A PROJECT REPORT

ON

”Online Grocery Delivery System”

Submitted to

UNIVERSITY OF MUMBAI

In Partial Fulfilment of the Requirement for the Award of

BACHELOR’S DEGREE IN COMPUTER ENGINEERING

BY

Nooras Fatima Mohammed Hashim Ansari	16CO01
Amina Sharif Sarang	16CO08
Mariyam Anis Shaikh	16CO12

UNDER THE GUIDANCE OF
PROF. Abdul Salam Shaikh

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
2019-2020

AFFILIATED TO

UNIVERSITY OF MUMBAI

A PROJECT II REPORT
ON

"ONLINE GROCERY DELIVERY SYSTEM"

Submitted to
UNIVERSITY OF MUMBAI

In Partial Fulfilment of the Requirement for the Award of

**BACHELOR'S DEGREE IN
COMPUTER ENGINEERING**

BY

Nooras Fatima Mohammed Hashim Ansari	16CO01
Amina Sharif Sarang	16CO08
Mariyam Anis Shaikh	16CO12

UNDER THE GUIDANCE OF
PROF. Abdul Salam Shaikh

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

2019-2020
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

"Online Grocery Delivery System"

submitted by

Nooras Fatima Mohammed Hashim Ansari	**16CO01**
Amina Sharif Sarang	**16CO08**
Mariyam Anis Shaikh	**16CO12**

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 2019-2020, under our guidance.

Date: / /

PROF. ABDUL SALAM SHAIKH
Project Supervisor

PROF. KALPANA BODKE
Project Coordinator

PROF. TABREZ KHAN
HOD, Computer Department

DR. ABDUL RAZAK HONNUTAGI
Director

External Examiner

Acknowledgements

We would like to take the opportunity to express our sincere thanks to our guide **Prof. Abdul Salam Shaikh**, 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.

We are grateful to him for his timely feedback which helped us track and schedule the process effectively. His time, ideas and encouragement that he gave helped us to complete our 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 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 us directly or indirectly during this course of work.

Nooras Fatima Mohammed Hashim Ansari
Amina Sharif Sarang
Mariyam Anis Shaikh

Project I Approval for Bachelor of Engineering

This project entitled ***Önline Grocery Delivery System¨*** by ***Nooras Fatima Mohammed Hashim Ansari, Amina Sharif Sarang and Mariyam Anis Shaikh*** is approved for the degree of ***Bachelor of Engineering in Department of Computer Engineering.***

Examiners

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

Supervisors

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

Chairman

....

Declaration

We declare that this written submission represents our ideas in our own words and where others ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We 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.

Nooras Fatima Mohammed Hashim Ansari
16CO01

Amina Sharif Sarang
16CO08

Mariyam Anis Shaikh
16CO12

ABSTRACT

Online shopping has become a trend in all the sectors of life. As businesses are expanding, it is important for businesses to understand the consumer's behaviour. The style of grocery shopping such as online grocery shopping made by retailers to attract customers. It is a technology which is popular mostly in big and metropolitan cities of India. We aim to provide techniques which can help the customers by spending less time on paying for products by customizing their experience of shopping and analyzing their feelings towards the shops of their choice. This system has three modules: Customer, Shopkeeper and Delivery Boy. This System focuses on building a website which can make the online grocery shopping a lot easier for customers.

This automated system provides many functionalities which will make it better than the previous systems available. The facilities provided will be–registering, signing in, searching, viewing recommended products based on different categories, reviewing the products according to your experience and ordering the grocery items with secured online transaction. It is a beneficial system for small retailers to expand their business as they can directly deliver the products to the customers in less time. Voice to text, smart list are some add-on features new for an online grocery store. Not only this customer can also track products. A smart list is a list of products based on previous history of frequently bought products

Keywords: Machine Learning, Natural language Processing (NLP), Text Analysis, Speech Recognition, Product Recommendations, Artificial intelligence.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Building a system provide an Online Grocery Shopping solution to buyers and sellers. Online Grocery delivery system will automate some of the basic actions of an online grocery store. The scope would be to give fundamental functionalities using a web application that is product recommendation, automated smart list, Voice search, effective UI, comment review and many more. A way for customer to start a new business by registering as a shopkeeper. Flexible system that includes all the small and big retailers.

This system will provide cost effective products to customers without going out. This is an automated system that provides features like product recommendation, automated smart list, voice search etc. It is a user friendly system. The sellers are authorized which will lower the chances of online shopping frauds. Problem of carrying heavy bags from shopping will finish. It offers Cash back. Season based recommendation to users. Users can give rating to product.

1.1 Purpose

Online grocery shopping is a new way of buying desired grocery products for household.This will give a platform for small retailers to expand their business. With this system user can buy grocery from their nearest preferred shop. User can buy the product from anywhere at any time. Shopkeeper can accept the order and pack it they can also add or delete the products. Delivery boy can see the product which was packed and accept that product and deliver it to desired customers. It reduces the time of customers. It provide functionalities such as product recommendation, voice over manually search. You can buy grocery and also review the products.

1.2 Project Scope

It automates some of the basic actions of an online grocery store. The scope would be to give fundamental functionalities using a web application that is product recommendation, automated smart list, Voice search, effective UI, comment review and many more. Customer can start new business by registering as a Shopkeeper. This system is time effective and reduce the work. The sellers are authorized which will lower the chances of online shopping frauds. Problem of carrying heavy bags from shopping will finish.

1.3 Project Goals and Objectives

1.3.1 Goals

- To provide better user interface.
- To provide flexible system to user.
- To provide more functionalities to user.
- Selling products online using a system.
- Promoting a service or product online.
- Providing product support or customer service.
- User can use it from anywhere through mobile or computer.

1.3.2 Objectives

Building a system that provide Online Grocery Shopping solution to consumers and vendors. It will automate some of the basic operations of an online grocery store. Objective would be to provide basic functionalities using a web application that is buying the product from system, tracking the product, add / delete the product from shopkeeper side, accept or rejection of a project, deliver the product to users, product recommendation, automated smart list, Voice search etc. A way for customer to start a new business by registering as a shopkeeper. Flexible system that includes all the small and big retailers.

1.4 Organization of Report

Chapter 1: Gives a brief introduction about our project.

Chapter 2 : Describes the literature review of the existing papers and the description about the application.

Chapter 3 : Discuss about the project planning and different roles and capability of the team members. Also talks about the budget of the project.

Chapter 4 : Describe the brief description of the srs and the other requirements of the project.

Chapter 5 : Shows the system design, functional requirements and different diagram of the project.

Chapter 6 : Shows Implementation of the websites and coding.

Chapter 7 : Shows the different testings performed and the problems faced. It also shows snapshots of the current working application.

Chapter 8 : Describes the closure to the book and tries to conclude the work in the project and also mentions the future scopes as to where it would be used Chapter.

Chapter 9 : Describes a steps guide about using the final report.

Chapter 2

Literature Survey

2.1 The Smart Shopping List

Harsha Jayawilal W.A. and Saminda Premeratne introduced 'The Smart Shopping List' is a mobile software service which helps the customers to fulfil their online grocery shopping experience by defeating the hindrance of writing down items on a mobile device or paper. In Smart Shopping List, several modules are used; Interactive Shopping List to add/remove or cross items, Shop Locator to assist the user to find an ideal supermarket where they can buy the products in one place, Product Recommender based on Apriori algorithm to notify the user of possible missing products or products they may be interested in and 'bring me!', which is a text-to-app feature that can help to share shopping lists between the users[1].There is a drawback is customer have to prepare a manual smart list that is user have to enter the products he/she wants to buy and add it to the list. So, An automated smart list which will create list according to frequently bought products of users[1].

2.1.1 Advantages of Paper

a. In Smart Shopping List, several modules are used: Interactive Shopping List to add/remove or cross items, Shop Locator to assist the user to find an ideal supermarket where they can buy the products in one place.

b. 'bring me!', which is a text-to-app feature.

2.1.2 Disadvantages of Paper

a. Customer have to prepare a manual smart list that is user have to enter the products he/she wants to buy and add it to the list.

b. An automated smart list which will create list according to frequently bought products of users[1].

2.1.3 How to overcome the problems mentioned in Paper

a. An automated smart list which will create list according to frequently bought products of user.

2.2 Smart Shopping Technologies for Indoor Markets

Ioana Rogojanu, Maria-Cristina Ditu , Adrian Pasat and George Suciu, explore several ways to execute on indoor supermarkets that can enhance the regular practice of shopping. They intend to provide different ways for reducing the time spent over a user to buy the goods and observing their behaviour towards the stores to customize the experience of them. It is an online assistance that helps to recognise the customer's viewpoint. Retailers responsibility is to understand the behaviour of their customer's hopes and expectations through NaturalLanguage Processing which is a form of Artificial Intelligence, namely to encode and observe thought on feedbacks by reflecting the circumstances and the conditions. Voice over text application is developed considering the vocabulary formed from words that can highlight and show great emotions like "right", "wrong", "offensive", ”healthy" and can also investigate if the sentence is true or false, and if the sentence is false to notice if the customer said "not right", "not wrong" and so on.. Review and feedback system using sentiment analysis which will classify the object based on negative and positive ratings[2].

2.2.1 Advantages of Paper

a. It provides different ways for reducing the time spent over a user to buy the goods.

b. Observing user behaviour towards the stores to customize the experience of them.

2.2.2 Disadvantages of Paper

a. Only implemented basic review system without sentiment analysis.

2.2.3 How to overcome the problems mentioned in Paper

a. Review and feedback system using sentiment analysis which will classify the object based on negative and positive ratings.

2.3 Assessing the Profitable Conditions of Online Grocery Using Simulation

Ahmed Alzubairi and Abdullah Alrabghi evaluated favourable situations to large retailers of Jeddah in Saudi Arabia for online grocery. They initiate different plans and evaluated them for online grocery by generating simualation models. The two major uncertainties are Operations challenges and cost uncertainty[3].

2.3.1 Advantages of Paper

a. They genrated simulations models for online grocery.

b. From the research of simulation they showed that simulation is a great decision support mechanism for foretelling and impersonating of Online Grocery Shopping prior of actual implementation.

2.3.2 Disadvantages of Paper

a. Customer authentication is available but not authenticating shopkeeper and his details.

2.3.3 How to overcome the problems mentioned in Paper

a. Along with customer authentication using Geo tagging to authenticate shopkeeper.

2.4 A Method to Study How Older Adults Navigate in an Online Grocery Shopping Site

This paper shows navigation of older adults on grocery website. It is a methodolgy for study to examine the behaviour of older adults while using an Online Grocery Shopping sites. This study includes examing the characteristics of existing Online Grocery Shopping sites, observing the perfomance of online grocery shopping sites using a chosen website they also took interviews to get more knowledge of parts that are asssociated with it and factors or actions that will help the navigation of older adults[4].

2.4.1 Advantages of Paper

a. Provide navigation of older adults on a grocery website.

b. Study of how the adults get lost on a website.

2.4.2 Disadvantages of Paper

a. Does not focus on age group other than adults.

b. Focus on only study, does not provide a solution to the problem.

2.4.3 How to overcome the problems mentioned in Paper

a. Can provide efficient user interface so that adults can use the website without any problem.

2.5 Artifical Intelligence – Created Automated Insights fro Customer Relationship Management

In this research paper, Suman kumar Deb, Ruchi Jain and Varsha Deb studied various AI enabled tools and finalize five Ai enabled tools to study the consumer awareness, effectiveness and Loyalty[5]. Five AI enable tools are Product Recommender, Virtual Agent, Email Management, Speech Recognition and Visual Perception.

2.5.1 Advantages of Paper

a. For developing such online system they finalize five AI tools like Product Recommender, Virtual Agent, Email Management, Speech Recognition and Visual Perception.

b. Provided awareness about AI tools.

2.5.2 Disadvantages of Paper

a. They have only research on this tools so there is no real implementation in this paper.

b. They have focused only on AI tools

2.5.3 How to overcome the problems mentioned in Paper

a. Try to implement recommender system in real world.

b. Try to implement speech recognition while searching the products.

2.6 Technical Review

Out system is based on Flask. Which is a mini web framework of python. For front-end we are using HTML and CSS. For backend we are using python and Machine Learning Algorithms. Here are description of technologies.

- Flask:
  Flask is an API of Python that allows to build up web-applications. It was developed by Armin Ronacher. Flask's framework is more explicit than Django's framework and is also easier to learn because it have less base code to implement a simple web-Application. A Web-Application Framework or Web Framework is the collection of modules and libraries that helps the developer to write applications without writing the low-level codes such as protocols, thread management, etc.

