW44MXk1MG1jODJ0cGVlZ2syamNhMyJ9.
    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.
    eyJ1IjoiZXplY2EiLCJhIjoiY2tndW44MXk1MG1jODJ0cGVlZ2syamNhMyJ9.
    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.
    eyJ1IjoiZXplY2EiLCJhIjoiY2tndW44MXk1MG1jODJ0cGVlZ2syamNhMyJ9.
    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.
    eyJ1IjoiZXplY2EiLCJhIjoiY2tndW44MXk1MG1jODJ0cGVlZ2syamNhMyJ9.
    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;

```

```

270     return coordin;
271   })
272   .then((coordin) => {
273     var geometry = this.state.route.features[0].geometry;
274     console.log('routes:', coordin);
275     console.log('cord:', geometry);
276     geometry.coordinates = coordin;
277
278     this.setState({geometry});
279     console.log(cord);
280     console.log(
281       'state route',
282       this.state.route.features[0].geometry.coordinates,
283     );
284   })
285   .catch((error) => {
286     console.log(error);
287   });
288 }
289 _handlePress() {
290   console.log(this.state.location);
291   const searchUrl =
292     'https://api.mapbox.com/geocoding/v5/mapbox.places/' +
293     this.state.location +
294     '?bbox=72.64472835639785,18.777422428792846,73.31946817024394,19.581105982221317&
295     access_token=pk.eyJHJoiZXplY2EiLCJhIjoieY2ndW44MXk1MG1jODJ0cGVlZ2syamNhMyJ9.
296     DxN05YR2CCkIGBq8cU0ZJg';
297   console.log(searchUrl);
298   axios
299     .get(searchUrl)
300     .then((response) => {
301       // console.log(response.request._response);
302       resData = response.request._response;
303       // var finalData = resData.replace(/\\/g, '');
304       var obj = JSON.parse(resData);
305       console.log(obj.features[0].center);
306       var featuresLen = obj.features.length;
307       console.log('len' + featuresLen);
308       var i;
309       var places = [];
310
311       for (i = 0; i < 5; i++) {
312         console.log(obj.features[i].place_name);
313         placeName = obj.features[i].place_name;
314         coordinate = obj.features[i].center;
315
316         places.push({
317           key: i,
318           name: placeName,
319           coordinates: coordinate,
320         });
321       }
322       console.log(places);
323       return places;
324
325       // console.log(finalData.features);
326     })
327     .then((places) => {
328       console.log(places);

```



```

327     this.setState({ placeResult: places });
328     console.log('-----');
329     console.log(this.state.placeResult);
330   })
331   .catch((error) => {
332     console.log(error);
333   });
334 }
335
336 // openPopUp() {
337 //   this.bottomSheet.open();
338 // }
339 find_dimensions(layout) {
340   const {x, y, width, height} = layout;
341   console.warn(x);
342   console.warn(y);
343   console.warn(width);
344   console.warn(height);
345   this.setState({ upperLayoutHeight: height });
346 }
347 updateSearch = (search) => {
348   this.setState({ search });
349 };
350 getLocation = async () => {
351   try {
352     const granted = await PermissionsAndroid.request(
353       PermissionsAndroid.PERMISSIONS.ACCESS_FINE_LOCATION,
354     );
355     if (granted === PermissionsAndroid.RESULTS.GRANTED) {
356       Geolocation.getCurrentPosition(
357         (position) => {
358           console.log('pos', position);
359           let region = {
360             latitude: position.coords.latitude,
361             longitude: position.coords.longitude,
362             latitudeDelta: 0.00922 * 1.5,
363             longitudeDelta: 0.00421 * 1.5,
364           };
365           this.setState({ mapRegion: region });
366           // var source = this.state.source;
367           // source.lat = position.coords.latitude;
368           // source.lon = position.coords.longitude;
369           // this.setState({ source });
370         },
371         (error) => {
372           // See error code charts below.
373           console.log(error.code, error.message);
374         },
375         {enableHighAccuracy: true, timeout: 15000, maximumAge: 10000},
376       );
377     } else {
378       alert('Location permission denied');
379     }
380   } catch (err) {
381     console.warn(err);
382   }
383 };
384 onRegionChange = async (region, lastLat, lastLong) => {
385   await this.setState({
386     mapRegion: region,
387     // If there are no new values set the current ones

```

```

388     lastLat: lastLat || this.state.lastLat ,
389     lastLong: lastLong || this.state.lastLong ,
390   });
391   console.log(region.latitude , region.longitude);
392 };
393
394 componentDidMount = async () => {
395   {
396     this.getLocation();
397   }
398   {
399     this.watchID = Geolocation.watchPosition(
400       (position) => {
401         // Create the object to update this.state.mapRegion through the
402         // onRegionChange function
403         let region = {
404           latitude: position.coords.latitude ,
405           longitude: position.coords.longitude ,
406           latitudeDelta: 0.00922 * 1.5 ,
407           longitudeDelta: 0.00421 * 1.5 ,
408         };
409         this.onRegionChange(region, region.latitude , region.longitude);
410       },
411       (error) => console.log(error) ,
412     );
413   }
414 };
415 componentWillUnmount () {
416   Geolocation.clearWatch(this.watchID);
417 }
418 renderContent = () => (
419   <View
420     style={{
421       backgroundColor: 'white' ,
422       padding: 16 ,
423       height: 300 ,
424     }}>
425     <View>
426       <View style={styles.panelHandle} />
427     </View>
428
429     <View style={styles.searchContainer}>
430       <TextInput
431         placeholder="Pickup"
432         name="source"
433         style={styles.searchBox}
434         value={this.state.sourceName}
435         onChangeText={(text) => {
436           this.setState({ location: text });
437           this.setState({ fromInput: 'source' });
438           this.setState({ sourceName: text });
439           this._handlePress();
440         }}></TextInput>
441       <TextInput
442         placeholder="Search Destination"
443         name="dest"
444         style={styles.searchBox}
445         value={this.state.destName}
446         onChangeText={(text) => {
447           this.setState({ location: text });

```

```

448     this.setState({ fromInput: 'dest' });
449     this.setState({ destName: text });
450     this._handlePress();
451   }}></TextInput>
452 </View>
453 <View>
454   <FlatList
455     data={this.state.placeResult}
456     renderItem={({item}) => (
457       <TouchableHighlight onPress={() => this.getCoord(item.key)}>
458         <Text key={item.key.toString()} bottomDivider>
459           {item.name}
460         </Text>
461       </TouchableHighlight>
462     )}
463   />
464 </View>
465 </View>
466 );
467
468 render() {
469   return (
470     <View style={{flex: 1, height: '100%', width: '100%'}}>
471       <View>
472         <View
473           style={styles.searchContainer}
474           onLayout={(event) => {
475             this.find_dimesions(event.nativeEvent.layout);
476           }}>
477           {this.state.toggleInput && (
478             <View style={{marginBottom: 8, marginTop: 8}}>
479             /* <TextInput
480              // placeholder="Pickup"
481              name="source"
482              style={styles.searchBox}
483              value={this.state.sourceName}
484              onFocus={() => {
485                if (this.state.sourceName == 'Your Location') {
486                  this.setState({sourceName: ''});
487                }
488              }}
489              onChangeText={(text) => {
490                this.setState({location: text});
491                this.setState({fromInput: 'source'});
492                this.setState({sourceName: text});
493                this._handlePress();
494              }}></TextInput> */}
495             <SearchBarCustom
496               textValue={this.state.sourceName}
497               onFocusBox={() => {
498                 if (this.state.sourceName == 'Your Location') {
499                   this.setState({sourceName: ''});
500                   this.setState({focusedInput: true});
501                   this.setState({focusedSourceInput: true});
502                 }
503               }}
504               valueChange={async (text) => {
505                 this.setState({location: text});
506                 this.setState({fromInput: 'source'});
507                 this.setState({sourceName: text});
508                 this._handlePress();

```

```

509         }}></SearchBarCustom>
510     </View>
511 )}
512 { /* <TextInput
513     placeholder="Search Destination"
514     name="dest"
515     style={styles.searchBox}
516     value={this.state.destName}
517     onFocus={() => this.setState({toggleInput: true})}
518     onChangeText={(text) => {
519         this.setState({location: text});
520         this.setState({fromInput: 'dest'});
521         this.setState({destName: text});
522         this._handlePress();
523     }}></TextInput> */}
524 <View style={{marginBottom: 8, marginTop: 8}}>
525     <SearchBarCustom
526         textValue={this.state.destName}
527         onFocusBox={() => {
528             this.setState({toggleInput: true});
529             this.setState({focusedInput: true});
530         }}
531         valueChange={async (text) => {
532             this.setState({location: text});
533             this.setState({fromInput: 'dest'});
534             this.setState({destName: text});
535             this._handlePress();
536         }}></SearchBarCustom>
537 </View>
538 {this.state.toggleInput && this.state.noStops >= 3 && (
539     <View style={{flexDirection: 'row'}}>
540     <View style={{flex: 1, marginBottom: 6}}>
541         <SearchBarCustom
542             onFocusBox={() => {}}
543             textValue={this.state.dest2Name}
544             valueChange={async (text) => {
545                 this.setState({location: text});
546                 this.setState({fromInput: 'stop2'});
547                 await this.setState({dest2Name: text});
548                 this._handlePress();
549                 console.log(text);
550                 console.log(this.state.dest2Name);
551             }}></SearchBarCustom>
552     </View>
553     <TouchableOpacity>
554         <Icon
555             name="close-outline"
556             style={{
557                 marginRight: 15,
558                 marginTop: 8,
559                 marginLeft: 5,
560                 flex: 1,
561             }}
562             onPress={() => {
563                 this.setState({noStops: this.state.noStops - 1});
564             }}
565             size={25}></Icon>
566     </TouchableOpacity>
567 </View>
568 )}
569 {this.state.toggleInput && this.state.noStops >= 4 && (

```