Figure 2.1: Flask

- Machine Learning:
  Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that which makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect.[10]

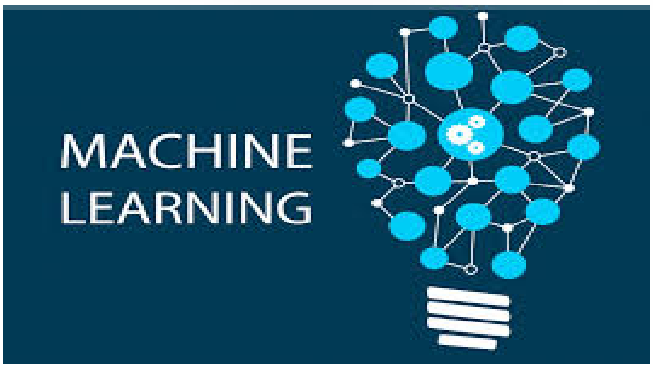


Figure 2.2: Machine Learning

- HTML

  HTML stands for Hyper Text Markup Language. It is used to design web pages using markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. Markup language is used to define the text document within tag which defines the structure of web pages.[11]

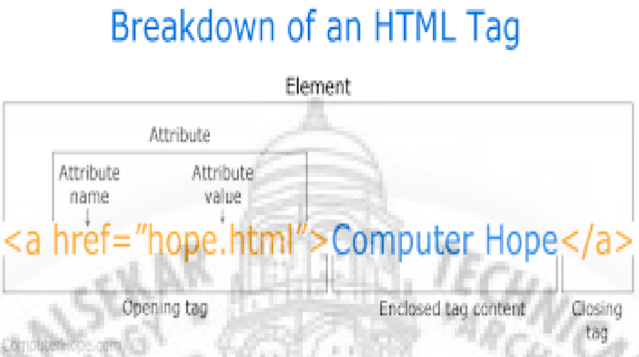


Figure 2.3: Html

- CSS

  Cascading Style Sheets, fondly referred to as CSS, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page.[12]

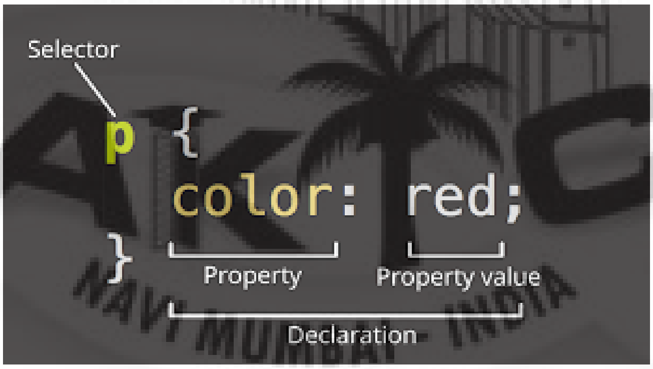


Figure 2.4: CSS

- Python

  Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.[13]

Figure 2.5: Flask

2.6.1 Advantages of Technology

- Flask:
  Flask have built-in development server and fast debugger, it is integrated support for unit testing, provides RESTful request dispatching and Jinja2 templating [9].

- Machine Learning:
  ML easily identifies trends and patterns. No human intervention needed (automation). It handles multi-dimensional and multi-variety data.Now a days it haves wide applications in day to day life.

- HTML:
  HTML is Easy to Learn and Use. It is Supported by all Browsers. IT is the Most Friendly Search Engine. It is Simple to Edit. It can Integrate Easily with Other Languages. It is Lightweight. It is Basic of all Programming Languages.

- CSS:
  Css is easier to maintain and update. Greater consistency in design. It have More formatting options. It is Lightweight code. Search engine optimization benefits due to CSS.

- Python:
  Python provides a large standard library which includes areas like internet protocols, string operations, web services tools and operating system interfaces. Python language is developed under an OSI-approved open source license, which makes it free to use and distribute, including for commercial purposes. Python offers excellent readability and uncluttered simple-to-learn syntax.

2.6.2 Reasons to use this Technology

- Flask:
  Flask is a minimalistic python framework for building web apps. It does not offer and bend you one way or another. Jinja2 is an awesome template framework, flask comes out of box support for that, but it does not force that way. It relies on middlewares, a pluggable approach to add functionalities as needed. So not so boated unlike django. [8].

- Machine Learning:
  As humans become more addicted to machines, we're witnesses to a new revolution that's taking over the world, and that is going to be the future of Machine Learning. More online businesses are integrating machine learning into their operations, with the bigger and established ones trailblazing the revolution. Machine learning has brought myriad opportunities and improved strategies to help business owners foster customer relationships and get more profit and conversions.ML easily identifies trends and patterns. No human intervention needed (automation). It handles multi-dimensional and multi-variety data.Now a days it haves wide applications in day to day life.

- HTML:
  HTML allows you to write clear and descriptive code, semantic code that allows you to easily separate meaning from style and content. HTML code ensures the proper formatting of text and images so that your Internet browser may display them as they are intended to look. Without HTML, a browser would not know how to display text as elements or load images or other elements. HTML also provides a basic structure of the page, upon which Cascading Style Sheets are overlaid to change its appearance.

- CSS:
  With CSS, we are able to create rules, and apply those rules to many elements within the website. CSS is Easy to Work. Css is easier to maintain and update.

- Python:
  Python provides a large standard library. It is one of the most easy language for understanding. Python have small function to do large task.

Chapter 3

Project Planning

3.1 Members and Capabilities

Table 3.1: Table of Capabilities

SR. No	Name of Member	Capabilities
1	Nooras Fatima Ansari	Mysql, Front-end, Backend, Flask, JS
2	Amina Sarang	Mysql, UI Design, Front-end
3	Mariyam Shaikh	Mysql, Front-end, Backend, Android

Work Breakdown Structure

3.2 Roles and Responsibilities

Table 3.2: Table of Responsibilities

SR. No	Name of Member	Role	Responsibilities
1	Nooras Fatima Ansari	Team Leader	Database design, Back-end, Front-end Development and Documentation
2	Amina Sarang	Member	Database design, UI Desing, Font-end development and Documentation
3	Mariyam Shaikh	Member	Database design, Back-end and Front-end Development

3.3 Assumptions and Constraints

3.3.1 Assumptions

- You will get all the resources you need.
- All of the groceries are fresh.
- The Delivery boy will deliver products on time.
- Site will be available 24*7.
- The Shopkeeper accepts the product order instantly after receiving.

- Query will be resolve within 2 days.

3.3.2 Constraints

- Authentic User can only buy the product.
- Authentic Shopkeeper only can become a seller.
- Authentic Delivery boy can become a delivery boy.
- After paying amount of groceries user can get the products.
- Lending goods is forbidden.

3.4 Project Management Approach

- We are following Iterative approach in our project.

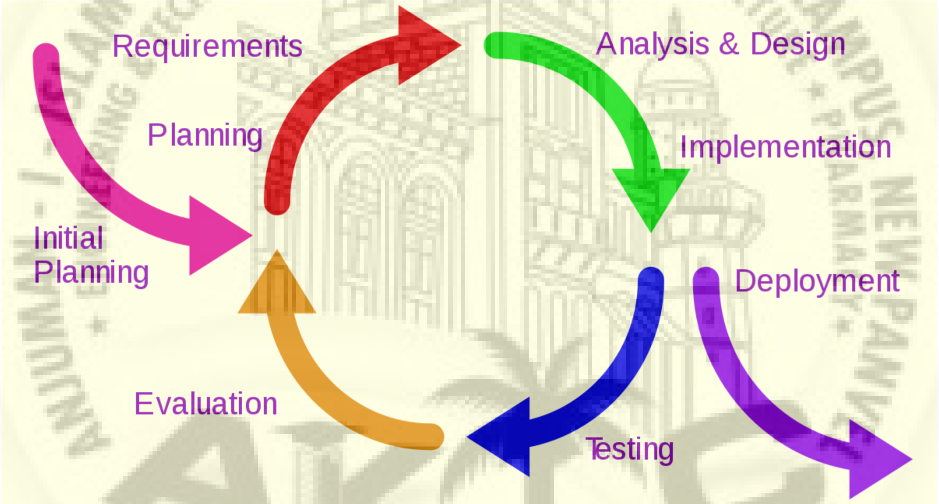


Figure 3.1: Iterative Approach

- Phases of Iterative approach are:

a. Planning Requirements:
In this phase, we planned the project approach and gathered requirements considering the target audience. In our project, the target audiences are people who buy grocery normally women. Customer requirements are effective UI, easy navigation, etc.

b. Analysis Design:
After the planning is complete, an analysis is performed to nail down the appropriate business logic, database models. In this phase, we perform analysis and design a wire frame of the website that is to be developed.

c. Implementation:
With the planning and analysis out of the way, the actual implementation and coding process is done in this phase. All planning, specification, and design up to this point is implemented and coded here.

d. Testing:
Once this current build iteration has been coded and implemented, the next step is to go through a series of testing procedures to identify and locate any potential bugs or issues that have cropped up.

e. Evalution:
Once all prior stages have been completed, it is time for a thorough evaluation of development up to this stage. This allows the entire team, as well as clients or other outside parties, to examine where the project is at, where it needs to be, what can or should change, and so on

3.5 Ground Rules for the Project

a. Each team member have to work together with others member.

b. Each team member can share past experience with other members.

c. Each team member have to work on assigned task.

d. Members can share their idea.

e. Team members have to report daily to respective leader.

f. Talk softly with other members.

g. Participate in meeting.

h. Inform the leader about unavailability.

3.6 Project Budget

The budget for this project is very low as most of the tools we use are open source. Following are the budget for the project.

a. Operating System : Linux mint(Open Source)

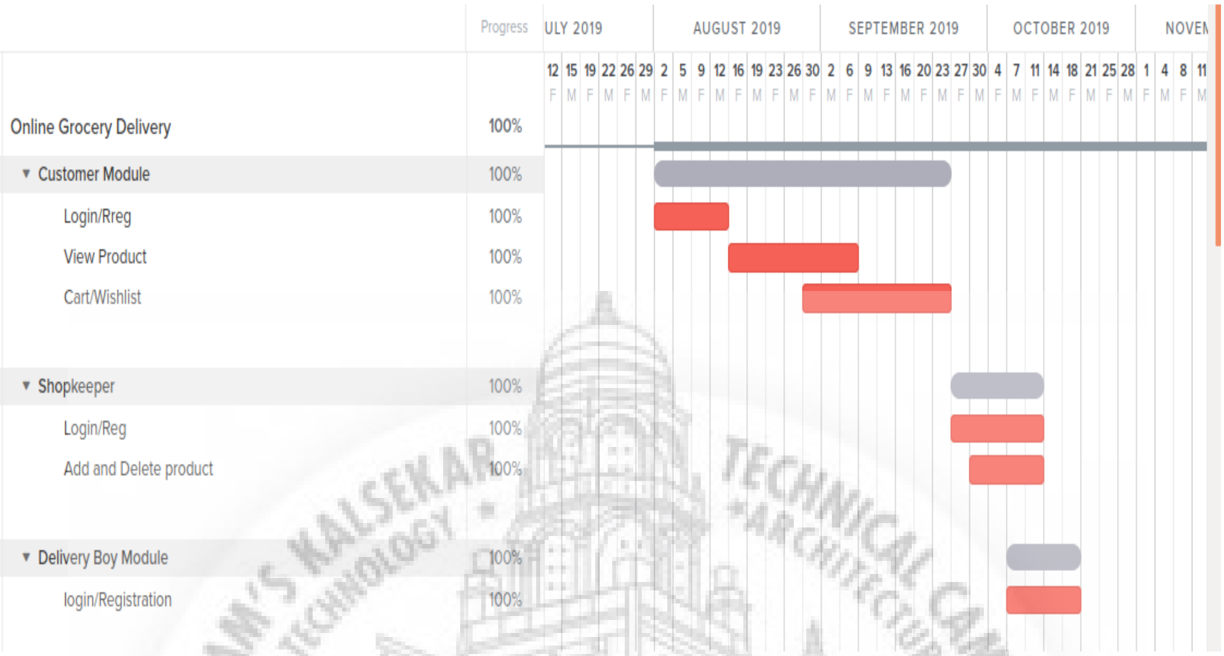
b. Python Programming Language(Open Source)

c. MySQL : Open Source

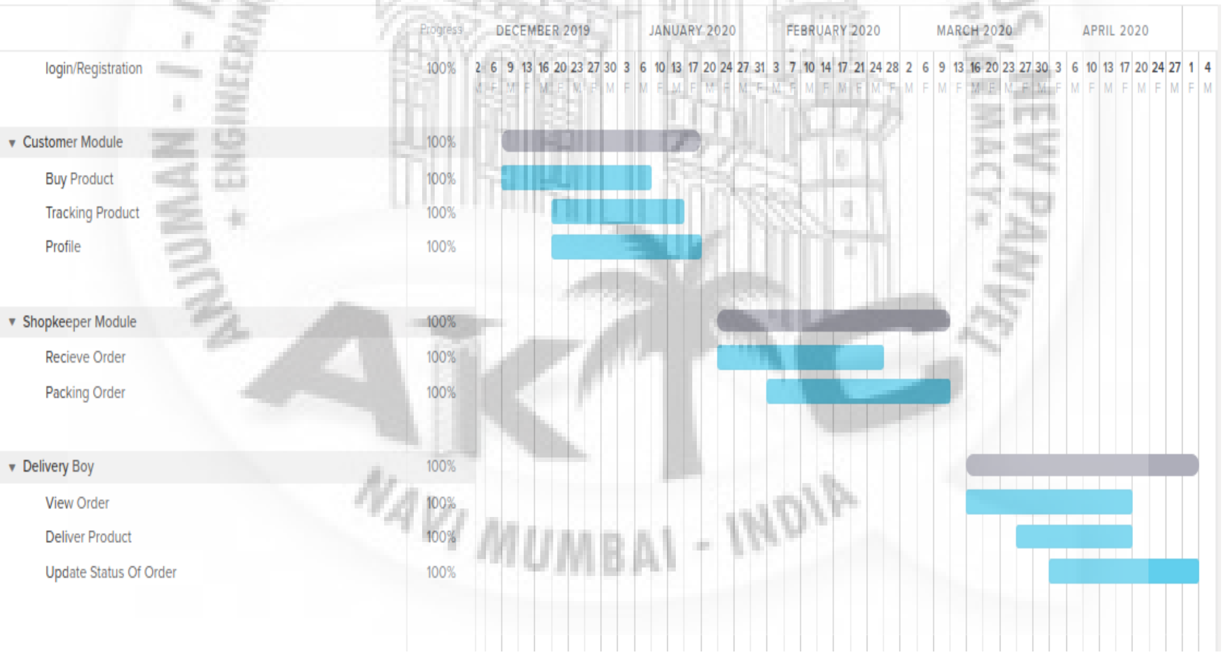
d. Frame Work : Flask(Open Source)

e. 0.50 dollar per month to host website through Amazon.

3.7 Project Timeline



	Progress
Online Grocery Delivery	100%
▾ Customer Module	100%
Login/Rreg	100%
View Product	100%
Cart/Wishlist	100%
▾ Shopkeeper	100%
Login/Reg	100%
Add and Delete product	100%
▾ Delivery Boy Module	100%
login/Registration	100%



	Progress
login/Registration	100%
▾ Customer Module	100%
Buy Product	100%
Tracking Product	100%
Profile	100%
▾ Shopkeeper Module	100%
Recieve Order	100%
Packing Order	100%
▾ Delivery Boy	100%
View Order	100%
Deliver Product	100%
Update Status Of Order	100%

Chapter 4

Software Requirements Specification

4.1 Overall Description

4.1.1 Product Perspective

Online Grocery Delivery System will automate some of the basic functionality of the grocery store. The aim is to provide a website through which the user can buy the product from anywhere. Whether they are in the office or at home or in school from anywhere they can buy the groceries. It replaces the offline grocery where people gathered in large quantities. User will be far away from this rush. Online grocery provides flexibility to users. Users don't have to carry heavy bags. If any grown-up people have difficulties so that they can buy from their home. Shopkeepers can add or delete products from the site. Shopkeeper pack the products. After packing delivery boy can deliver it to respective user.[1].

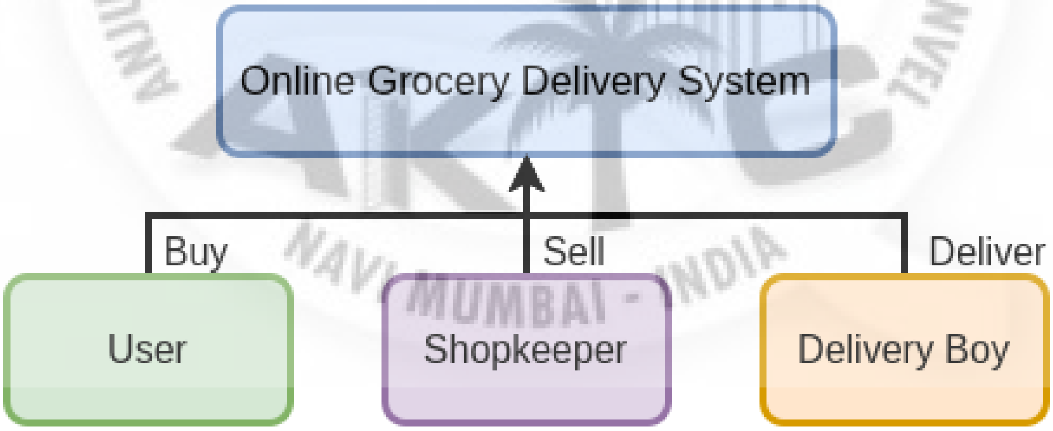


Figure 4.1: Major components of system

4.1.2 Product Features

Online Grocery System have many features like Smart List is generated according to the products that are frequently bought by user, Speech Recognition is used for identifying the verbal information and phrases and change it into a machine readable language, Product Recommendations is a filtering tool that can recommend every single user a list of relevant products that can increase shopping experience of user and save time. Text analysis on comments and review and Order batch time and limit. Online grocery system also provide option for customer to 'become a seller'.

4.1.3 User Classes and Characteristics

Different users will use the product differently depending on their needs and hence, the user class will change according to the need of the user and who the user is that is customer, shopkeeper or delivery boy. But the basic characteristics of the classes remain the same where the user will primary interact will three main class of product that is registration, enter the required details, get the pitch report.The rest are less important according to this three class. The characteristics of the class will change according to the user.

- If the user is customer he/she will interact with three main class that is registration, product shopping and payment.
- If the user is shopkeeper he/she will interact with three main class that is registration, product request handling and product update.Done
- If the user is delivery boy he will interact with three main class that is registration, delivery request handling and delivery.

4.1.4 Operating Environment

The environment in which the system will operate is platform-independent. The only important software that the use will need is any web browser where the user can use our system on web efficiently. Python version 3.6 is used. Html5 and CSS3 is used in Front end.

4.1.5 Design and Implementation Constraints

As the project is a website it was difficult to build front end and web end simultaneously as the front end is incomplete without the back end added. Database management is necessary to store and retrieve product and user details. Internet is necessary to operate the website. One can not access the website without authentication.

4.2 System Features

The major feature of our system is to provide a better user experience for grocery shopping. To automate the basic operations of online shopping.

4.2.1 System Feature

- User Authentication and Authorization
- Product Recommendation
- Delivery from a trusted shop

Description and Priority

- User Authentication:
  This feature will authenticate user for the further use of features provided by our system.
- Product recommendation:
  This feature will recommend user the products they frequently buy.
- Delivery from a trusted shop:
  This feature will allow user to buy products from their nearest preferred shop.

Stimulus/Response Sequences

- For Customer:
  a. The customer will login to the system.
  b. They will order products from their preferred shop.
- For Shopkeeper:
  a. The shopkeeper will be authorized with Geo tagging.
  b. The shopkeeper will login to the system.
  c. They will add or delete the product from inventory.
  d. They will accept the order request from customer.
- For Delivery Boy:
  a. Delivery Boy will login to the system.
  b. He will accept the delivery request and will deliver the product to the rightful owner.

Functional Requirements

- The software provides good graphical interface for the user. The user can buy grocery products with different features available.
- The user can view the system any time.

Hardware Requirements

Hard disk: 50GB
RAM: 2GB

Software Reuirements

Operating system: Linux, Windows
Tool: Visual Studio Code
Libraries: Flask, Flask mysqldb, wtforms
Database: Mysql

4.3 External Interface Requirements

4.3.1 User Interfaces

User interface that will accessible through any internet browser the major ones being Google, Chrome and Internet Explorer, Mozilla Firefox. Through such software users can access the sites and can take advantage of our Online Grocery Delivery System.

4.3.2 Hardware Interfaces

We don't required any hardware interface in our project. So we required only software interface in our project.

4.3.3 Software Interfaces

- Operating system: Linux, Windows, Android
- Tool: Visual Studio Code
- Libraries: Flask, Flask mysqldb, wtforms
- Database: Mysql

Through different operating system, developers can communicate with the Hardware such as laptop to develop a website. For writing developers can use Visual Studio code. We have been using Flask mini framework of python. So in this, we are using different libraries to develop applications. Also we are using Html, CSS in frontend. In backend we are using python and MySQL database.

4.3.4 Communications Interfaces

- The website support all type of browser.
- The interface between the database and the system will be done by using http. protocol.
- For sending OTP to user Fast2SMS API is used.

4.4 Nonfunctional Requirements

4.4.1 Performance Requirements

The system must be interactive and the delay involved must be less. When we connecting to the server the delay is because the data stored or manage online very safely.

4.4.2 Safety Requirements

If there is any damage to the servers then the whole system will go down. The database should be periodically maintained and have to keep upon it. The data which is updated by the user would be committed in the database.

4.4.3 Security Requirements

The major security requirements for the system will be the safeguarding of the user data from any kind of exploit. In order to protect the user data the data is not stored in local databases we will be storing in the cloud for better security.

Chapter 5

System Design

5.1 System Requirements Definition

Online grocery system solves the problem for those people who are not able to go to shops for shopping. It helps them to order their groceries from their home or offices. Online Grocery System should provides functionalities like buying products from the website, location tracking by the user. Shopkeepers must be able to add or delete the products. Through the system, shopkeepers can accept the products. Delivery boy must get all current details of products like from where he can pick and deliver to respective users.

5.1.1 Functional requirements

1. The application provides a better User interface so that the user can buy the product from the Online Grocery Shopping System.
2. One can search product and can track the product through the system.
3. The system will be available 24*7 so that they can access it any time
4. Shopkeepers can add or delete products according to the product availability.
5. There are various products available on the website that user can buy.

Use-case Diagram

Use case diagram are usually referred to as behaviour diagram used to describe a set of actions(use case) that some system or systems(subject) should or can perform in collaboration with one or more external users of the system(actors). Each use case should provide some observable and valuable result to the actors or other stakeholder of the system.

The below figure shows the use case diagram of our system which contains the following component.

- Authentication
- Product View
- Searching
- Checkout
- Tracking
- Add and Delete product
- Pack Order
- Pick Order
- Delivery
- Payment

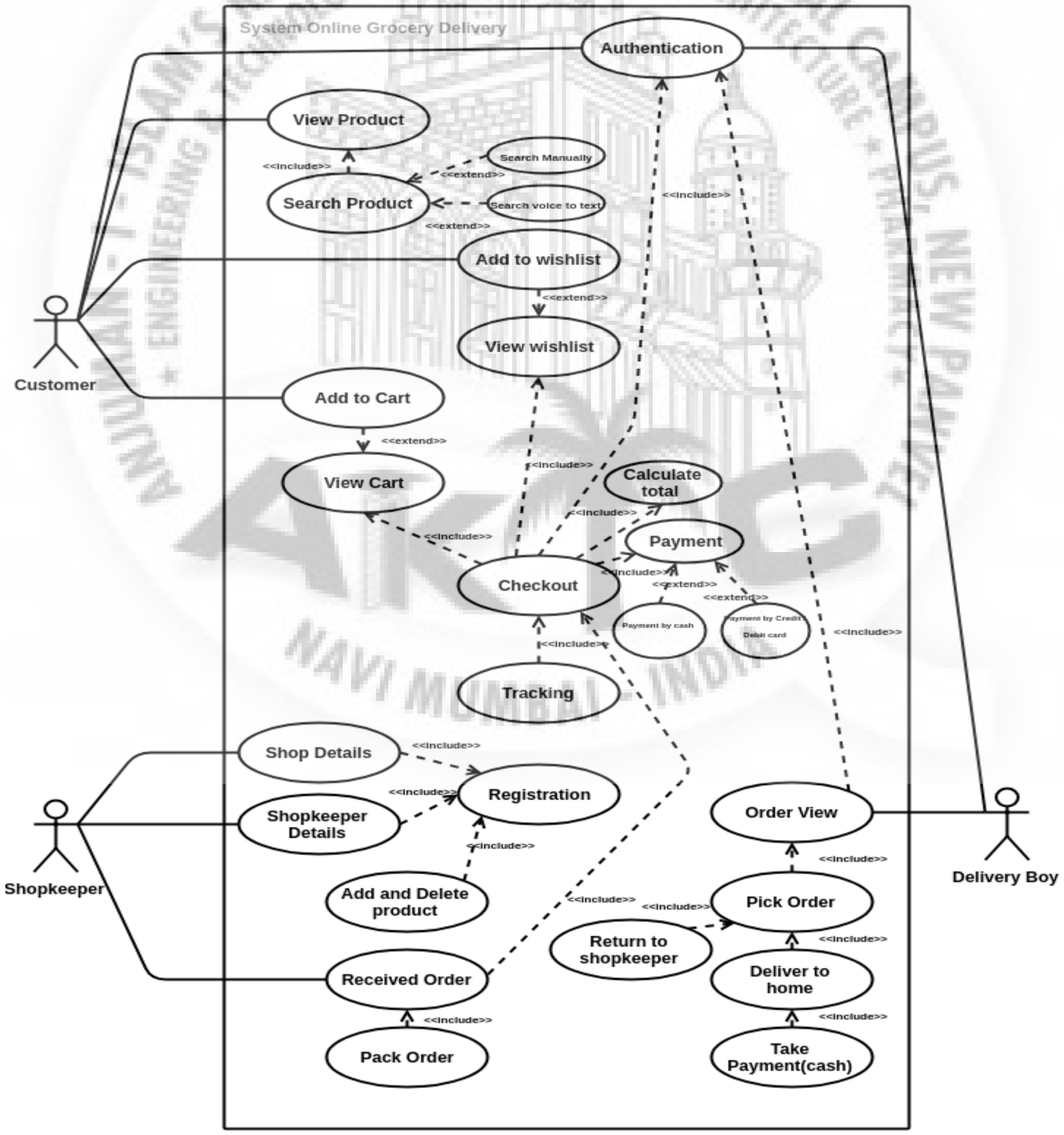


Figure 5.1: Usecase Diagram

Data-flow Diagram

DFD Level 0 : DFD Level 0 depicting main outcome of the system, Customer will buy the product,Shopkeeper will add the product,Shopkeeper sell and receive the product,Order Done by customer and Delivery Boy deliver product to customer.

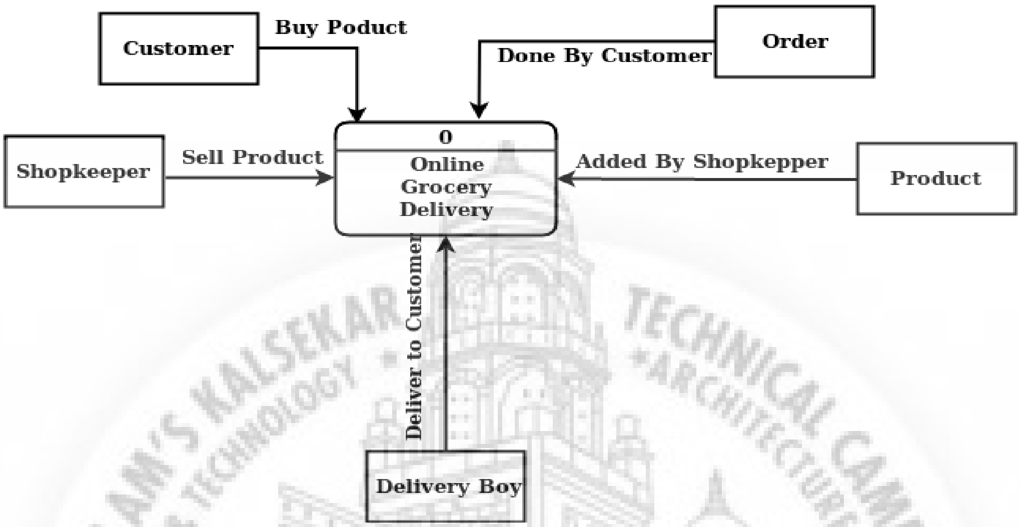


Figure 5.2: DFD Level 0 for Online Grocery Deliver System

DFD Level 1 : DFD Level 1 for Online Grocery Delivery showing their main process flow in the system via registration details being stored into Database.