```

570 <View style={{flexDirection: 'row'}}>
571 <View style={{flex: 1, marginBottom: 6}}>
572 <SearchBarCustom
573   onFocusBox={() => {}}
574   textValue={this.state.dest3Name}
575   valueChange={async (text) => {
576     this.setState({location: text});
577     this.setState({fromInput: 'stop3'});
578     await this.setState({dest3Name: text});
579     this.handlePress();
580     console.log(text);
581     console.log(this.state.dest3Name);
582     console.log(this.state.dest2Name);
583   }}></SearchBarCustom>
584 </View>
585 <TouchableOpacity>
586 <Icon
587   name="close-outline"
588   style={{
589     marginRight: 15,
590     marginTop: 8,
591     marginLeft: 5,
592     flex: 1,
593   }}
594   onPress={() => {
595     this.setState({noStops: this.state.noStops - 1});
596   }}
597   size={25}></Icon>
598 </TouchableOpacity>
599 </View>
600 )}
601 {this.state.toggleInput && this.state.noStops <= 3 && (
602 <Button
603   title="Add Stop"
604   onPress={() => {
605     this.setState({noStops: this.state.noStops + 1});
606   }}></Button>
607 )}
608 {this.state.toggleInput && (
609 <TouchableOpacity
610   style={{margin: 10}}
611   onPress={() => {
612     this.setState({toggleInput: false});
613     Keyboard.dismiss();
614     this.setState({focusedInput: false});
615   }}>
616 <Text style={{textAlign: 'right'}}>Done</Text>
617 </TouchableOpacity>
618 )}
619 </View>
620 </View>
621
622 {this.state.focusedInput && (
623 <TouchableOpacity
624   onPress={() => {
625     this.setState({focusedInput: !this.state.focusedInput});
626   }}>
627 <View style={{height: 700, backgroundColor: 'transparent'}}>
628 <ScrollView>
629   {this.state.focusedSourceInput && (
630 <TouchableOpacity

```

```

631         onPress={() => {
632             this.setState({ location: 'Your Location' });
633             this.setState({ fromInput: 'source' });
634             this.setState({ sourceName: 'Your Location' });
635             this.getLocation();
636             this.setState({ focusedSourceInput: false });
637         }}>
638         <Text>Your Location </Text>
639     </TouchableOpacity>
640 )}
641
642 <FlatList
643     data={this.state.placeResult}
644     renderItem={({ item }) => (
645         <TouchableHighlight onPress={() => this.getCoord(item.key)}>
646             <Text key={item.key.toString()} bottomDivider>
647                 {item.name}
648             </Text>
649         </TouchableHighlight>
650     )}
651 />
652 </ScrollView>
653 </View>
654 </TouchableOpacity>
655 )}
656
657 <MapboxGL.MapView
658     styleURL={MapboxGL.StyleURL.Street}
659     zoomLevel={16}
660     centerCoordinate={[this.state.lat, this.state.lon]}
661     style={{
662         flex: 1,
663     }}
664     onPress={(feature) =>
665         console.log('Coords:', feature.geometry.coordinates)
666     }>
667     <MapboxGL.Camera
668         zoomLevel={16}
669         centerCoordinate={[this.state.lat, this.state.lon]}
670         // centerCoordinate={[
671         //     this.state.mapRegion.latitude,
672         //     this.state.mapRegion.longitude,
673         // ]}
674         animationMode={'flyTo'}
675         animationDuration={20}></MapboxGL.Camera>
676     {this.state.sourceName === 'Your Location' && (
677         <MapboxGL.PointAnnotation
678             key="key1"
679             id="id1"
680             title="sourcePointer"
681             coordinate={[
682                 this.state.mapRegion.latitude,
683                 this.state.mapRegion.longitude,
684             ]}></MapboxGL.PointAnnotation>
685     )}
686     {this.state.sourceName !== 'Your Location' && (
687         <MapboxGL.PointAnnotation
688             key="key1"
689             id="id1"
690             title="sourcePointer"
691             coordinate={

```

```

692         this.state.source.lat ,
693         this.state.source.lon ,
694     ]}></MapboxGL.PointAnnotation>
695     )}
696 <MapboxGL.PointAnnotation
697     key="key2"
698     id="id2"
699     title="destPointer"
700     coordinate=[
701         this.state.destination.lat ,
702         this.state.destination.lon ,
703     ]}></MapboxGL.PointAnnotation>
704 {this.state.destination2.lat != null && (
705     <MapboxGL.PointAnnotation
706         key="key3"
707         id="id3"
708         title="dest2Pointer"
709         coordinate=[
710             this.state.destination2.lat ,
711             this.state.destination2.lon ,
712         ]}></MapboxGL.PointAnnotation>
713     )}
714 {this.state.destination3.lat != null && (
715     <MapboxGL.PointAnnotation
716         key="key4"
717         id="id4"
718         title="dest3Pointer"
719         coordinate=[
720             this.state.destination3.lat ,
721             this.state.destination3.lon ,
722         ]}></MapboxGL.PointAnnotation>
723     )}
724 <MapboxGL.ShapeSource id="progressSource" shape={this.state.route}>
725     <MapboxGL.LineLayer
726         id="progressFill"
727         style={{
728             lineColor: '#ffcc5c',
729             lineWidth: 6,
730             lineCap: 'round',
731         }}
732     />
733 </MapboxGL.ShapeSource>
734 <MapboxGL.UserLocation visible={true} />
735 </MapboxGL.MapView>
736
737 <View
738     style={{
739         width: 80,
740         backgroundColor: 'transparent',
741         position: 'absolute',
742
743         top: '85%',
744         left: '70%',
745         zIndex: 10,
746     }}>
747 {this.state.destination.lat != '' && (
748     <Button
749         onPress={() => {
750             console.log('$$$$$$$$$$$$$$$$$ ..... ', this.state.noStops);
751             // console.log('$$$$$ ..... ', this.state.noStops);
752

```

```

753     if (this.state.noStops == 2) {
754         console.log('sourCord', this.state.source);
755         this.props.navigation.navigate('Details', {
756             mapRegion: this.state.mapRegion,
757             sourceName: this.state.sourceName,
758             destName: this.state.destName,
759             sourceCord: this.state.source,
760             destCord: this.state.destination,
761             noStops: this.state.noStops,
762         });
763     } else if (this.state.noStops == 3) {
764         this.props.navigation.navigate('Details', {
765             mapRegion: this.state.mapRegion,
766             sourceName: this.state.sourceName,
767             destName: this.state.destName,
768             dest2Name: this.state.dest2Name,
769             sourCord: this.state.source,
770             destCord: this.state.destination,
771             dest2Cord: this.state.destination2,
772             noStops: this.state.noStops,
773         });
774     } else if (this.state.noStops == 4) {
775         this.props.navigation.navigate('Details', {
776             mapRegion: this.state.mapRegion,
777             sourceName: this.state.sourceName,
778             destName: this.state.destName,
779             dest2Name: this.state.dest2Name,
780             dest3Name: this.state.dest3Name,
781             sourCord: this.state.source,
782             destCord: this.state.destination,
783             dest2Cord: this.state.destination2,
784             dest3Cord: this.state.destination3,
785             noStops: this.state.noStops,
786         });
787     }
788 }
789 title="GO"></Button>
790 )}
791
792 {/* <Button
793     onPress={() => this.props.navigation.navigate('Swipe')}
794     title="Swipe"></Button> */}
795 </View>
796
797 {/* <BottomSheet
798     // ref={sheetRef}
799     // height - this.state.upperLayoutHeight - 80
800     snapPoints={[200, 30]}
801     initialSnap={0}
802     enabledGestureInteraction={true}
803     enabledBottomClamp={true}
804     borderRadius={25}
805     renderContent={this.renderContent}
806     /> */}
807 </View>
808 );
809 }
810 }
811
812 const styles = StyleSheet.create({
813     container: {

```



```
814     flex: 1,
815     padding: 24,
816     backgroundColor: '#eaeaea',
817   },
818   buttonStyle: {
819     width: 80,
820     borderColor: 'gray',
821     borderWidth: 1,
822     right: 0,
823     bottom: 0,
824     position: 'absolute',
825     backgroundColor: 'transparent',
826   },
827   crossButtonStyle: {
828     width: 80,
829     borderColor: 'gray',
830     borderWidth: 1,
831     margin: 20,
832     height: 10,
833     flex: 1,
834     backgroundColor: 'transparent',
835   },
836   searchBox: {
837     height: 40,
838     borderColor: 'gray',
839     borderWidth: 2,
840     margin: 8,
841     backgroundColor: 'white',
842   },
843   searchBoxStop: {
844     height: 40,
845     borderColor: 'gray',
846     borderWidth: 2,
847     margin: 8,
848     flex: 1,
849     backgroundColor: 'white',
850   },
851   searchContainer: {
852     backgroundColor: 'white',
853     // backgroundColor: '#EBF5FB',
854   },
855   panelHandle: {
856     width: 40,
857     height: 8,
858     borderRadius: 4,
859     backgroundColor: '#000004',
860     marginBottom: 10,
861     marginLeft: 170,
862   },
863 });
864
865 export default App;
```

## 6.2 Order Detail