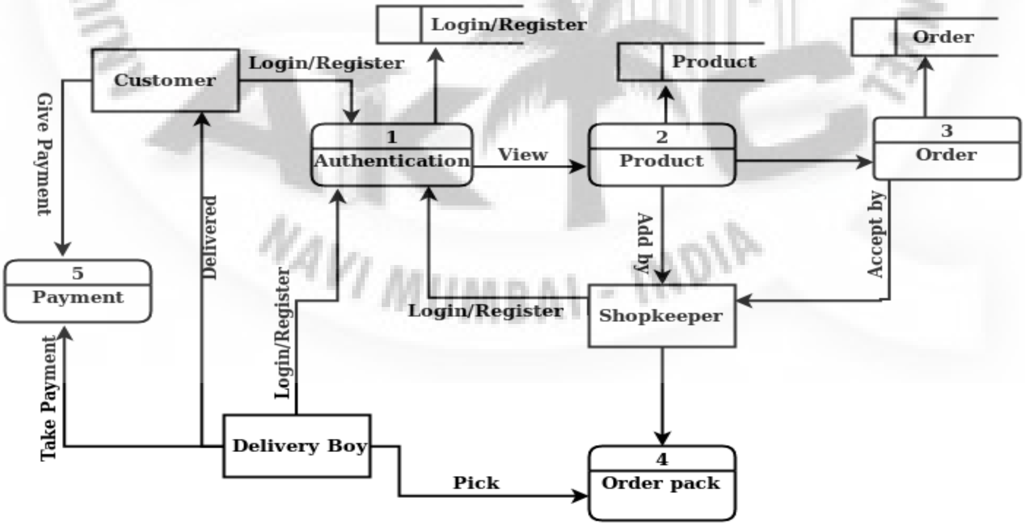


Figure 5.3: DFD Level 1 for Online Grocery Deliver System

DFD Level 2 : DFD Level 2 for mentors showing their detail login process flow in the system via authentication modules, ,from product searching to product buying flow is there in dfd level 2 ,shopkeeper will receive the product and pack the product,delivery boy deliver it to customer and take payment.

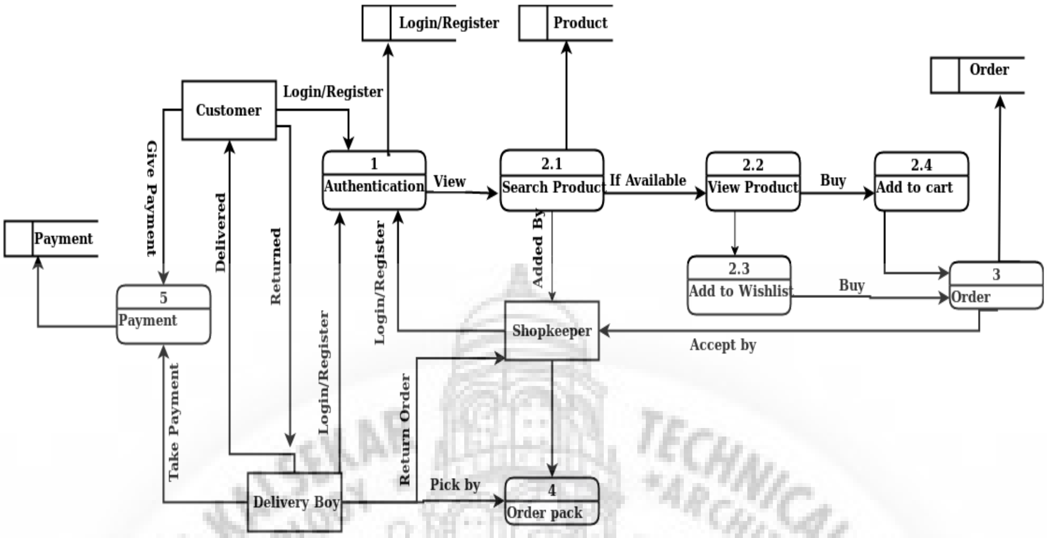


Figure 5.4: DFD Level 2 for Online Grocery Deliver System

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. Availability - Application will be available 24*7.

b. Usability - Application implementation is feasible using technologies that are accessible to the end-users.

c. Portability - The interfaces are compatible with Desktop.

d. Performance Efficiency -Application is able to perform well in a proper time constraint.

e. 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.

f. Time Efficiency - Time taken for the executing of system is less.

5.2 System Architecture Design

System architecture of our system gives the overview of the project.

- User can shop on the website and order for a product.

- User can search the product and User can also track the product status.
- Shopkeeper can accept user's request and pack the item for delivery.
- Shopkeeper can add and delete the product from their profiles.
- Delivery boy can pick the item from shopkeeper and deliver it to the respected customer.

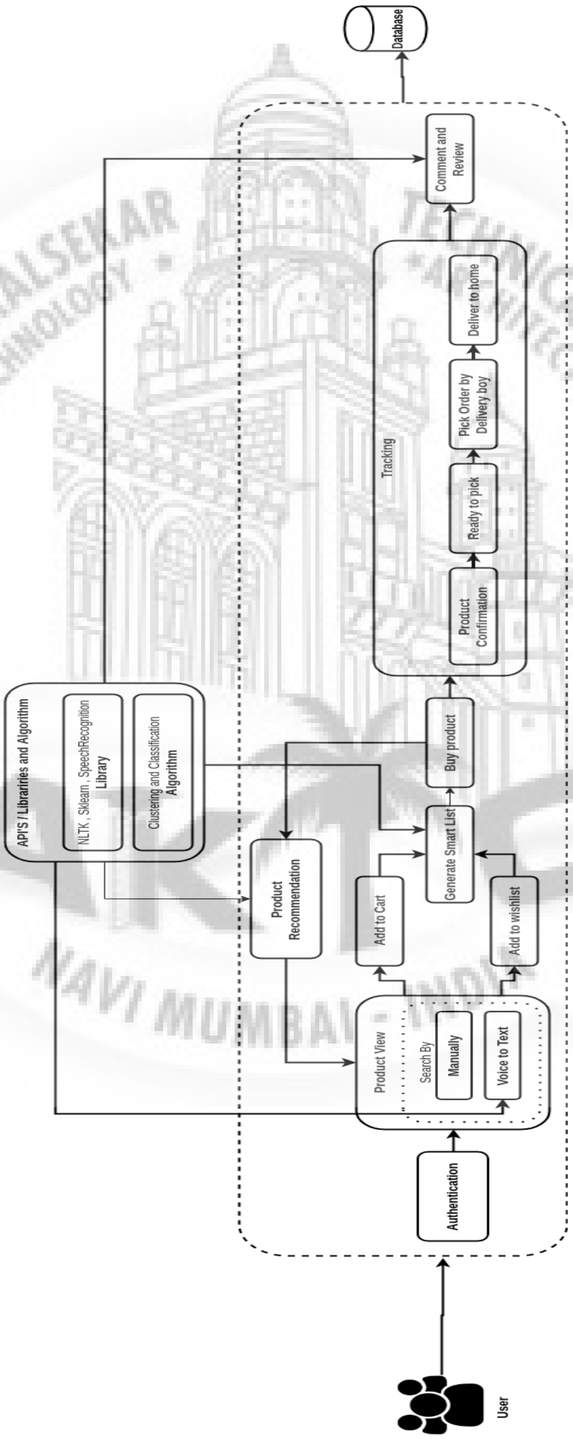


Figure 5.5: User view

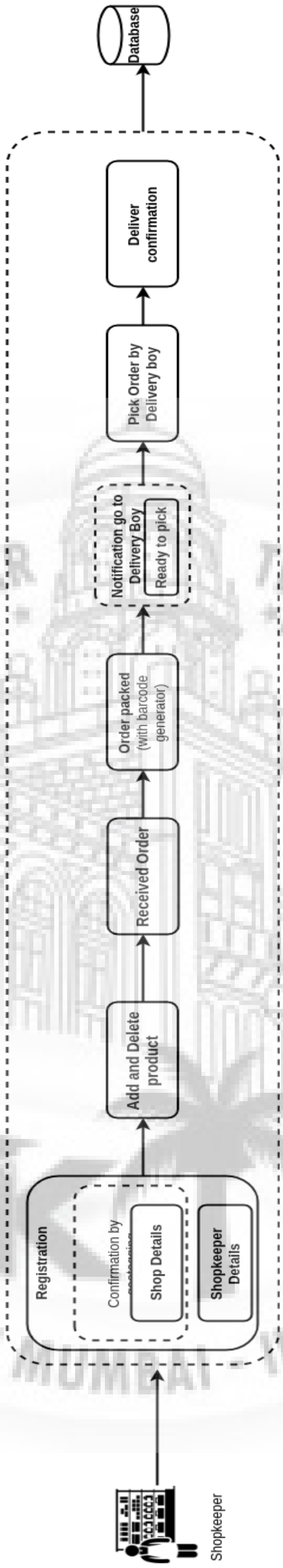


Figure 5.6: Shopkeeper view

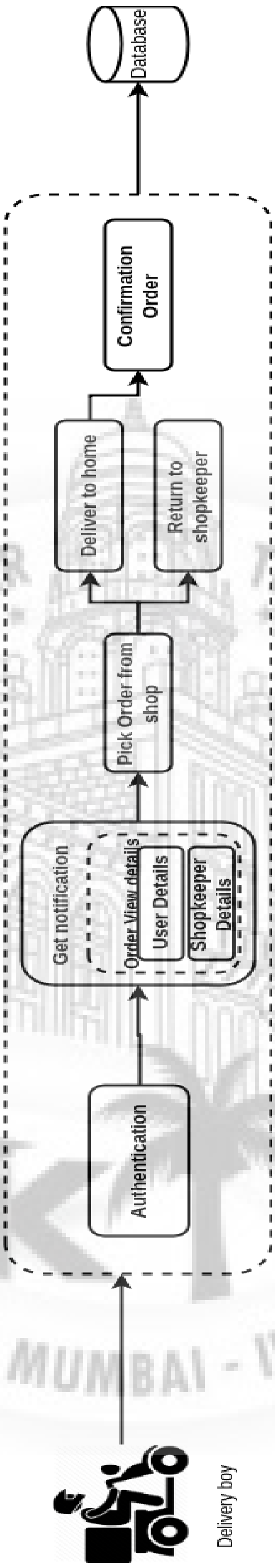


Figure 5.7: Delivery Boy view

5.3 Sub-system Development

Below Module describes how our system is working and tell what features are there in our system. Like Product recommendation, voice search, smart list generation, many more features. Our system consists of 3 modules:

5.3.1 User View

This is the first view of the system. This system is visible only to users. In this view, the user will be authenticated by the system. After that, the user can search for products manually or by their voice using voice to text option. As this system is online customers have a variety of options to buy the products. They can buy the product and track the delivery of their products. The system has features such as product recommendation for this it will track all the information of the user like what they search often, what they are buying frequently and what product they have added in the wish list and so on. Product recommendation tracks all user's history by applying an algorithm to give recommendations to users. A smart list is also generated by tracking the information of the user. Which types of products are users buying concurrently on that basis smart list will generated and recommended to users. This view includes all user activities from the beginning that is from the login till the end that is checkout and delivery tracking.

5.3.2 Shopkeeper View

This is the second view of the system. This system is visible only to the shopkeeper. The responsibilities of the shopkeeper include adding and deleting products in the system. The Shopkeeper and the location of his shop will be authenticated by the geotagging method. The shopkeeper will be asked to upload the picture of his shop and the location fetched from the image will be compared with the actual location.

The shopkeeper will always keep track of the products available in his Inventory. If a product is not available in the shopkeeper's inventory then the shopkeeper should make the product not available for users.

The shopkeeper is also responsible to confirm or reject the user's order. If the shopkeeper accepts the user order then he/she will pack the order for delivery. After the order is set to deliver then the shopkeeper will put the notification for the delivery boy to accept the order for delivery.

5.3.3 Delivery boy View

This is the third view of the system. This view is only available to the delivery boys. The responsibilities of the delivery boy include the delivery of the product to the right customer in the shortest time possible. The Delivery boy will also be authenticated by the system before being part of the system.

The delivery boy can see the notification of the orders available for delivery from their nearest shops. These orders are made available by the shopkeeper to the delivery boy. The delivery boy will accept the notification from the shopkeeper. He will pick the product from the respective shop and will deliver it to the customer. If the customer is not available at the time of delivery, the delivery boy will return the order to the shopkeeper.