```

1  import React, {Component} from 'react';
2  import {
3    Text,
4    View,
5    TextInput,
6    StyleSheet,
7    Button,
8    ScrollView,
9  } from 'react-native';
10 import {Card} from 'react-native-elements';
11
12 import AsyncStorage from '@react-native-async-storage/async-storage';
13
14 class orderDetails extends Component {
15   state = {
16     sourceMobile: null,
17     destMobile: null,
18     dest2Mobile: null,
19     dest3Mobile: null,
20
21     sourceAddressNote: '',
22     destAddressNote: '',
23     dest2AddressNote: '',
24     dest3AddressNote: ''
25   };
26
27   // saveOrder = async () => {
28   //   const value = await AsyncStorage.getItem('token');
29   //   console.log('$$$$$$$$$$$$$$$$$.....', value);
30
31   //   const body = JSON.stringify({
32   //     packageName: this.state.packageName,
33   //     source: this.state.source,
34   //     destination: this.state.destination,
35   //     dimLength: this.state.dimLength,
36   //     dimWidth: this.state.dimWidth,
37   //     dimHeight: this.state.dimHeight,
38   //     weight: this.state.weight,
39   //   });
40
41   //   const headers = {
42   //     Accept: 'application/json',
43   //     'Content-Type': 'application/json',
44   //     Authorization: 'Bearer ' + value,
45   //   };
46
47   //   const settings = {
48   //     method: 'POST',
49   //     headers,
50   //     body,
51   //   };
52
53   //   try {
54   //     const fetchResponse = await fetch(
55   //       'http://192.168.0.108:3000/postOrder',
56   //       settings,
57   //     );
58   //     const data = await fetchResponse.json();
59   //     console.log('+++++', data);

```

```

60 //     if (data.token) {
61 //         this.onOrderSuccess(data);
62 //     }
63
64 //     return data; // your response data
65 // } catch (e) {
66 //     console.log(e);
67 //     return e; // handle your error here
68 // }
69 // };
70 // onOrderSuccess = async (data) => {
71 //     console.log('Order Success', data);
72 //     this.props.navigation.navigate('YourOrders', {
73 //         packageName: this.state.packageName,
74 //     });
75
76 //     const value = await AsyncStorage.getItem('token');
77 //     console.log('$$$$$$$$$$$$$$$$$---', value);
78 // };
79 navigateToSchedule (
80     sourceName ,
81     destName ,
82     dest2Name ,
83     dest3Name ,
84     sourceCordin ,
85     destCord ,
86     dest2Cord ,
87     dest3Cord ,
88     noStops ,
89 ) {
90     console.log('$$$$$$$$$$$$$$$$$ ..... ', noStops);
91
92     this.props.navigation.navigate('ScheduleOrder', {
93         sourceMobile: this.state.sourceMobile ,
94         destMobile: this.state.destMobile ,
95         dest2Mobile: this.state.dest2Mobile ,
96         dest3Mobile: this.state.dest3Mobile ,
97         sourceAddressNote: this.state.sourceAddressNote ,
98         destAddressNote: this.state.destAddressNote ,
99         dest2AddressNote: this.state.dest2AddressNote ,
100        dest3AddressNote: this.state.dest3AddressNote ,
101        sourceName: sourceName ,
102        destName: destName ,
103        dest2Name: dest2Name ,
104        dest3Name: dest3Name ,
105        sourceCord: sourceCordin ,
106        destCord ,
107        dest2Cord ,
108        dest3Cord ,
109        noStops ,
110    });
111 }
112
113 render () {
114     const sourceName = this.props.navigation.getParam(
115         'sourceName' ,
116         'Your Loaction' ,
117     );
118     const destName = this.props.navigation.getParam('destName' , 'Destination');
119     const dest2Name = this.props.navigation.getParam(
120         'dest2Name' ,

```

```

121     'Destination2',
122   );
123   const dest3Name = this.props.navigation.getParam(
124     'dest3Name',
125     'Destination3',
126   );
127
128   const mapRegion = this.props.navigation.getParam('mapRegion', null);
129   const sourceCord = this.props.navigation.getParam('sourceCord', null);
130   const destCord = this.props.navigation.getParam('destCord', null);
131   const dest2Cord = this.props.navigation.getParam('dest2Cord', null);
132   const dest3Cord = this.props.navigation.getParam('dest3Cord', null);
133
134   const noStops = this.props.navigation.getParam('noStops', 2);
135
136   return (
137     <View style={styles.container}>
138       <ScrollView>
139         <Text style={styles.headerText}>Address Details </Text>
140         <Card style={{borderRadius: 8}}>
141           <Text>{sourceName}</Text>
142           <TextInput
143             keyboardType="numeric"
144             maxLength={10}
145             style={styles.textBox}
146             onChangeText={(num) => {
147               this.setState({sourceMobile: num});
148             }}
149             placeholder="Mobile No"></TextInput>
150           <TextInput
151             style={styles.multiLineBox}
152             multiline={true}
153             onChangeText={(text) => {
154               this.setState({sourceAddressNote: text});
155             }}
156             placeholder="Source Address Note (Ex.Flat No., Compound, Street
157               ...)"></TextInput>
158         </Card>
159         <Card>
160           <Text>{destName}</Text>
161           <TextInput
162             keyboardType="numeric"
163             maxLength={10}
164             style={styles.textBox}
165             onChangeText={(num) => {
166               this.setState({destMobile: num});
167             }}
168             placeholder="Receivers Mobile No"></TextInput>
169           <TextInput
170             style={styles.multiLineBox}
171             multiline={true}
172             onChangeText={(text) => {
173               this.setState({destAddressNote: text});
174             }}
175             placeholder="Destination Address Note (Ex.Flat No., Compound,
176               Street ...)"></TextInput>
177         </Card>
178         {dest2Name !== 'Destination2' && (
179           <Card>
180             <Text>{dest2Name}</Text>
181             <TextInput

```

```

180         keyboardType="numeric"
181         maxLength={10}
182         style={styles.textBox}
183         onChangeText={(num) => {
184             this.setState({dest2Mobile: num});
185         }}
186         placeholder="Receivers Mobile No"></TextInput>
187     <TextInput
188         style={styles.multiLineBox}
189         multiline={true}
190         onChangeText={(text) => {
191             this.setState({dest2AddressNote: text});
192         }}
193         placeholder="Destination Address Note (Ex.Flat No., Compound,
194             Street ...)"></TextInput>
195 </Card>
196 )}
197 {dest3Name != 'Destination3' && (
198     <Card>
199         <Text>{dest3Name}</Text>
200         <TextInput
201             keyboardType="numeric"
202             maxLength={10}
203             style={styles.textBox}
204             onChangeText={(num) => {
205                 this.setState({dest3Mobile: num});
206             }}
207             placeholder="Receivers Mobile No"></TextInput>
208         <TextInput
209             style={styles.multiLineBox}
210             multiline={true}
211             onChangeText={(text) => {
212                 this.setState({dest3AddressNote: text});
213             }}
214             placeholder="Destination Address Note (Ex.Flat No., Compound,
215                 Street ...)"></TextInput>
216         </Card>
217     )}
218     { /* <Text style={styles.headerText}>Package Dimensions </Text> */}
219     { /* <TextInput
220         style={styles.textBox}
221         onChangeText={(text) => {
222             this.setState({dimLength: text});
223         }}
224         placeholder="Length"></TextInput>
225     <TextInput
226         style={styles.textBox}
227         onChangeText={(text) => {
228             this.setState({dimWidth: text});
229         }}
230         placeholder="Width"></TextInput>
231     <TextInput
232         style={styles.textBox}
233         onChangeText={(text) => {
234             this.setState({dimHeight: text});
235         }}
236         placeholder="Height"></TextInput>
237     <TextInput
238         style={styles.textBox}
239         onChangeText={(text) => {
240             this.setState({weight: text});

```

```

239     }}
240     placeholder="Weight Approx"></TextInput> */}
241   {/* <TextInput
242     style={styles.textBox}
243     onChangeText={(text) => {
244       this.setState({packageName: text});
245     }}
246     placeholder="Give a name to your Package"></TextInput> */}
247 </ScrollView>
248 <View
249   style={{
250     width: 90,
251     backgroundColor: 'transparent',
252     position: 'absolute',
253     top: '85%',
254     left: '70%',
255     zIndex: 10,
256   }}>
257 <Button
258   onPress={() => {
259     console.log('nostops: ', noStops);
260     if (sourceName == 'Your Location') {
261       var sourceCordin = {
262         lat: mapRegion.longitude,
263         lon: mapRegion.latitude,
264       };
265
266       this.navigateToSchedule(
267         sourceName,
268         destName,
269         dest2Name,
270         dest3Name,
271
272         sourceCordin,
273         destCord,
274         dest2Cord,
275         dest3Cord,
276         noStops,
277       );
278     } else {
279       var sourceCordin = sourceCord;
280       console.log('sourceCord:', sourceCord);
281       console.log('sourceCord:', sourceCordin);
282       this.navigateToSchedule(
283         sourceName,
284         destName,
285         dest2Name,
286         dest3Name,
287
288         sourceCordin,
289         destCord,
290         dest2Cord,
291         dest3Cord,
292         noStops,
293       );
294     }
295   }}
296   title="Next"></Button>
297 </View>
298 </View>
299 );

```

```
300   }
301 }
302
303 const styles = StyleSheet.create({
304   container: {
305     backgroundColor: '#EBF5FB',
306     //backgroundColor: '#E9967A',
307     height: '100%',
308   },
309   textBox: {
310     height: 40,
311     borderColor: 'gray',
312     borderWidth: 1,
313
314     margin: 20,
315   },
316   multiLineBox: {
317     // height: 40,
318     borderColor: 'gray',
319     borderWidth: 1,
320
321     marginLeft: 20,
322     marginRight: 20,
323   },
324   headerText: {
325     textAlign: 'center',
326     margin: 10,
327   },
328   buttonStyle: {
329     width: 40,
330     borderColor: 'gray',
331     borderWidth: 1,
332
333     margin: 20,
334   },
335 });
336
337 export default orderDetails;
```

## 6.3 Schedule Order

```

1  import React, {Component} from 'react';
2  import {Text, View, TextInput, StyleSheet, ScrollView} from 'react-native';
3  import {Card, Slider, Button, ButtonGroup} from 'react-native-elements';
4  import {Animated} from 'react-native';
5  import Icon from 'react-native-vector-icons/Ionicons';
6  import DatePicker from 'react-native-date-picker';
7
8  import AsyncStorage from '@react-native-async-storage/async-storage';
9  import {Directions} from 'react-native-gesture-handler';
10
11 class scheduleScreen extends Component {
12   constructor() {
13     super();
14     this.state = {
15       weightValue: 1,
16       sourceMobile: null,
17       destMobile: null,
18       dest2Mobile: null,
19       dest3Mobile: null,
20
21       sourceName: '',
22       destName: '',
23       dest2Name: '',
24       dest3Name: '',
25
26       sourceAddressNote: '',
27       destAddressNote: '',
28       dest2AddressNote: '',
29       dest3AddressNote: '',
30
31       sourceCord: {lat: null, lon: null},
32       destCord: {lat: null, lon: null},
33       dest2Cord: {lat: null, lon: null},
34       dest3Cord: {lat: null, lon: null},
35
36       selectedIndex: 2,
37       scheduleDateTime: new Date(),
38       currentTime: null,
39       packageName: '',
40       amount: 50,
41     };
42   }
43
44   saveOrder = async (
45     sourceName,
46     destName,
47     dest2Name,
48     dest3Name,
49     noStops,
50     sourceCord,
51     destCord,
52     dest2Cord,
53     dest3Cord,
54     sourceAddressNote,
55     destAddressNote,
56     dest2AddressNote,
57     dest3AddressNote,
58     sourceMobile,
59     destMobile,

```



```
60     dest2Mobile ,
61     dest3Mobile ,
62 ) => {
63     const value = await AsyncStorage.getItem('token');
64     console.log('$$$$$$$$$$$$$$$$$ ..... ', value);
65     console.log('$$$$$$$$$$$$$$$$$ ..... ', noStops);
66
67     var details = {
68         sourceName ,
69         destName ,
70         dest2Name ,
71         dest3Name ,
72         noStops ,
73         sourceCord ,
74         destCord ,
75         dest2Cord ,
76         dest3Cord ,
77         sourceAddressNote ,
78         destAddressNote ,
79         dest2AddressNote ,
80         dest3AddressNote ,
81         sourceMobile ,
82         destMobile ,
83         dest2Mobile ,
84         dest3Mobile ,
85     };
86
87     const body = JSON.stringify({
88         packageName: this.state.packageName ,
89         sourceName ,
90         destName ,
91         dest2Name ,
92         dest3Name ,
93
94         sourceCord ,
95         destCord ,
96         dest2Cord ,
97         dest3Cord ,
98
99         sourceAddressNote ,
100        destAddressNote ,
101        dest2AddressNote ,
102        dest3AddressNote ,
103
104        sourceMobile ,
105        destMobile ,
106        dest2Mobile ,
107        dest3Mobile ,
108
109        noStops ,
110
111        scheduleType: this.state.selectedIndex ,
112        scheduleDateTime: this.state.scheduleDateTime ,
113        amount: this.state.amount ,
114        weight: this.state.weightValue ,
115    });
116
117     const headers = {
118         Accept: 'application/json' ,
119         'Content-Type': 'application/json' ,
120         Authorization: 'Bearer ' + value ,
```

```
121     };
122
123     const settings = {
124       method: 'POST',
125       headers,
126       body,
127     };
128
129     try {
130       const fetchResponse = await fetch(
131         'http://192.168.0.107:3000/postOrder',
132         settings,
133       );
134       const data = await fetchResponse.json();
135       console.log('+++++', data);
136       if (data.packageName) {
137         this.onOrderSuccess(data);
138       }
139
140       return data; // your response data
141     } catch (e) {
142       console.log(e);
143       return e; // handle your error here
144     }
145   };
146   onOrderSuccess = async (data) => {
147     console.log('Order Success', data);
148     this.props.navigation.navigate('YourOrders', {
149       packageName: this.state.packageName,
150     });
151     // if (this.props.selectedIndex == 1){
152     //   this.props.navigation.navigate('YourOrders', {
153     //     packageName: this.state.packageName,
154     //   });
155     // }
156     // else if (this.props.selectedIndex == 0){
157     //   this.props.navigation.navigate('Tracking', {
158     //     data: details,
159     //   });
160     // }
161
162     const value = await AsyncStorage.getItem('token');
163     console.log('$$$$$$$$$$$$$$$$$---', value);
164   };
165
166   updateIndex = async (selectedIndex) => {
167     await this.setState({ selectedIndex });
168     console.log(selectedIndex);
169     console.log(this.state.selectedIndex);
170   };
171
172   navigateToSchedule (
173     sourceName,
174     destName,
175     dest2Name,
176     dest3Name,
177     sourceCord,
178     destCord,
179     dest2Cord,
180     dest3Cord,
181     noStops,
```

```

182 ) {
183   this.props.navigation.navigate('YourOrders', {
184     sourceMobile: this.state.sourceMobile,
185     destMobile: this.state.destMobile,
186     dest2Mobile: this.state.dest2Mobile,
187     dest3Mobile: this.state.dest3Mobile,
188     sourceAddressNote: this.state.sourceAddressNote,
189     destAddressNote: this.state.destAddressNote,
190     dest2AddressNote: this.state.dest2AddressNote,
191     dest3AddressNote: this.state.dest3AddressNote,
192     sourceName: sourceName,
193     destName: destName,
194     dest2Name: dest2Name,
195     dest3Name: dest3Name,
196     sourceCord,
197     destCord,
198     dest2Cord,
199     dest3Cord,
200     noStops: noStops,
201   });
202 }
203 componentDidMount = async () => {
204   // if (this.state.initialLoad) {
205   var date = new Date();
206   // var timePostHr = date.getHours();
207   await this.setState({currentTime: date.getHours()});
208   console.log('---', this.state.currentTime);
209   // }
210 };
211
212 render() {
213   const sourceName = this.props.navigation.getParam(
214     'sourceName',
215     'Your Location',
216   );
217   const destName = this.props.navigation.getParam('destName', 'Destination');
218   const dest2Name = this.props.navigation.getParam(
219     'dest2Name',
220     'Destination2',
221   );
222   const dest3Name = this.props.navigation.getParam(
223     'dest3Name',
224     'Destination3',
225   );
226   const noStops = this.props.navigation.getParam('noStops', 2);
227
228   const sourceCord = this.props.navigation.getParam('sourceCord', null);
229   const destCord = this.props.navigation.getParam('destCord', null);
230   const dest2Cord = this.props.navigation.getParam('dest2Cord', null);
231   const dest3Cord = this.props.navigation.getParam('dest3Cord', null);
232
233   const sourceAddressNote = this.props.navigation.getParam(
234     'sourceAddressNote',
235     null,
236   );
237   const destAddressNote = this.props.navigation.getParam(
238     'destAddressNote',
239     null,
240   );
241   const dest2AddressNote = this.props.navigation.getParam(
242     'dest2AddressNote',

```

```

243     null ,
244   );
245   const dest3AddressNote = this.props.navigation.getParam(
246     'dest3AddressNote' ,
247     null ,
248   );
249
250   const sourceMobile = this.props.navigation.getParam('sourceMobile' , null);
251   const destMobile = this.props.navigation.getParam('destMobile' , null);
252   const dest2Mobile = this.props.navigation.getParam('dest2Mobile' , null);
253   const dest3Mobile = this.props.navigation.getParam('dest3Mobile' , null);
254
255   // var date = new Date();
256   // var month = new Date().getMonth();
257   // var year = new Date().getFullYear();
258   // var dateToday = new Date().getDate();
259   // var timePostHr = date.getHours() + 1;
260   // console.log(typeof date);
261   // console.log(date);
262   // console.log(date.getHours() + 1);
263   // console.log(date.getUTCDate());
264   // var dateForm =
265   //   year.toString() + '-' + month.toString() + '-' + dateToday.toString();
266   // console.log(dateForm);
267   // console.log(typeof dateForm);
268
269   return (
270     <View style={styles.container}>
271       <ScrollView>
272         <Text style={styles.headerText}>schedule Details </Text>
273         <Card>
274           <TextInput
275             style={{borderBottomWidth: 1, height: 40}}
276             placeholder="Package Name"
277             onChangeText={(text) => {
278               this.setState({packageName: text});
279             }}></TextInput>
280         </Card>
281         <Card style={{borderRadius: 8}}>
282           <Text>Weight: Upto {this.state.weightValue} Kg</Text>
283           <Slider
284             value={this.state.weightValue}
285             //onValueChange={setWeight}
286             maximumValue={20}
287             minimumValue={1}
288             step={5}
289             onValueChange={(value) => {
290               if (value == 6 || value == 11 || value == 16) {
291                 this.setState({weightValue: value - 1});
292               } else if (value == 1 || value == 20) {
293                 this.setState({weightValue: value});
294               }
295             }}
296             thumbStyle={{
297               height: 25,
298               width: 25,
299               backgroundColor: '#F1948A' ,
300             }}
301           />
302         </Card>
303

```