5.3.4 Activity Diagram

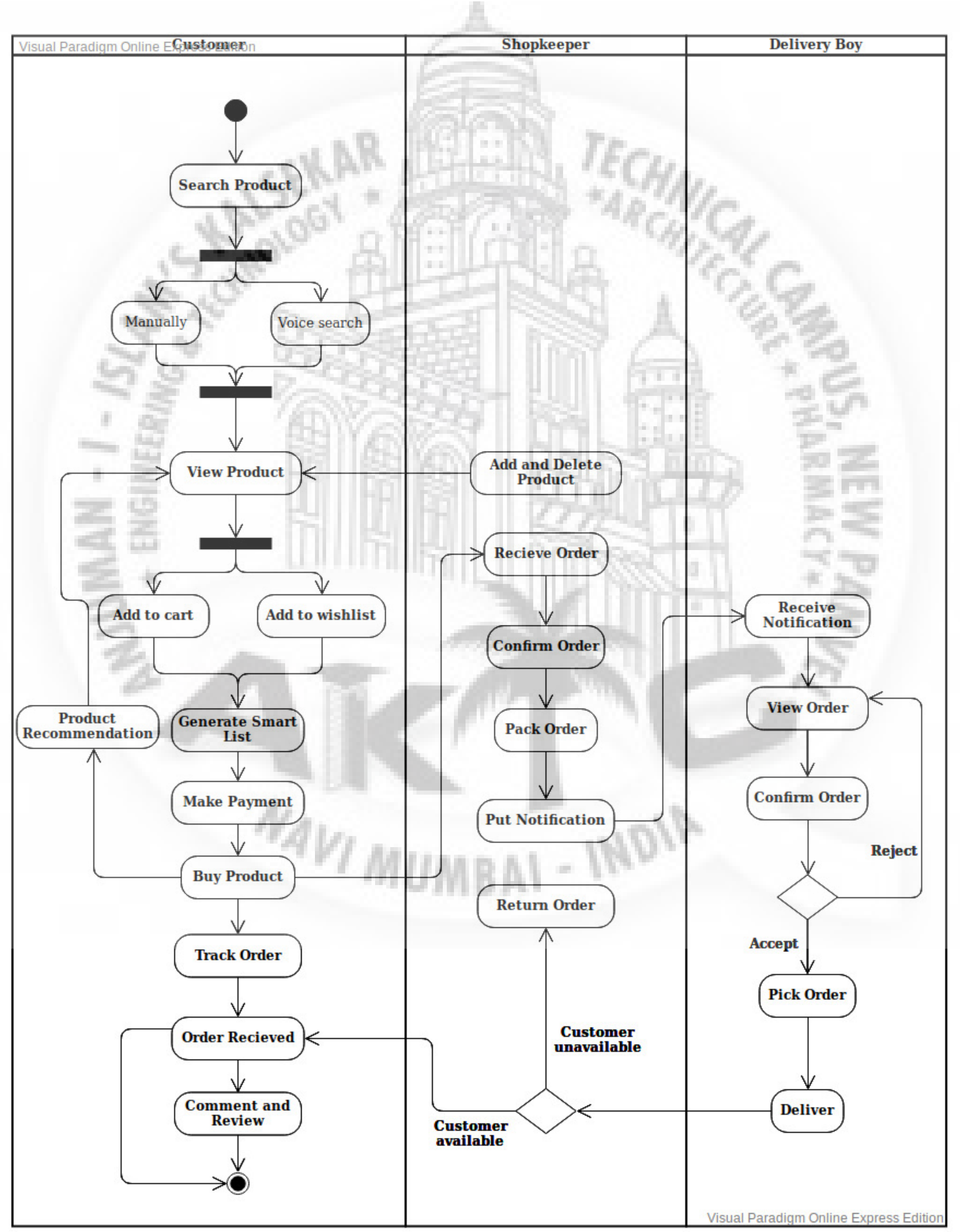


Figure 5.8: Activity Diagram for Online Grocery Delivery System

5.4 Systems Integration

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

5.4.1 Class Diagram

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing system's classes, their attributes, operations, and the relationship among objects. Class diagram showing each modules interconnection and relation between how one module is interacting with others.

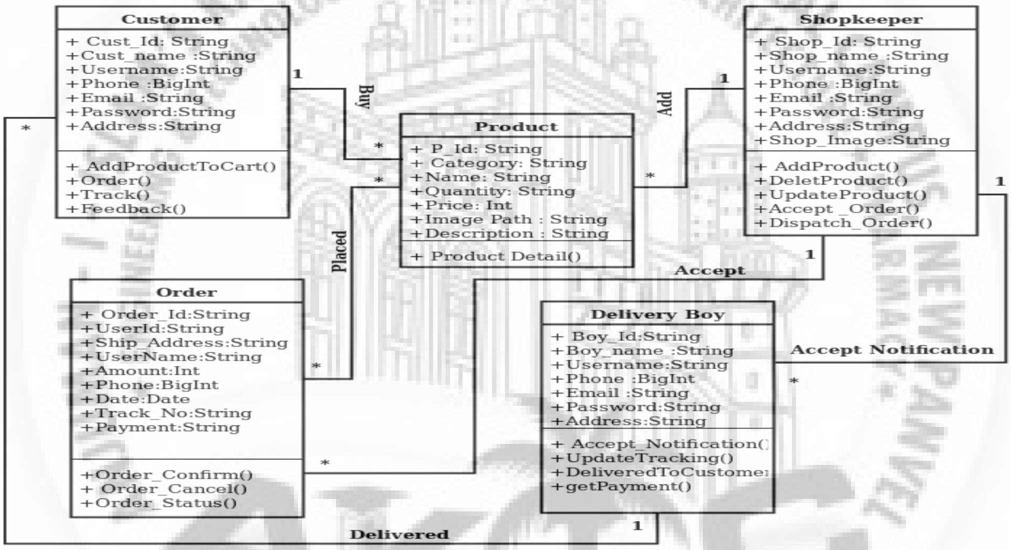


Figure 5.9: Class Diagram for Online Grocery Delivery

5.4.2 Sequence Diagram

A sequence diagram is an interaction diagram that shows how objects operates with one another and in what order. It is a construct of a message sequence chart. The following figure describe the sequence diagram for Online grocery delivery system. It shows the sequence of customer ,shopkeeper and delivery boy.

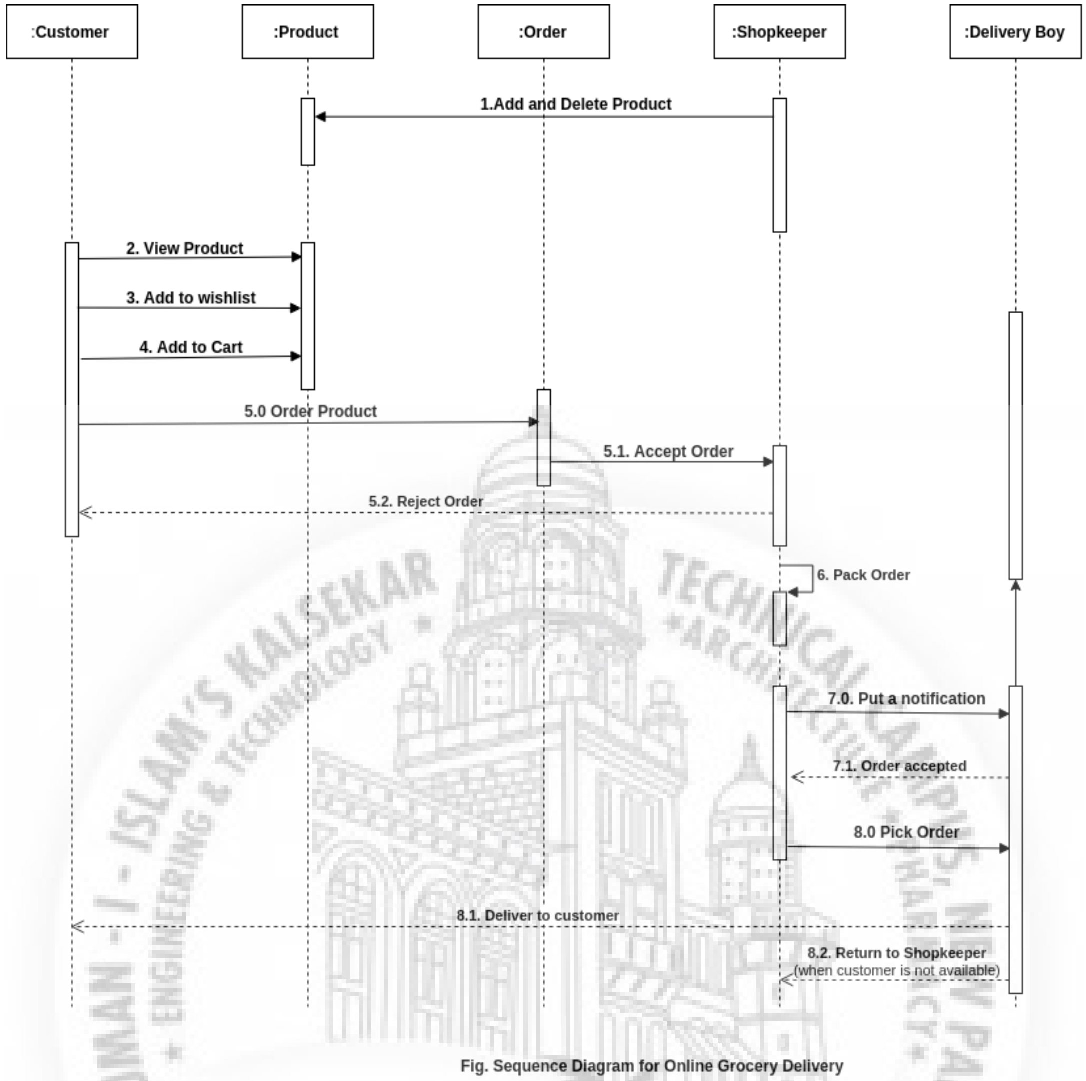


Figure 5.10: Sequence Diagram for Online Grocery Delivery

5.4.3 Component Diagram

Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.

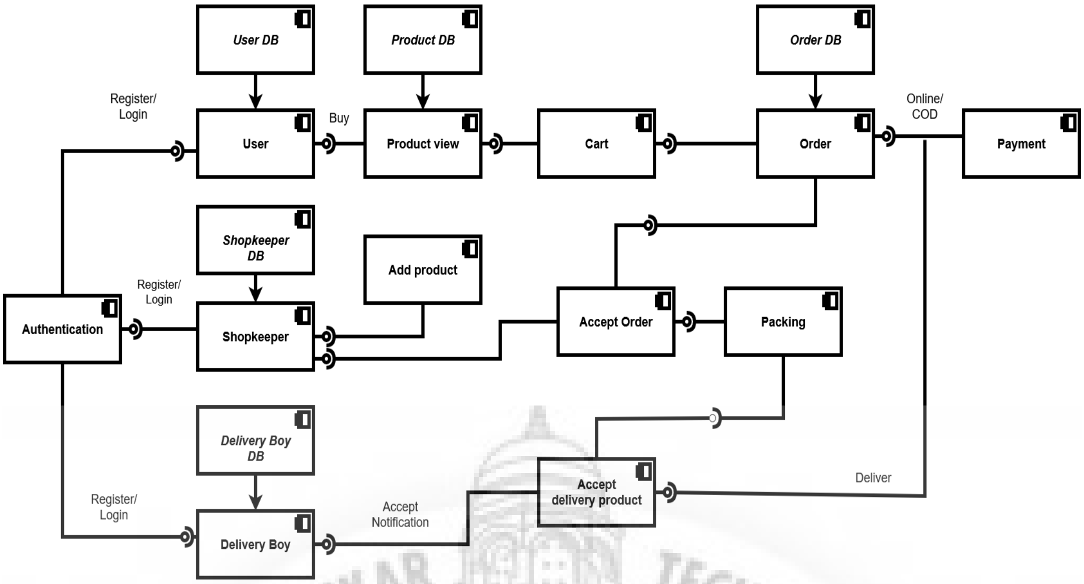


Figure 5.11: Component Diagram for Online Grocery Delivery

Chapter 6

Implementation

6.1 Module 1 - User View

This is the first view of the system. This system is visible only to users. In this view, the user will be authenticated by the system. Here the user can search for products manually or by their voice using voice to text option. As this system is online customers have a variety of options to buy the products.They can buy the product and track the delivery of their products.

The system has features such as product recommendation for this it will track all the information of the user like what they search often, what they are buying frequently and what product they have added in the wish list and so on. Product recommendation tracks all user's history by applying an algorithm to give recommendations to users. A smart list is also generated by tracking the information of the user. Which types of products are users buying concurrently on that basis smart list will generated and recommended to users. This view includes all user activities from the beginning that is from the login till the end that is checkout and delivery tracking.

- app.py

```
#home
@app.route(’/home’)
def home():
    return render_template(’home.html’)

#product_find Function
@app.route(’/product_find/<ty>’,methods=[’GET’, ’POST’])
def product_find(ty):
    form = RegistrationForm()
    if(session.get(’email’) is not None):
        session[’user’] = session[’email’]
        email = session[’user’]
        cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cur.execute(”SELECT uid from Users WHERE email=%s”,(email,))
        user = cur.fetchone()
        session[’uid’]=int(user[’uid’])
        print(ty)
        cur = mysql.connection.cursor()
        cur.execute(”select Product.Product_id ,Product.P_name,Product.
            P_actual_price ,Product.P_quantity ,Product.P_quantity_type ,Shopkeeper
            .shop_name ,Shopkeeper.shop_add ,Shopkeeper.shop_city ,Shopkeeper.
            shop_district ,Shopkeeper.shop_state from Product join Shopkeeper on
            Product.shop_id = Shopkeeper.shop_id where P_category= %s” ,(ty ,))
        data = cur.fetchall()
        if not data:
            flash(”Currenly Not avaiable !!”)
            return render_template(’product_find.html’)
        a=[]
        b=[]
        temp = []
        for da in data:
            if (da[1] not in temp):
                fruitname = da[1] + ’.jpeg’
                fpath = os.path.join(’/’,app.config[’UPLOAD_FOLDER’], fruitname)
                fpathhh = os.path.join(app.config[’UPLOAD_FOLDER’], fruitname)
                if not os.path.exists(fpathhh):
                    fruitname = da[1] + ’.jpg’
                    fpath = os.path.join(’/’,app.config[’UPLOAD_FOLDER’],
                        fruitname)
                a.append(fpath)
                a.append(da[0]) #id
                a.append(da[1]) #name
                a.append(da[2]) #actual price
                a.append(da[3]) #Available quantity
                a.append(da[4]) #weight/unit
                a.append(da[5]) #shopname
                a.append(da[6]) #shopadd
                a.append(da[7]) #shopcity
                a.append(da[8]) #shopDistrict
                a.append(da[9]) #shopstate
                b.append(a)
                a=[]
                print(fpath,da[0],da[1],da[2],da[3],da[4],da[5],da[6],da[7],da
                    [8],da[9])
                print(b)
                temp.append(da[1])
        cur.close()
        return render_template(’product_find.html’, data=b)
    else:
        flash(”You have to login first!!”)
        return redirect(url_for(’register’))

#Function for adding product in cart
@app.route(’/addHome/<name>/<shopname>’,methods=[’GET’, ’POST’])
def addHome(name=None,shopname=None):
    print(name,shopname)
    cursor = None
    if(request.method == ’POST’):
        cur = mysql.connection.cursor()
        cur.execute(”select Product_id from Product where P_name= %s”,(name,)) #
            Only for loop
        data = cur.fetchall()
        print(”Data”,data)
        for da in data:
            x=str(da[0])
            print(”XXX”,x)
            #print(”REQQQ”,request.form.get(”3”))
            #print(”REQQQ”,request.form.get(”7”))
            if(request.form.get(x) == None):
                pass
            else:
                print(da[0],request.form.get(x),type(request.form.get(x)))
                quantity = int(request.form[x])
                print("QUANTITY",quantity)
        db = MySQLdb.connect(host='localhost', db='grocery', user='ansari', passwd='ansari123')
        dict_cursor = db.cursor(MySQLdb.cursors.DictCursor)
        dict_cursor.execute("select Product.Product_id,Product.P_name,Product.P_actual_price,Product.P_quantity,Product.P_quantity_type,Shopkeeper.shop_name,Shopkeeper.shop_add,Shopkeeper.shop_city,Shopkeeper.shop_district,Shopkeeper.shop_state from Product join Shopkeeper on Product.shop_id = Shopkeeper.shop_id where P_name=%s and shop_name=%s",(name,shopname))
        row = dict_cursor.fetchone()
        print("ROW",row)
        itemArray = { str(row['Product_id']) : { 'name':row['P_name'], 'price':row['P_actual_price'], 'quantity': quantity,'quan_type':row['P_quantity_type'], 'total_price':quantity*row['P_actual_price'], 'shopname':row['shop_name'], 'shop_add':row['shop_add'],'shop_city':row['shop_city'],'shop_district':row['shop_district'], 'shop_state':row['shop_state'] }}
        print("Item array",itemArray)
        all_total_price = 0
        all_total_quantity = 0
        print("Session Bfore",session)
        session.modified  = True
        print("Session AFtre",session)
        if 'cart_item' in session:
            if(str(row['Product_id']) in session['cart_item']):
                print("IFFFF")
                for key,value in session['cart_item'].items():
                    if(str(row['Product_id']) == key):
                        print("InnerIFF")
                        old_quantity = session['cart_item'][key]['quantity']
                        total_quantity = old_quantity + quantity
                        session['cart_item'][key]['quantity'] = total_quantity
                        session['cart_item'][key]['total_price'] = total_quantity * row['P_actual_price']
                        print('total ::',total_quantity,old_quantity,quantity)
            else:
                print("elsee",session,"Session",session['cart_item'])
                session['cart_item'] = array_merge(session['cart_item'], itemArray)
                print("Session cart item ",session['cart_item'])

            for key, value in session['cart_item'].items():
                individual_quantity = int(session['cart_item'][key]['quantity'])
                individual_price = int(session['cart_item'][key]['total_price'])
                all_total_quantity = all_total_quantity + individual_quantity
                all_total_price = all_total_price + individual_price
                print("Forrr")
        else:
            session['cart_item'] = itemArray
            all_total_quantity = all_total_quantity + quantity
            all_total_price = all_total_price + quantity * row['P_actual_price']
        print("Before",session)
        session['all_total_quantity'] = all_total_quantity
        session['all_total_price'] = all_total_price
        print("LAst",session)
```

```python

        return render_template('home.html')
    else:
        print('Else')
        print(request.form)
        return 'Error while adding item to cart'
    return redirect(url_for('.home'))

#Checkout Page
@app.route('/checkout')
def checkout():
    return render_template('checkout.html')

#Order confirmation of users and data will go into cart_order and order_item
@app.route('/confirm',methods=['GET', 'POST'])
def confirm():
    if request.method == 'POST':
        uid = int(session['uid'])
        lane1 = request.form['lane1']
        lane2 = request.form['lane2']
        city = request.form['city']
        state = request.form['state']
        #For cart_order insert
        total_quantity = session['all_total_quantity']
        total_price = session['all_total_price']
        func()
        cur = mysql.connection.cursor()
        cur.execute("INSERT INTO cart_order(order_id,total_price,total_quantity,lane1,lane2,city,state,uid) VALUES(%s,%s,%s,%s,%s,%s,%s,%s)",(id(x),total_price,total_quantity,lane1,lane2,city,state,uid))
        #Inserting data into order_item
        for key, value in session['cart_item'].items():
            Product_id = int(key)
            name = session['cart_item'][key]['name']
            price = int(session['cart_item'][key]['price'])
            individual_quantity = int(session['cart_item'][key]['quantity'])
            quantity_type = session['cart_item'][key]['quan_type']
            individual_price = int(session['cart_item'][key]['total_price'])
            shop_name = session['cart_item'][key]['shopname']
            cur = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
            cur.execute("SELECT shop_id from Shopkeeper WHERE shop_name=%s",(shop_name,)) #fetching shop_id for inserting it in order_item (foreign key)
            user = cur.fetchone()
            cur.execute("SELECT P_quantity from Product WHERE Product_id=%s",(Product_id,))
            quant = cur.fetchone()
            Find_quantity = quant['P_quantity'] - individual_quantity   #Finding decreses product quantity
            cur = mysql.connection.cursor()
            cur.execute("update Product set P_quantity =%s where Product_id=%s",(abs(Find_quantity),Product_id))
            cur.execute("INSERT INTO order_item(Product_id,P_name,P_price,P_quantity,P_quantity_type,P_total_price,order_id,shop_id,status_value) VALUES(%s,%s,%s,%s,%s,%s,%s,%s,0)",(Product_id,name,price,individual_quantity,quantity_type, individual_price,id(x),user['shop_id']))
            mysql.connection.commit()
            cur.close()
        session.pop('all_total_quantity',None)
        session.pop('all_total_price',None)
        session.pop('cart_item', None)
    return redirect(url_for('.track'))

#Tracking of product
@app.route('/track',methods=['GET', 'POST'])
def track():
    uid = int(session['uid'])
    cur = mysql.connection.cursor()
    cur.execute("select order_item.P_name , order_item.P_price , order_item.P_quantity , order_item.P_total_price ,order_item.order_id , cart_order.total_price , cart_order.total_quantity , cart_order.lane1 , cart_order.lane2 , cart_order.city , cart_order.state , cart_order.uid , order_item.status_value ,Shopkeeper.shop_name , Shopkeeper.shopkeeper_contact , Shopkeeper.shop_add , Shopkeeper.shop_city , order_item.cart_item_id from order_item , cart_order , Shopkeeper where order_item.order_id = cart_order.order_id and order_item.shop_id = Shopkeeper.shop_id and cart_order.uid=%s",(uid,))
    data = cur.fetchall()
    temp_orderid = {} #storing order_id with product id
    temp_shop_product1 = {} #storing product details of shop
    for da in data:
        key = da[4]
        if(da[4] not in temp_orderid):
            print("YEEE>>>>>>>>>>> ",da[4])
            temp_orderid.setdefault(key,[])
            temp_orderid[key].append(da[17])
            key_shop = da[17]
            temp_shop_product1.setdefault(key_shop,[])
            listA = []
            listA.append(da[0])
            listA.append(da[1])
            listA.append(da[2])
            listA.append(da[3])
            listA.append(da[4])
            listA.append(da[5])
            listA.append(da[6])
            listA.append(da[7])
            listA.append(da[8])
            listA.append(da[9])
            listA.append(da[10])
            listA.append(da[11])
            listA.append(da[12])
            listA.append(da[13])
            listA.append(da[14])
            listA.append(da[15])
            listA.append(da[16])
            listA.append(da[17])
            temp_shop_product1[key_shop] = listA
        else:
            temp_orderid[key].append(da[17])
            key_shop = da[17]
            temp_shop_product1.setdefault(key_shop,[])
            listA = []
            listA.append(da[0])
            listA.append(da[1])
            listA.append(da[2])
            listA.append(da[3])
            listA.append(da[4])
            listA.append(da[5])
            listA.append(da[6])
            listA.append(da[7])
            listA.append(da[8])
            listA.append(da[9])
            listA.append(da[10])
            listA.append(da[11])
            listA.append(da[12])
            listA.append(da[13])
            listA.append(da[14])
            listA.append(da[15])
            listA.append(da[16])
            listA.append(da[17])
            temp_shop_product1[key_shop] = listA
    return render_template('track.html',odrer_id=temp_orderid,product1=temp_shop_product1)

#For cart empty
@app.route('/empty_cart')
def empty_cart():
    try:
        session.pop('all_total_quantity',None)
        session.pop('all_total_price',None)
        session.pop('cart_item', None)
        return redirect(url_for('.home'))
    except Exception as e:
        print(e)

#For delete product from cart
@app.route('/delete_product/<string:name>')
def delete_product(name):
    try:
        all_total_price = 0
        all_total_quantity = 0
        session.modified = True
        for item in session['cart_item'].items():
            print(item[0],name)
            if item[0] == name:
                session['cart_item'].pop(item[0], None)
                if 'cart_item' in session:
                    for key, value in session['cart_item'].items():
                        individual_quantity = int(session['cart_item'][key]['quantity'])
                        individual_price = int(session['cart_item'][key]['total_price'])
                        all_total_quantity = all_total_quantity + individual_quantity
                        all_total_price = all_total_price + individual_price
                break
        if all_total_quantity == 0:
            session.clear()
        else:
            session['all_total_quantity'] = all_total_quantity
            session['all_total_price'] = all_total_price
        return redirect(url_for('.home'))
    except Exception as e:
        print(e)
```

6.2 Module 2 - Shopkeeper View

This is the second view of the system. This system is visible only to the shopkeeper. The responsibilities of the shopkeeper include adding and deleting products in the system. The Shopkeeper and the location of his shop will be authenticated by the geotagging method. The shopkeeper will be asked to upload the picture of his shop and the location fetched from the image will be compared with the actual location.

The shopkeeper will always keep track of the products available in his Inventory. If a product is not available in the shopkeeper's inventory then the shopkeeper should make the product not available for users. The shopkeeper is also responsible to confirm or reject the user's order. If the shopkeeper accepts the user order then he/she will pack the order for delivery. After the order is set to deliver then the shopkeeper will put the notification for the delivery boy to accept the order for delivery.