```

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         onChange={(date) => {
323           this.setState({scheduledDateTime: date});
324         }}
325         console.log(typeof date);
326       // }
327     )}
328   </View>
329 </Card>
330
331 <Card>
332   <Text>Total: 50</Text>
333 </Card>
334 <Text style={styles.headerText}>Package Dimensions </Text> */}
335 { /* <TextInput
336   style={styles.textBox}
337   onChangeText={(text) => {
338     this.setState({dimLength: text});
339   }}
340   placeholder="Length"></TextInput>
341 <TextInput
342   style={styles.textBox}
343   onChangeText={(text) => {
344     this.setState({dimWidth: text});
345   }}
346   placeholder="Width"></TextInput>
347 <TextInput
348   style={styles.textBox}
349   onChangeText={(text) => {
350     this.setState({dimHeight: text});
351   }}
352   placeholder="Height"></TextInput>
353 <TextInput
354   style={styles.textBox}
355   onChangeText={(text) => {
356     this.setState({weight: text});
357   }}
358   placeholder="Weight Approx"></TextInput> */}
359 { /* <TextInput
360   style={styles.textBox}
361   onChangeText={(text) => {
362     this.setState({packageName: text});
363   }}
364 }

```

```

365     placeholder="Give a name to your Package"></TextInput > */}
366 </ScrollView>
367 <View
368     style={{
369         width: 370,
370         backgroundColor: 'transparent',
371         position: 'absolute',
372         top: '85%',
373         margin: 20,
374         // left: '10%',
375         // zIndex: 10,
376     }}>
377     <Button
378         onPress={() => {
379             this.saveOrder(
380                 sourceName,
381                 destName,
382                 dest2Name,
383                 dest3Name,
384                 noStops,
385                 sourceCord,
386                 destCord,
387                 dest2Cord,
388                 dest3Cord,
389                 sourceAddressNote,
390                 destAddressNote,
391                 dest2AddressNote,
392                 dest3AddressNote,
393                 sourceMobile,
394                 destMobile,
395                 dest2Mobile,
396                 dest3Mobile,
397             );
398             // this.navigateToSchedule(
399             //     sourceName,
400             //     destName,
401             //     dest2Name,
402             //     dest3Name,
403             //     noStops,
404             //     sourceCord,
405             //     destCord,
406             //     dest2Cord,
407             //     dest3Cord,
408             // );
409         }}
410         title="Place Order"></Button>
411     </View>
412 </View>
413 );
414 }
415 }
416
417 const styles = StyleSheet.create({
418     container: {
419         backgroundColor: '#EBF5FB',
420         // backgroundColor: '#E9967A',
421         height: '100%',
422     },
423     textBox: {
424         height: 40,
425         borderColor: 'gray',

```

```
426     borderWidth: 1,
427
428     margin: 20,
429   },
430   multiLineBox: {
431     // height: 40,
432     borderColor: 'gray',
433     borderWidth: 1,
434
435     marginLeft: 20,
436     marginRight: 20,
437   },
438   headerText: {
439     textAlign: 'center',
440     margin: 10,
441   },
442   buttonStyle: {
443     width: 40,
444     borderColor: 'gray',
445     borderWidth: 1,
446
447     margin: 20,
448   },
449 });
450
451 export default scheduleScreen;
```

## 6.4 Your Order

```

1  import React, {Component} from 'react';
2  import {
3    Text,
4    View,
5    TextInput,
6    StyleSheet,
7    Button,
8    FlatList,
9  } from 'react-native';
10
11 class yourOrders extends Component {
12   state = {packageName: ''};
13   packName = () => {
14     var pName = this.props.navigation.getParam('packageName');
15     this.setState({packageName: pName});
16   };
17
18   render() {
19     const pName = this.props.navigation.getParam('packageName', 'Package1');
20     console.log(pName);
21     var date = new Date().getDate();
22     var hours = new Date().getHours(); //To get the Current Hours
23     var min = new Date().getMinutes();
24
25     return (
26       <View>
27         <Text style={styles.headerText}>Your Orders</Text>
28
29         <FlatList
30           data={['1']}
31           style={styles.container}
32           renderItem={({item}) => (
33             <View style={styles.rowStyle}>
34               <Text style={styles.rightText}>{pName}</Text>
35               <Text style={styles.leftText}>{hours}</Text>
36               <Text>:{min}</Text>
37             </View>
38           )}
39           //keyExtractor={item => item.id}
40         />
41         /* <FlatList
42           data={this.state.placeResult}
43           renderItem={({item}) => (
44             <TouchableHighlight onPress={() => this.getCoord(item.key)}>
45               <Text key={item.key.toString()} bottomDivider>
46                 {item.name}
47               </Text>
48             </TouchableHighlight>
49           )}
50         /> */
51       </View>
52     );
53   }
54 }
55
56 const styles = StyleSheet.create({
57   container: {
58     padding: 10,
59     marginTop: 3,
60     backgroundColor: '#AED6F1',

```



```
60 // backgroundColor: '#d9f9b1',
61
62 // alignItems: 'center',
63 },
64 rightText: {
65   marginRight: 'auto',
66 },
67 leftText: {
68   textAlign: 'right',
69 },
70 headerText: {
71   textAlign: 'center',
72   margin: 10,
73 },
74 rowStyle: {
75   flexDirection: 'row',
76 },
77 });
78
79 export default yourOrders;
```



## 6.5 Admin Navigator

```

1  //import { StatusBar } from "expo-status-bar";
2  import React from 'react';
3  import { StyleSheet, Text, View } from 'react-native';
4  import AsyncStorage from '@react-native-async-storage/async-storage';
5  import { createAppContainer } from 'react-navigation';
6  import { createStackNavigator } from 'react-navigation-stack';
7  import { createSwitchNavigator } from 'react-navigation';
8  import { createDrawerNavigator } from 'react-navigation-drawer';
9  import { createBottomTabNavigator } from 'react-navigation-tabs';
10 adminDashboard from './src/adminScreens/adminDashboard';
11 import acceptDriver from './src/adminScreens/acceptDriver';
12 import viewDriver from './src/adminScreens/viewDriver';
13 import viewDetailDriver from './src/adminScreens/viewDriverDetails';
14 import verifyDetailDriver from './src/adminScreens/acceptDriverDetail';
15 import drawerContentComponents from './src/screens/drawerScreen';
16 //import orderDetails from './src/screens/orderDetails';
17 import yourOrders from './src/screens/yourOrders';
18 import Icon from 'react-native-vector-icons/Ionicons';
19
20 const detailsStackNavigator = createStackNavigator({
21   adminList: {
22     screen: adminDashboard,
23     navigationOptions: {
24       headerShown: false,
25     },
26   },
27   Accept: {
28     screen: acceptDriver,
29   },
30   ViewDriver: {
31     screen: viewDriver,
32   },
33   ViewDetailDriver: {
34     screen: viewDetailDriver,
35   },
36   VerifyDetail: {
37     screen: verifyDetailDriver,
38   },
39 });
40
41 const adminTabNavigator = createBottomTabNavigator({
42   BookCourier: detailsStackNavigator,
43
44   Schedule: {screen: yourOrders},
45 });
46
47 // const AppDrawerNavigator = createDrawerNavigator({
48 //   contentComponent: drawerContentComponents,
49 // });
50
51 const adminStackNavigator = createStackNavigator({
52   AdminCourio: {
53     screen: adminTabNavigator,
54     navigationOptions: ({ navigation }) => {
55       // headerShown: false;
56       return {
57         headerLeft: (
58           <Icon
59             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

```

1  import React, {Component} from 'react';
2  import {
3    Text,
4    View,
5    TextInput,
6    StyleSheet,
7    Button,
8    Picker,
9    ScrollView,
10 } from 'react-native';
11 import AsyncStorage from '@react-native-async-storage/async-storage';
12
13 class SignupView extends Component {
14   constructor(props) {
15     super(props);
16
17     this.state = {
18       name: '',
19       email: '',
20       password: '',
21       confPass: '',
22       phoneNo: '',
23       userType: '',
24       validEmail: true,
25       validPassword: true,
26       validConfPass: true,
27       validPhoneNo: true,
28
29       validInput: true,
30     };
31   }
32   validateEmail(email) {
33     var re = /\S+@\S+\.\S+\/;
34     console.log('Validating', re.test(email));
35
36     if (re.test(email) !== true) {
37       this.setState({validEmail: false});
38     } else {
39       this.setState({validEmail: true});
40     }
41
42     return re.test(email);
43   }
44   validatePhoneNo(phoneNo) {
45     if (this.state.phoneNo.length !== 10) {
46       console.log('Invalid PhoneNo. ');
47       this.setState({validPhoneNo: false});
48       return false;
49     } else {
50       this.setState({validPhoneNo: true});
51     }
52   }
53   // validateUserType(userType) {
54   //   if (this.state.userType.length > 0) {
55   //     console.log('Invalid userType. ');
56   //     this.setState({validUserType: false});
57   //     return false;
58   //   } else {
59   //     this.setState({validUserType: true});

```

```
60 // }
61 // }
62 validatePassword(password) {
63   if (this.state.password.length < 8) {
64     console.log('Minimum Password length should be 8');
65     this.setState({validPassword: false});
66     return false;
67   } else {
68     this.setState({validPassword: true});
69   }
70 }
71 validateConfPass(confPass) {
72   if (this.state.password !== this.state.confPass) {
73     console.log('Password not Matched');
74     this.setState({validConfPass: false});
75     return false;
76   } else {
77     this.setState({validConfPass: true});
78   }
79 }
80 validateForm() {
81   console.log(
82     '-----',
83     this.state.name,
84     this.state.email,
85     this.state.phoneNo,
86     this.state.password,
87     this.state.confPass,
88   );
89   var val = this.validateEmail(this.state.email);
90
91   if (
92     this.validEmail == false ||
93     this.validPhoneNo == false ||
94     this.validPassword == false ||
95     this.validConfPass == false
96   ) {
97     console.log('Error');
98     return 0;
99   } else {
100     if (this.state.userType == 'user') {
101       this.sendToDB();
102     }
103   }
104 }
105 sendToDB = async () => {
106   console.log('-----');
107   const body = JSON.stringify({
108     name: this.state.name,
109     email: this.state.email,
110     password: this.state.password,
111     phoneNo: this.state.phoneNo,
112   });
113
114   const headers = {
115     Accept: 'application/json',
116     'Content-Type': 'application/json',
117   };
118
119   const settings = {
120     method: 'POST',
```

```

121     headers ,
122     body ,
123   };
124
125   try {
126     const fetchResponse = await fetch(
127       'http://192.168.0.108:3000/signup ',
128       settings ,
129     );
130     const data = await fetchResponse.json();
131     console.log('+++++', data);
132     if (data.token) {
133       this.onSignupSuccess(data);
134     }
135
136     return data; // your response data
137   } catch (e) {
138     console.log(e);
139     return e; // handle your error here
140   }
141 };
142
143 onSignupSuccess = async (data) => {
144   console.log('SignUp Success', data);
145   await AsyncStorage.setItem('token', data.token);
146   this.props.navigation.navigate('Dashboard');
147
148   const value = await AsyncStorage.getItem('token');
149   console.log('$$$$$$$$$$$$$$$$$', value);
150 };
151
152 render() {
153   return (
154     <View>
155       <ScrollView>
156         <Text style={styles.appName}>Courio </Text>
157         <Text style={styles.headerText}>Signup </Text>
158         <Text style={styles.pickerHeader}>User Type </Text>
159         <View style={styles.pickerStyle}>
160           <Picker
161             selectedValue={this.state.userType}
162             style={styles.pickerStyle}
163             mode="dropdown"
164             onChange={(itemValue, itemIndex) => {
165               this.setState({userType: itemValue});
166               // this.validateUserType(this.state.userType);
167             }}>
168             <Picker.Item label="Customer" value="user" />
169             <Picker.Item label="Driver" value="driver" />
170           </Picker>
171         </View>
172
173         <TextInput
174           placeholder="Name"
175           style={styles.textBox}
176           onChangeText={(text) => {
177             this.setState({name: text});
178           }}></TextInput>
179
180         <TextInput
181           placeholder="Email"

```

```

182     style={styles.textBox}
183     onChangeText={({text} => {
184         this.setState({email: text});
185     })}
186     onEndEditing={({text} => {
187         this.validateEmail(this.state.email);
188     })}></TextInput>
189 <View>
190     {this.state.validEmail ? null : <Text>Invalid Email</Text>}
191 </View>
192
193 <TextInput
194     placeholder="Phone No."
195     style={styles.textBox}
196     onChangeText={({text} => {
197         this.setState({phoneNo: text});
198     })}
199     onEndEditing={({text} => {
200         this.validatePhoneNo(text);
201     })}></TextInput>
202 <View>
203     {this.state.validPhoneNo ? null : <Text>Invalid Phone No</Text>}
204 </View>
205 <TextInput
206     secureTextEntry={true}
207     placeholder="Password"
208     style={styles.textBox}
209     onChangeText={({text} => {
210         this.setState({password: text});
211     })}
212     onEndEditing={({text} => {
213         this.validatePassword(text);
214     })}></TextInput>
215 <View>
216     {this.state.validPassword ? null : (
217         <Text>Password length must be greater than equal to 8</Text>
218     )}
219 </View>
220 <TextInput
221     secureTextEntry={true}
222     placeholder="Confirm Password"
223     style={styles.textBox}
224     onChangeText={({text} => {
225         this.setState({confPass: text});
226     })}
227     onEndEditing={({text} => {
228         this.validateConfPass(text);
229     })}></TextInput>
230 <View>
231     {this.state.validConfPass ? null : (
232         <Text>Password didn't match</Text>
233     )}
234 </View>
235 </ScrollView>
236 <View
237     style={{
238         width: 90,
239         backgroundColor: 'transparent',
240         // position: 'absolute',
241         top: '10%',
242         left: '40%',

```

```

243     zIndex: 10,
244   }}>
245   <Button
246     onPress={() => {
247       // this.props.navigation.navigate('Dashboard');
248       console.log(
249         '-----',
250         this.state.name,
251         this.state.email,
252         this.state.phoneNo,
253         this.state.password,
254         this.state.confPass,
255       );
256       this.validateForm();
257     }}
258     title="Register"></Button>
259 </View>
260 </View>
261 );
262 }
263 }
264 const styles = StyleSheet.create({
265   container: {
266     flex: 1,
267     padding: 24,
268     backgroundColor: '#eaeaea',
269   },
270   headerText: {
271     textAlign: 'center',
272     margin: 10,
273   },
274   appName: {
275     textAlign: 'center',
276     margin: 20,
277     fontSize: 20,
278   },
279   textBox: {
280     height: 40,
281     borderColor: '#e29578',
282     backgroundColor: 'white',
283     borderWidth: 1,
284     margin: 20,
285   },
286   pickerHeader: {
287     // marginLeft: 20,
288     // marginRight: 20,
289     margin: 16,
290   },
291   pickerStyle: {
292     marginLeft: 20,
293     marginRight: 20,
294
295     borderColor: '#e29578',
296     borderWidth: 1.5,
297     color: '#168aad',
298   },
299   pickerItem: {
300     marginLeft: 20,
301     marginRight: 20,
302   },
303 });

```



304

305 `export default SignupView;`



## 6.7 Swipe Screen

```

1  import * as React from 'react';
2  import {StyleSheet, Text, View, Button} from 'react-native';
3  import Animated from 'react-native-reanimated';
4  import BottomSheet from 'reanimated-bottom-sheet';
5
6  export default function App() {
7    const renderContent = () => (
8      <View
9        style={{
10         backgroundColor: 'white',
11         padding: 16,
12         height: 450,
13       }}>
14        <Text>Swipe down to close</Text>
15      </View>
16    );
17
18    const sheetRef = React.useRef(null);
19
20    return (
21      <
22        <View
23          style={{
24            flex: 1,
25            backgroundColor: 'papayawhip',
26            alignItems: 'center',
27            justifyContent: 'center',
28          }}>
29          <Button
30            title="Open Bottom Sheet"
31            onPress={() => sheetRef.current.snapTo(0)}
32          />
33        </View>
34        <BottomSheet
35          ref={sheetRef}
36          snapPoints={[450, 300, 100]}
37          initialSnap={0}
38          enabledGestureInteraction={true}
39          enabledBottomClamp={true}
40          borderRadius={50}
41          renderContent={renderContent}
42        />
43      </>
44    );
45  }

```

## 6.8 Welcome Screen

```

1  import React, {Component} from 'react';
2  import {Text, View, TextInput, StyleSheet, Button} from 'react-native';
3  import AsyncStorage from '@react-native-async-storage/async-storage';
4
5  class Welcome extends Component {
6    constructor(props) {
7      super(props);
8
9      this.state = {
10       username: '',
11       password: '',
12     };
13   }
14
15   validateUser = async () => {
16     console.log('*****', this.state.username, this.state.password);
17     const body = JSON.stringify({
18       email: this.state.username,
19       password: this.state.password,
20     });
21
22     const headers = {
23       Accept: 'application/json',
24       'Content-Type': 'application/json',
25       Authorization: 'Bearer',
26     };
27
28     const settings = {
29       method: 'POST',
30       headers,
31       body,
32     };
33
34     try {
35       const fetchResponse = await fetch(
36         'http://192.168.0.108:3000/signin',
37         settings,
38       );
39       const data = await fetchResponse.json();
40       console.log('+++++', data);
41       if (data.token) {
42         this.onLoginSuccess(data);
43       }
44
45       return data; // your response data
46     } catch (e) {
47       console.log(e);
48       return e; // handle your error here
49     }
50   };
51   onLoginSuccess = async (data) => {
52     console.log('Login Success', data);
53     await AsyncStorage.setItem('token', data.token);
54     this.props.navigation.navigate('Dashboard');
55
56     const value = await AsyncStorage.getItem('token');
57     console.log('$$$$$$$$$$$$$$$$$', value);
58   };
59