- app.py

```

#Showing Add product page to Shopkeeper
@app.route('/index')
def index():
    form = ShopkeeperForm()
    if(session.get('id') is not None):
        session['shopk'] = session['id']
        cur = mysql.connection.cursor()
        cur.execute("SELECT * FROM Product where shop_id=%s",(session['shopk'],))
        data = cur.fetchall()
        cur.close()
        return render_template('adding.html', prod=data )
    else:
        flash("You have to login first!!")
        return render_template('regshop.html', title='Shopkeeper Reg', form=form)

#Adding Product in Shopkeeper Account
@app.route('/insert', methods = ['POST'])
def insert():
    if request.method == "POST":
        name = request.form['name']
        price = request.form['price']
        available = request.form['available']
        category = request.form['category']
        weight = request.form['weight']
        manufacture = request.form['manufacture']
        expiry = request.form['expiry']
        file = request.files['ifile']
        cur = mysql.connection.cursor()
        cur.execute("INSERT INTO Product (P_name, P_actual_price,P_category,P_quantity,P_quantity_type,shop_id,manu_date,exp_date,image) VALUES (%s, %s, %s,%s,%s,%s,%s,%s,%s)", (name, price,category, available,weight,session['shopk'],manufacture,expiry,file.filename))
        mysql.connection.commit()
        file = request.files['ifile']
        # if user does not select file, browser also
        # submit an empty part without filename
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
              file.save(os.path.join(app.config['UPLOAD_FOLDER'],name+'.jpg'))
            if(file.filename == False):
                file.save(os.path.join(app.config['UPLOAD_FOLDER'],name+'.jpeg'))
        #os.rename(UPLOAD_FOLDER +filename ,UPLOAD_FOLDER +'bear.jpg')
        return redirect(url_for('index'))

#Delete Product from Shopkeer Account
@app.route('/delete/<string:id_data>', methods = ['GET'])
def delete(id_data):
    flash("Record Has Been Deleted Successfully")
    cur = mysql.connection.cursor()
    cur.execute("DELETE FROM Product WHERE product_id=%s", (id_data,))
    mysql.connection.commit()
    return redirect(url_for('index'))

#update Product from Shopkeeper Account
@app.route('/update',methods=['POST','GET'])
def update():
    if request.method == 'POST':
        id_data = request.form['id']
        name = request.form['name']
        price = request.form['price']
        category = request.form['category']
        available = request.form['available']
        weight = request.form['weight']
        manufacture = request.form['manufacture']
        expiry = request.form['expiry']
        file = request.files['imgfile']
        cur = mysql.connection.cursor()
        cur.execute("UPDATE Product SET P_name=%s, P_actual_price=%s,P_category=%s, P_quantity=%s,P_quantity_type=%s,shop_id=%s,manu_date=%s,exp_date=%s WHERE Product_id=%s",(name, price,category, available, weight,session['shopk'],manufacture,expiry, id_data))
        flash("Data Updated Successfully")
        mysql.connection.commit()
        file = request.files['imgfile']
        # if user does not select file, browser also
        # submit an empty part without filename
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'],name+'.jpg'))
        return redirect(url_for('index'))

#Shopkeeper will get Order
@app.route('/shopgetorder',methods=['GET', 'POST'])
def shopgetorder():
    form = ShopkeeperForm()
    if(session.get('id') is not None):
        cur = mysql.connection.cursor()
        cur.execute("select  order_item.cart_item_id ,order_item.P_name , order_item.P_price , order_item.P_quantity , order_item.P_quantity_type ,  order_item.P_total_price ,order_item.order_id , cart_order.total_price , cart_order.total_quantity , cart_order.lane1 , cart_order.lane2 , cart_order.city , cart_order.state , cart_order.uid , order_item.status_value  from order_item , cart_order , Shopkeeper where order_item.order_id = cart_order.order_id and order_item.shop_id = Shopkeeper.shop_id and order_item.shop_id=%s",(int(session['id']),))
        order = cur.fetchall()
        return render_template('shop_get_order.html',data=order)
    else:
        flash("You have to login first!!")
        return render_template('regshop.html', title='Shopkeeper Reg', form=form)

#Accepting order of Shopkeeper
@app.route('/accept_Order/<product_order_id>',methods=['GET','POST'])
def accept_Order(product_order_id):
    temp = 0
    cur = mysql.connection.cursor()
    cur.execute("select status_value  from order_item where order_item.cart_item_id=%s",(int(product_order_id),))
    status_value = cur.fetchone()
    temp = status_value[0] + 1
    cur.execute("update order_item set status_value =%s where cart_item_id = %s",(temp,int(product_order_id)))
    cur.execute("select  order_item.cart_item_id ,order_item.P_name , order_item.P_price , order_item.P_quantity , order_item.P_quantity_type , order_item.P_total_price ,order_item.order_id , cart_order.total_price , cart_order.total_quantity , cart_order.lane1 , cart_order.lane2 , cart_order.city , cart_order.state , cart_order.uid , order_item.status_value  from order_item , cart_order , Shopkeeper where order_item.order_id = cart_order.order_id and order_item.shop_id = Shopkeeper.shop_id and order_item.shop_id=%s",(int(session['shopk']),))
    order = cur.fetchall()
    mysql.connection.commit()
    cur.close()
    return render_template('shop_get_order.html',data=order)
```

6.3 Module 3 - Delivery Boy View

This is the third view of the system. This view is only available to the delivery boys. The responsibilities of the delivery boy include the delivery of the product to the right customer in the shortest time possible. The Delivery boy will also be authenticated by the system before being part of the system.The delivery boy can see the notification of the orders available for delivery from their nearest shops.

These orders are made available by the shopkeeper to the delivery boy. The delivery boy will accept the notification from the shopkeeper. He will pick the product from the respective shop and will deliver it to the customer. If the customer is not available at the time of delivery, the delivery boy will return the order to the shopkeeper.

- app.py

```
#Delivery Boy will get Order packed
@app.route('/Db_home',methods=['GET','POST'])
def Db_home():
    form = DBoyForm()
    if(session.get('db_id') is not None):
        cur = mysql.connection.cursor()
        cur.execute("select  order_item.cart_item_id , order_item.order_id ,
            cart_order.lane1 , cart_order.lane2 , cart_order.city , cart_order.
            state , cart_order.uid , order_item.status_value , Shopkeeper.
            shop_name , Shopkeeper.shopkeeper_contact , Shopkeeper.
            shopkeeper_name , Shopkeeper.shop_add , Shopkeeper.shop_city ,
            Shopkeeper.shop_district , Shopkeeper.shop_state  from order_item ,
             cart_order , Shopkeeper where order_item.order_id = cart_order.
            order_id and order_item.shop_id = Shopkeeper.shop_id and order_item.
            status_value=2")
        orderPacked = cur.fetchall()
        return render_template('Db_order_packed.html',data=orderPacked)
    else:
        flash("You have to login first!!")
        return render_template('regDBoy.html', title='Delivery Boy Reg', form=
            form)
    #return render_template('Db_home.html')

@app.route('/accept_Order_pick/<product_order_id>',methods=['GET','POST'])
def accept_Order_pick(product_order_id):
    form = DBoyForm()
    if(session.get('db_id') is not None):
        temp = 0
        cur = mysql.connection.cursor()
        cur.execute("select status_value  from order_item where order_item.
            cart_item_id=%s",(int(product_order_id),))
        status_value = cur.fetchone()
        if(status_value[0] == 2):
            temp = status_value[0] + 1
            cur.execute("update order_item set status_value =%s , db_id = %s
                where cart_item_id=%s",(temp,int(session['db_id']),int(
                product_order_id)))
            flash("Order Accepted for pick. Select Order Pick option from Menu")
        cur.execute(”select  order_item.cart_item_id , order_item.order_id ,
          cart_order.lane1 , cart_order.lane2 , cart_order.city , cart_order.
          state , cart_order.uid , order_item.status_value , Shopkeeper.
          shop_name , Shopkeeper.shopkeeper_contact , Shopkeeper.
          shopkeeper_name , Shopkeeper.shop_add , Shopkeeper.shop_city ,
          Shopkeeper.shop_district , Shopkeeper.shop_state  from order_item ,
           cart_order , Shopkeeper where order_item.order_id = cart_order.
          order_id and order_item.shop_id = Shopkeeper.shop_id and order_item.
          status_value=2”)
        orderPacked = cur.fetchall()
        mysql.connection.commit()
        cur.close()
        return render_template('Db_order_packed.html',data=orderPacked)
    else:
        flash(”You have to login first!!”)
        return render_template('regDBoy.html', title='Delivery Boy Reg', form=
          form)

@app.route('/Order_pick',methods=['GET','POST'])
def Order_pick():
    form = DBoyForm()
    if(session.get('db_id') is not None):
        flash(”Go to shop and pick order. Order is ready to pick. Deliver it to
          User”)
        cur = mysql.connection.cursor()
        cur.execute(”select  order_item.cart_item_id , order_item.order_id ,
          cart_order.lane1 , cart_order.lane2 , cart_order.city , cart_order.
          state , cart_order.uid , order_item.status_value , order_item.db_id
          , Shopkeeper.shop_name , Shopkeeper.shopkeeper_contact , Shopkeeper.
          shopkeeper_name , Shopkeeper.shop_add , Shopkeeper.shop_city ,
          Shopkeeper.shop_district , Shopkeeper.shop_state , Users.firstname
            from order_item , cart_order , Shopkeeper , Users where order_item
          .order_id = cart_order.order_id and order_item.shop_id = Shopkeeper.
          shop_id and cart_order.uid = Users.uid and order_item.status_value=3
           and order_item.db_id=%s”,(int(session['db_id']),))
        orderPick = cur.fetchall()
        mysql.connection.commit()
        cur.close()
        return render_template('Order_pick.html',data=orderPick)
    else:
        flash(”You have to login first!!”)
        return render_template('regDBoy.html', title='Delivery Boy Reg', form=
          form)

#OrderDelivered
@app.route('/Order_delivered/<product_order_id>',methods=['GET','POST'])
def Order_delivered(product_order_id):
    form = DBoyForm()
    if(session.get('db_id') is not None):
        temp = 0
        cur = mysql.connection.cursor()
        cur.execute(”select status_value  from order_item where order_item.
          cart_item_id=%s”,(int(product_order_id),))
        status_value = cur.fetchone()
        if(status_value[0] == 3):
            temp = status_value[0] + 1
            cur.execute(”update order_item set status_value =%s , db_id = %s
              where cart_item_id=%s”,(temp,int(session['db_id']),int(
              product_order_id)))
            flash(”Order is delivered. Select Order Delivered option from Menu!”
              )
        cur.execute("select  order_item.cart_item_id , order_item.order_id , cart_order.lane1 , cart_order.lane2 , cart_order.city , cart_order.state , cart_order.uid , order_item.status_value , order_item.db_id , Shopkeeper.shop_name , Shopkeeper.shopkeeper_contact , Shopkeeper.shopkeeper_name , Shopkeeper.shop_add , Shopkeeper.shop_city , Shopkeeper.shop_district , Shopkeeper.shop_state , Users.firstname  from order_item , cart_order , Shopkeeper , Users where order_item.order_id = cart_order.order_id and order_item.shop_id = Shopkeeper.shop_id and cart_order.uid = Users.uid and order_item.status_value=3  and order_item.db_id=%s",(int(session['db_id']),))
        orderPacked = cur.fetchall()
        mysql.connection.commit()
        cur.close()
        return render_template('Order_pick.html',data=orderPacked)
    else:
        flash("You have to login first!!")
        return render_template('regDBoy.html', title='Delivery Boy Reg', form=form)

@app.route('/Order_confirm',methods=['GET','POST'])
def Order_confirm():
    form = DBoyForm()
    if(session.get('db_id') is not None):
        flash("Successfully Deliver it to User")
        cur = mysql.connection.cursor()
        cur.execute("select  order_item.cart_item_id , order_item.order_id , cart_order.lane1 , cart_order.lane2 , cart_order.city , cart_order.state , cart_order.uid , order_item.status_value , order_item.db_id , Shopkeeper.shop_name , Shopkeeper.shopkeeper_contact , Shopkeeper.shopkeeper_name , Shopkeeper.shop_add , Shopkeeper.shop_city , Shopkeeper.shop_district , Shopkeeper.shop_state , Users.firstname  from order_item , cart_order , Shopkeeper , Users where order_item.order_id = cart_order.order_id and order_item.shop_id = Shopkeeper.shop_id and cart_order.uid = Users.uid and order_item.status_value=4  and order_item.db_id=%s",(int(session['db_id']),))
        orderPick = cur.fetchall()
        mysql.connection.commit()
        cur.close()
        return render_template('Order_pick.html',data=orderPick)
    else:
        flash("You have to login first!!")
        return render_template('regDBoy.html', title='Delivery Boy Reg', form=form)
```

Chapter 7

System Testing

System Testing is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Below shows the test cases of our system.

7.1 Test Cases and Test Results

Test ID	Test Case Title	Test Condition	System Behavior	Expected Result
T01	User Registration	All Valid Input	User Registered Successfully/ Unsuccessfully	Success/ Failed
T02	User Login	Username and Password Required	User login Successfully/ Unsuccessfully	Success/ Failed
T03	Add Item in cart	Data fetch and add into cart	Show items in Cart successfully	Successfully add item in cart
T04	Delete Item from cart	Date fetch and delete from cart	Show items in Cart removed successfully	Successfully delete item in cart
T05	Search Item	Date fetch and search	Show searched items successfully/Unsuccessfully	Successfully show search item
T06	Checkout	Input delivery address	Successfully confirmed order from user side	Successfully confirm order
T07	Track	Data Fetch and show all product order	Successfully show all ordered product	Successful track page

T08	Shopkeeper Registration	All Valid Input	Shopkeeper Registered Successfully/ Unsuccessfully	Success/ Failed
T09	Shopkeeper Login	Username and Password Required	Shopkeeper login Successfully/ Unsuccessfully	Success/ Failed
T10	Shopkeeper Add Product	Valid Input Required	Add Product to shop Successfully	Success
T11	Shopkeeper Delete Product	Data fetch and delete	Delete Product from shop Successfully	Success
T12	Shopkeeper Edit Product	Data fetch and Edit	Edit Product Successfully	Success
T13	Orders Product	Show all Ordered product to shopkeeper	Accept Delete Product	Accept/Reject
T14	Orders Product	Show all Ordered product to shopkeeper	Accept/ Packed/ Reject Product	Accept/ Packed/ Reject
T15	Delivery Boy Registration	All Valid Input	Delivery Boy Registered Successfully/ Unsuccessfully	Success/ Failed
T16	Delivery Boy Login	Username and Password Required	Delivery Boy login Successfully/ Unsuccessfully	Success/ Failed
T17	Delivery Boy Accept Order	Data Fetch and Accept	Order accepted by Delivery Boy Successfully/ Unsuccessfully	Accept
T18	Delivery Boy Pick Order	Data Fetch and Pick	Order picked by Delivery Boy Successfully/ Unsuccessfully	Picked
T19	Delivery Boy Delivered Order	Data Fetch and Delivered	Order delivered by Delivery Boy Successfully/ Unsuccessfully	Successfully Delivered

7.2 Test Cases

Title: User registration – Successfully register a new user.
Description: A new user should be able to successfully register themselves.

Precondition: The user has given valid credentials.
Assumption: a supported browser is being used.

Test Steps:

1. Click 'Sign Up' button.
2. Enter valid credentials in the field.
3. Enter OTP generate on mobile.
4. Click 'Register' button.

Expected Result: User should be successfully registered on the website.
Actual Result: User is successfully registered.

Title: User login – Successful login in website.
Description: A registered user should be able to successfully login in website.

Precondition: The User is pre-registered.
Assumption: a supported browser is being used.

Test Steps:

1. Click 'SignIn' option.
2. Enter gmail id.
3. Enter password.
4. Click 'Log In' button.

Expected Result: User should be successfully logged in and redirected to home page .

Actual Result: User is redirected to home page.

Title: Add Item in cart – Successfully add an item in cart.
Description: A registered user should be able to click on view cart and delete item from cat.

Precondition: The user must be logged in with their registered details.

Assumption: a supported browser is being used.

Test Steps:

1. Find the item to be added.
2. Click on Add to cart option.
3. Enter the quantity of product.
4. Click on View cart option.