```

```

60  render() {
61    return (
62      <View>
63        <Text style={styles.appName}>Courio </Text>
64        <Text style={styles.headerText}>Email </Text>
65        <TextInput
66          placeholder="Username"
67          style={styles.textBox}
68          autoCapitalize="none"
69          autoComplete={false}
70          onChangeText={(text) => {
71            this.setState({username: text});
72          }}></TextInput>
73        <Text style={styles.headerText}>Password </Text>
74        <TextInput
75          secureTextEntry={true}
76          placeholder="Password"
77          style={styles.textBox}
78          autoCapitalize="none"
79          autoComplete={false}
80          onChangeText={(text) => {
81            this.setState({password: text});
82          }}></TextInput>
83        <View
84          style={{
85            width: 90,
86            backgroundColor: 'transparent',
87            // position: 'absolute',
88            top: '10%',
89            left: '40%',
90            zIndex: 10,
91          }}>
92          <Button
93            onPress={() => {
94              console.log(
95                '-----',
96                this.state.username,
97                this.state.password,
98              );
99              this.validateUser();
100            }}
101            title="Login"></Button>
102        </View>
103        <View>
104          <Text
105            onPress={() => this.props.navigation.navigate('Signup')}
106            style={{color: 'blue', marginTop: 80}}>
107            Don't have an account SignUp
108          </Text>
109        </View>
110      </View>
111    );
112  }
113 }
114 const styles = StyleSheet.create({
115   container: {
116     flex: 1,
117     padding: 24,
118     backgroundColor: '#eaeaea',
119   },
120   textBox: {

```

```
121     height: 40,  
122     borderColor: 'gray',  
123     borderWidth: 1,  
124  
125     margin: 20,  
126   },  
127   headerText: {  
128     textAlign: 'center',  
129     margin: 10,  
130   },  
131   appName: {  
132     textAlign: 'center',  
133     margin: 20,  
134     fontSize: 20,  
135   },  
136   buttonStyle: {  
137     width: 40,  
138     borderColor: 'gray',  
139     borderWidth: 1,  
140  
141     margin: 20,  
142   },  
143 });  
144  
145 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 Notifica-tion	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

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

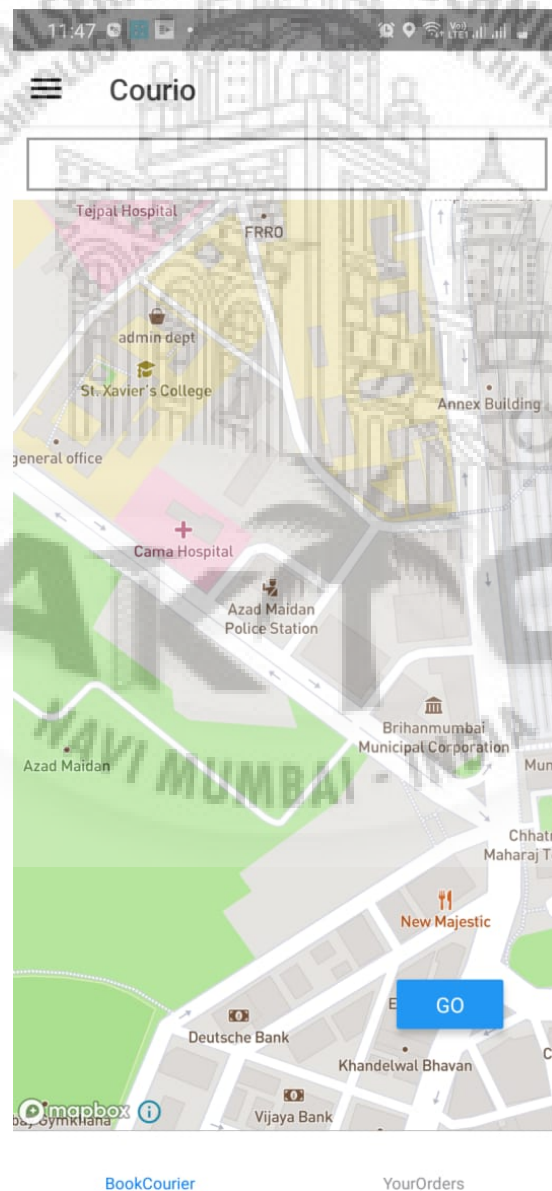


Figure 8.1: Select Pickup

## 8.2 Search Destination

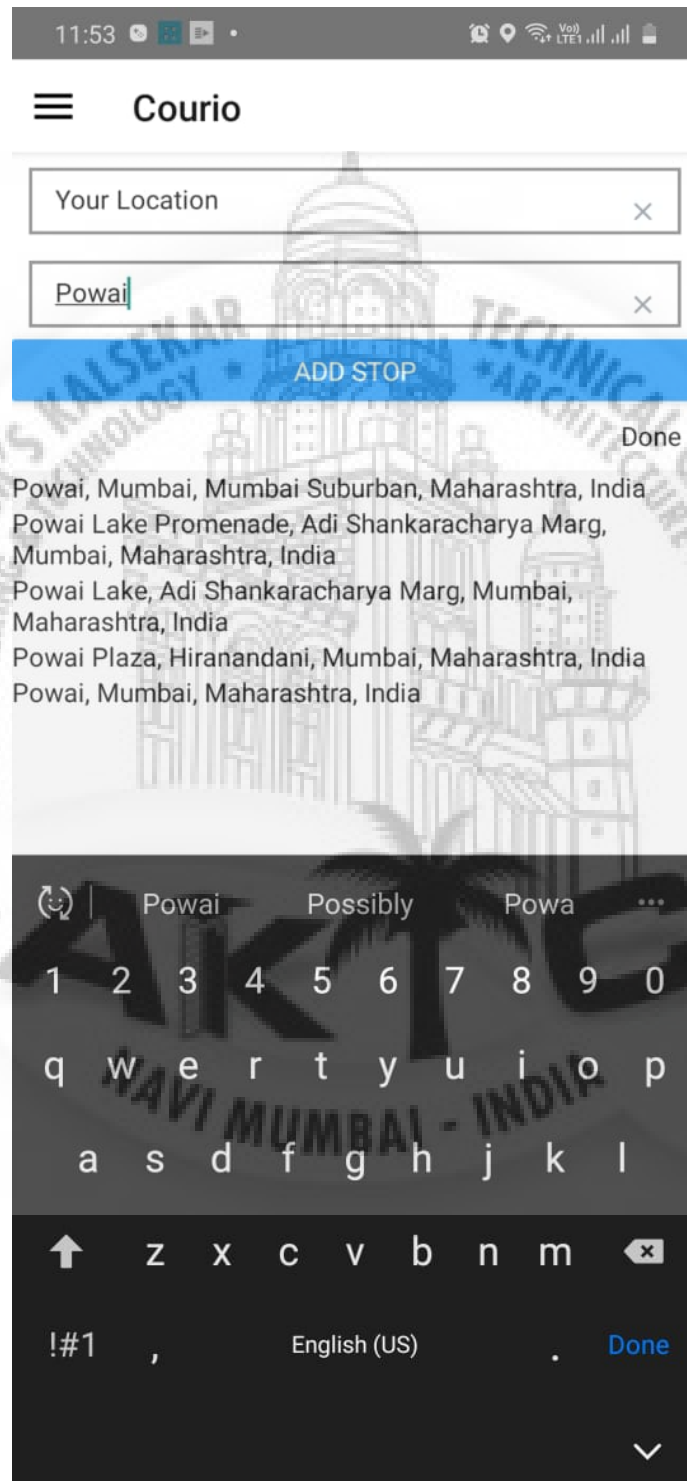


Figure 8.2: Search Destination

### 8.3 Add Multiple Delivery Points



Figure 8.3: Add Multiple Delivery Points

## 8.4 Map Route

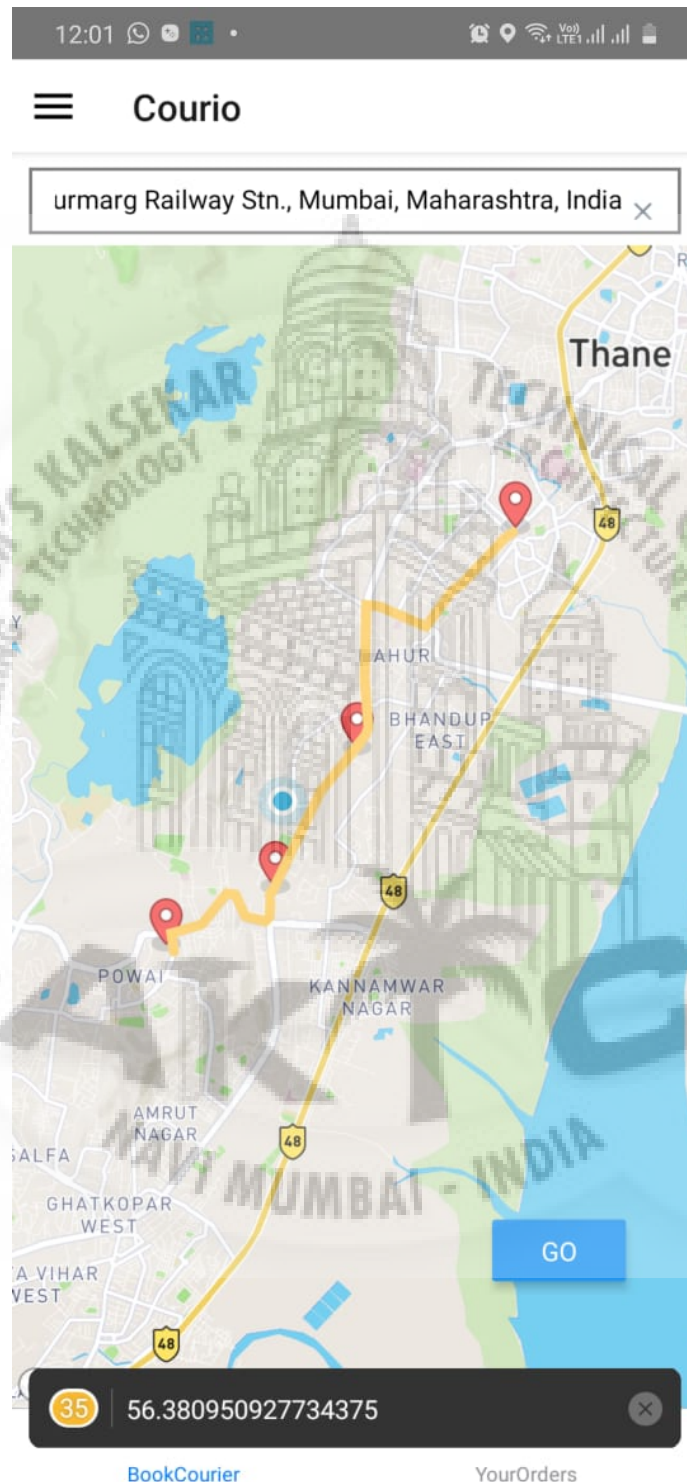


Figure 8.4: Map route

## 8.5 Add Address Details

The screenshot shows the Courio app interface. At the top, the status bar displays the time 12:01 and various system icons. Below the status bar, the app title "Courio" is visible. The main content area is titled "Address Details" and contains three address entries, each with a "Mobile No" field and a "Destination Address Note" field. The first entry is "Powai Plaza, Hiranandani, Mumbai, Maharashtra, India". The second entry is "Huma Big Cinemas Multiplex, Near Kanjurmarg Railway Stn., Mumbai, Maharashtra, India". The third entry is "Dreams Mall, Bhandup west, Mumbai, Maharashtra, India". A blue "NEXT" button is located at the bottom right of the third entry. The background of the app is watermarked with the AIKTC logo and text: "ANJUMAN - I - ISLAM'S KALSHAR TECHNICAL CAMPUS, NEW PANVEL PHARMACY + ARCHITECTURE + ENGINEERING & TECHNOLOGY".

12:01

Courio

Address Details

Powai Plaza, Hiranandani, Mumbai, Maharashtra, India

Mobile No

Source Address Note (Ex.Flat No., Compound, Street ...)

Huma Big Cinemas Multiplex, Near Kanjurmarg Railway Stn., Mumbai, Maharashtra, India

Receivers Mobile No

Destination Address Note (Ex.Flat No., Compound, Street ...)

Dreams Mall, Bhandup west, Mumbai, Maharashtra, India

Receivers Mobile No

Destination Address Note (Ex.Flat No., Compound, Street ...)

NEXT

Mulund Railway Station, Lokmanya Tilak Rd., Mumbai, Maharashtra, India

Figure 8.5: Address Details

## 8.6 Add Package Details

12:03

Courio

schedule Details

Pack 4

Weight: Upto 5 Kg

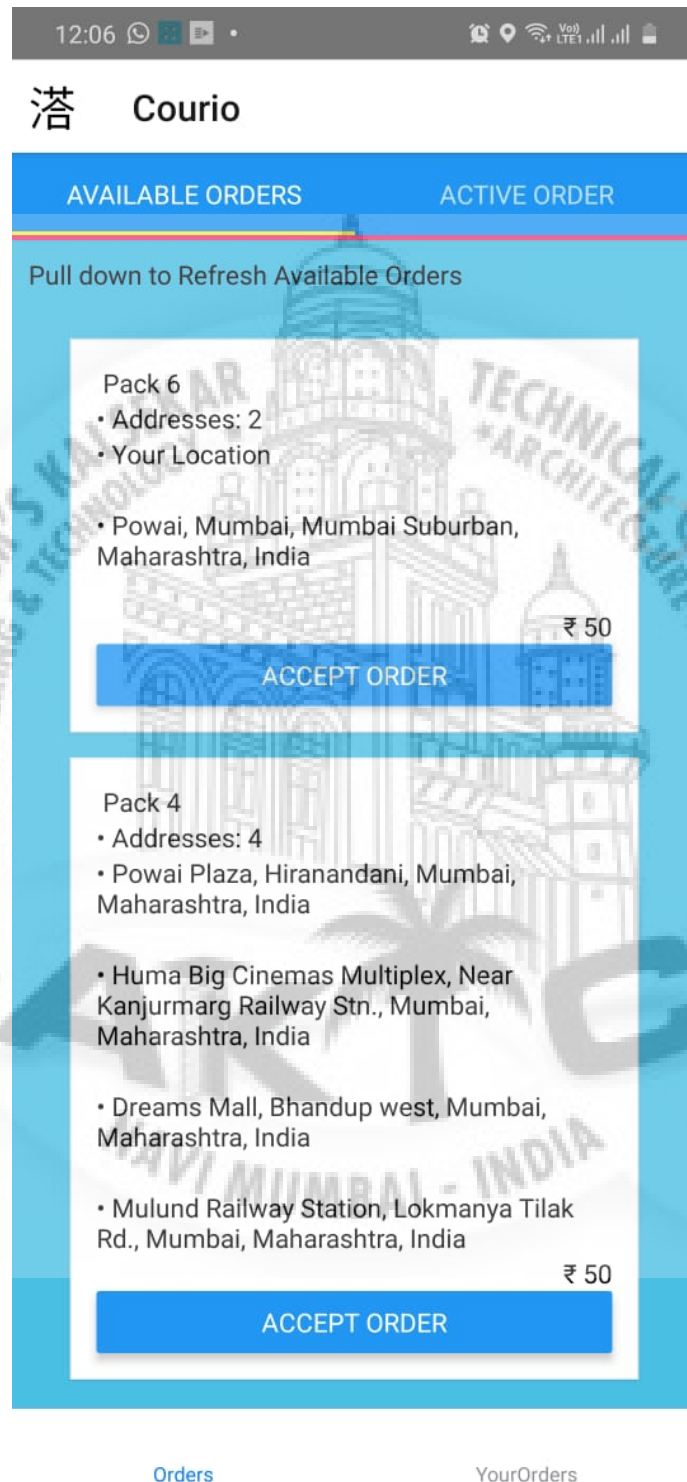
Book Now Schedule PickUp

Total: ₹ 50

Place Order

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

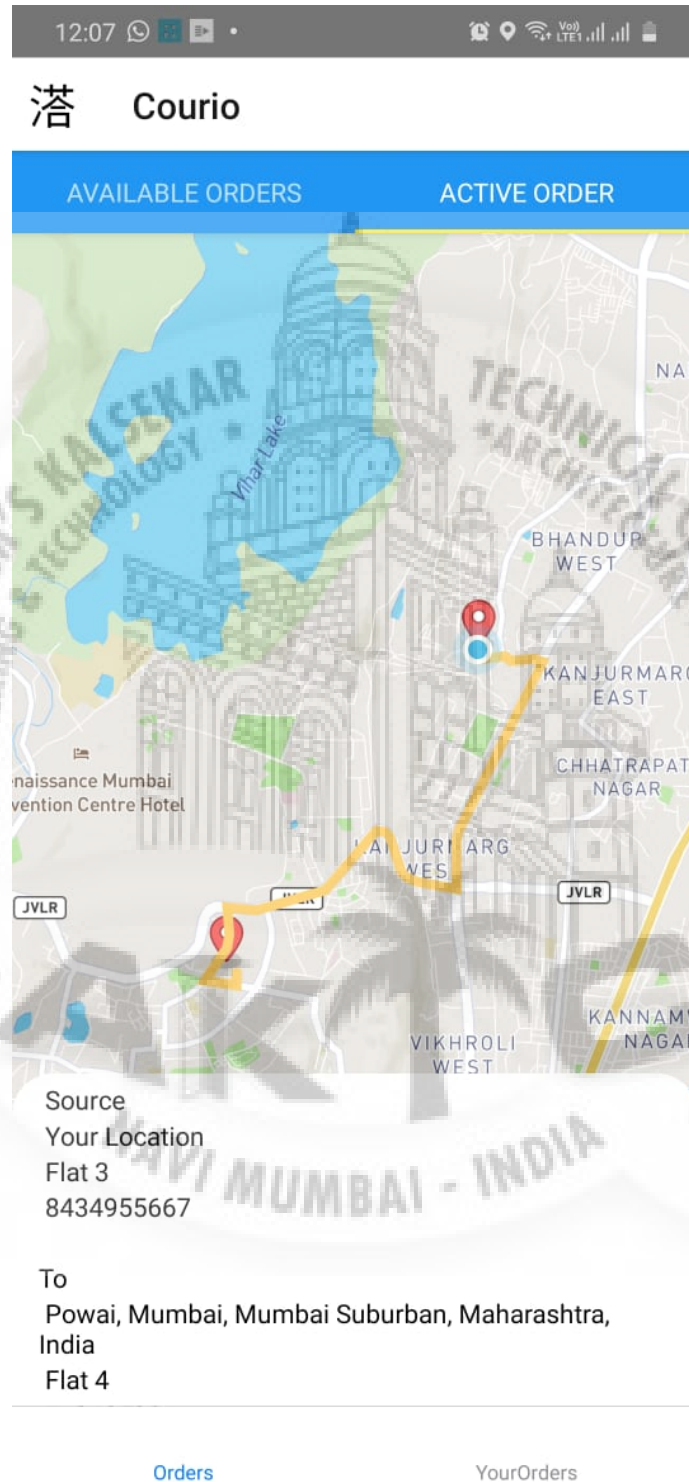


Figure 8.8: Order Route



## 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/>