Expected Result: An item is should be successfully added in cart list.
Actual Result: The item is successfully added and is available in cart.

Title: Delete Item From Cart – Successfully remove an item from cart
Description: User should be able to remove the selected from cart.

Precondition: One or more Item must be present in cart.

Assumption: a supported browser is being used.

Test Steps:

1. Click on View Cart.
2. Select the Item to delete
3. Click the Delete option.

Expected Result: An Item should be removed successfully from the cart.
Actual Result: Selected Item is removed from cart.

Title: Search Item – Successfully show the searched item.
Description: A registered user should be able to search an item and view it.

Precondition: The product name must be written correctly for search.

Assumption: a supported browser is being used.

Test Steps:

1. Click on search bar.
2. Type the product to search.
3. Click Enter or search.

Expected Result: The searched item should be displayed to user.
Actual Result: Searched item is displayed successfully.

Title: Checkout – Successfully confirm the user's order.
Description: A registered user should be able confirm the order and checkout successfully.

Precondition: The user has selected an Item to buy.

Assumption: a supported browser is being used.

Test Steps:

1. Navigate to 'View cart' page.
2. Click on 'Checkout' button.
3. Enter the address.
4. Confirm the Order.

Expected Result: User should be able to confirm an order successfully.
Actual Result: An product is ordered successfully.

Title: Track – Successfully track the order status.
Description: A registered user should be able to successfully track their ordered product.

Precondition: The user has ordered a product.

Assumption: a supported browser is being used.

Test Steps:

1. Navigate to 'Track Order' page
2. Click on track product.
3. Enter product id.
4. See your product status.

Expected Result: User should be able to track the product they ordered.
Actual Result: User can track the ordered product.

Title: Shopkeeper registration – Successfully register a new shopkeeper.
Description: A new user should be able to successfully register themselves as shopkeeper.

Precondition: The user has given valid credentials.

Assumption: a supported browser is being used.

Test Steps:

1. Click 'Sign Up' button.
2. Enter valid credentials in the field.
3. Enter OTP generate on mobile.
4. Click 'Register' button.

Expected Result: Shopkeeper should be successfully registered on the website.
Actual Result: Shopkeeper is successfully registered.

Title: Shopkeeper login – Successful login in website.
Description: A registered user should be able to successfully login in website.

Precondition: The shopkeeper is successfully registered.
Assumption: a supported browser is being used.

Test Steps:

1. Click ’SignIn’ option.
2. Enter gmail id.
3. Enter password.
4. Click ’Log In’ button.

Expected Result: Shopkeeper should be successfully logged in and redirected to dashboard page .
Actual Result: Shopkeeper can see it’s dashboard page.

Title: Shopkeeper Add Product – Successfully add a product to shop.
Description: A registered shopkeeper should be able to successfully add a product in shop.

Precondition: The shopkeeper has logged in.
Assumption: a supported browser is being used.

Test Steps:

1. Click ’Add Product Data’ button.
2. Enter product details.
3. Click ’Insert Data’ button.

Expected Result: Shopkeeper should be able to see the product they added.
Actual Result: Shopkeeper can see the added product.

Title: Shopkeeper Edit Product – Successfully edit product details.
Description: A registered shopkeeper should be able to successfully update details of any product.

Precondition: The user has added product in the list.
Assumption: a supported browser is being used.

Test Steps:

1. Click 'Edit' button.
2. Enter the details to change.
3. Click 'Update' button.

Expected Result: Shopkeeper should be able to see the edited details.
Actual Result: Shopkeeper can see the updated details.

Title: Ordered Products – Successfully accept pack or reject a order.
Description: A registered shopkeeper should be able to successfully accept, pack or reject a order.

Precondition: The user has ordered a from that shopkeeper's shop.
Assumption: a supported browser is being used.

Test Steps:

1. Click 'Accept' button to accept order.
2. Click 'Delete' button to delete order.
3. Click 'Pack' button to pack order.

Expected Result: Shopkeeper should be able to accept, pack or delete an order.
Actual Result: Shopkeeper can accept, pack or delete an order.

Title: Delivery boy registration – Successfully register a new delivery boy.
Description: A new user should be able to successfully register themselves as delivery boy.

Precondition: The user has given valid credentials.
Assumption: a supported browser is being used.

Test Steps:

1. Click 'Sign Up' button.
2. Enter valid credentials in the field.
3. Enter OTP generated on mobile.
4. Click 'Register' button.

Expected Result: Delivery boy should be successfully registered on the website.
Actual Result: Delivery boy is successfully registered.

Title: Delivery boy login – Successful login in website.
Description: A registered delivery boy should be able to successfully login in website.
Precondition: The delivery boy is successfully registered.
Assumption: a supported browser is being used.

Test Steps:

1. Click 'SignIn' option.
2. Enter gmail id.
3. Enter password.
4. Click 'Log In' button.

Expected Result: Delivery boy should be successfully logged in and redirected to dashboard page .
Actual Result: Delivery boy can see it's dashboard page.

Title: Delivery Boy Accept Order – Successful accept of an order.
Description: A registered delivery boy should be able to successfully accept an order for delivery.

Precondition: The shopkeeper has requested for a delivery.
Assumption: a supported browser is being used.

Test Steps:

1. Navigate to the View New Orders page.
2. Click 'Accept order for pick' button.

Expected Result: Delivery boy should be successfully accept an order for delivery .
Actual Result: Delivery boy can accept an order.

Title: Delivery Boy Pick Order – Successfully pick an order.
Description: A registered delivery boy should be able to successfully pick an order for delivery.
Precondition: The shopkeeper has requested for a delivery.
Assumption: a supported browser is being used.

Test Steps:

1. Navigate to the Pick Order page.
2. See the products ready for delivery..

Expected Result: Delivery boy should be successfully pick an order for delivery .
Actual Result: Delivery boy is able to pick an order.

Title: Delivery Boy Delivered Order – Successfully navigate to the delivered orders page .
Description: A registered delivery boy should be able to successfully navigate to the delivered orders page.
Precondition: The delivery has delivered a delivery.
Assumption: a supported browser is being used.

Test Steps:

1. Navigate to the Delivered Orders page.
2. See the products that are successfully delivered.

Expected Result: Delivery boy should be successfully see delivered products .
Actual Result: Delivery boy can see the delivered orders.

7.2.1 Software Quality Attributes

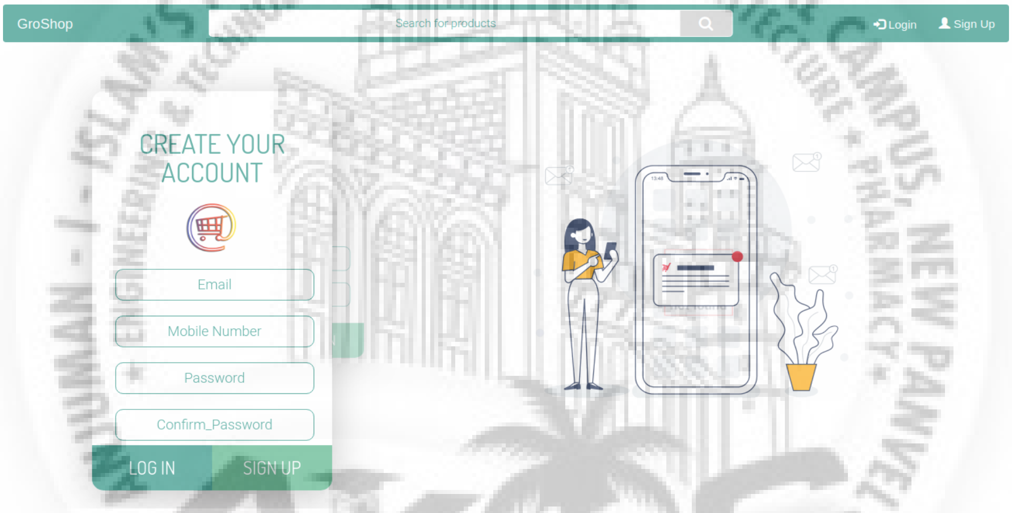
1. AVAILABILITY: The system should not be down, whenever the user use the system the specific data should be available to the user.
2. CORRECTNESS: As per the user search the the correct should be shown to the user like at time for searching the the similar type of startup the system should show all the similar startup.
3. MAINTAINABILITY: The administration of the system will maintain the system with effective updates though on air update if needed.

Chapter 8

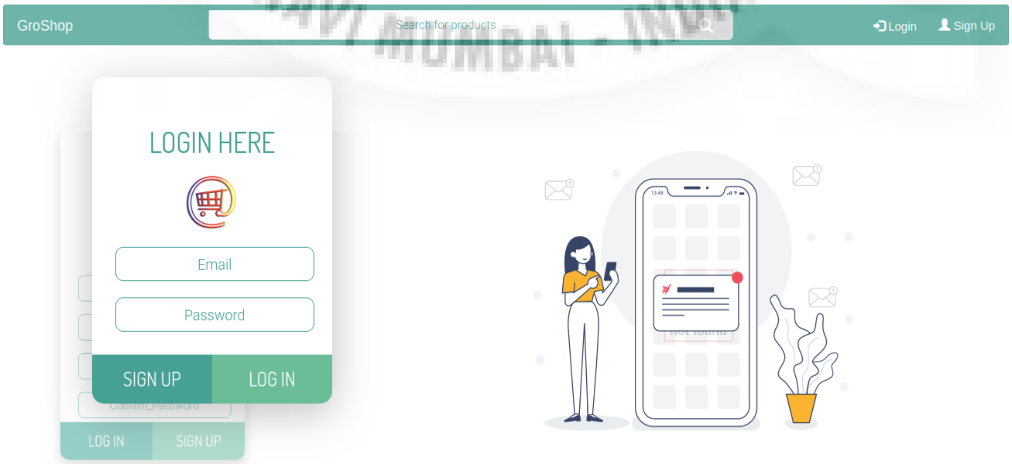
Screenshots of Project

8.1 User View

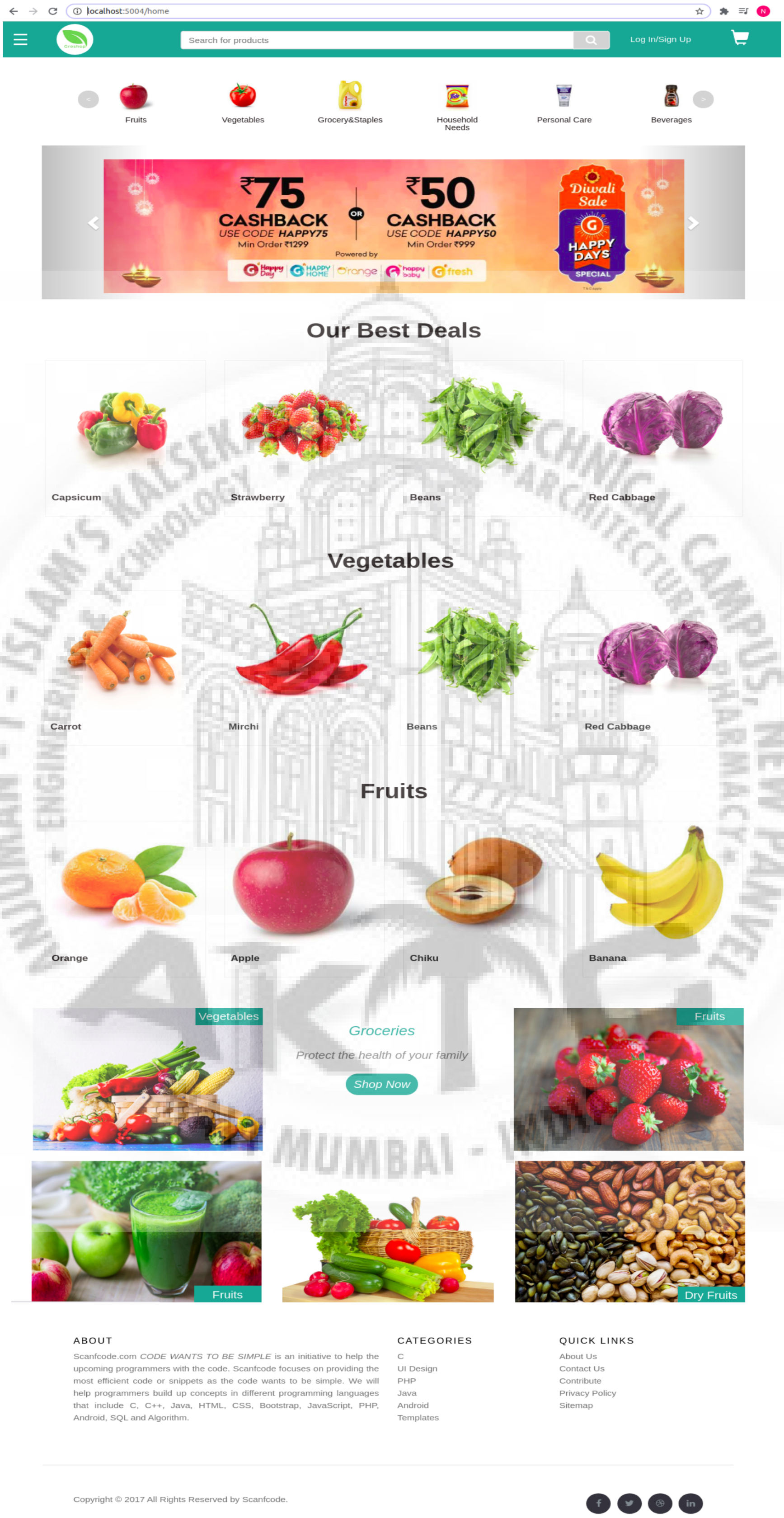
8.1.1 Registration



8.1.2 Login



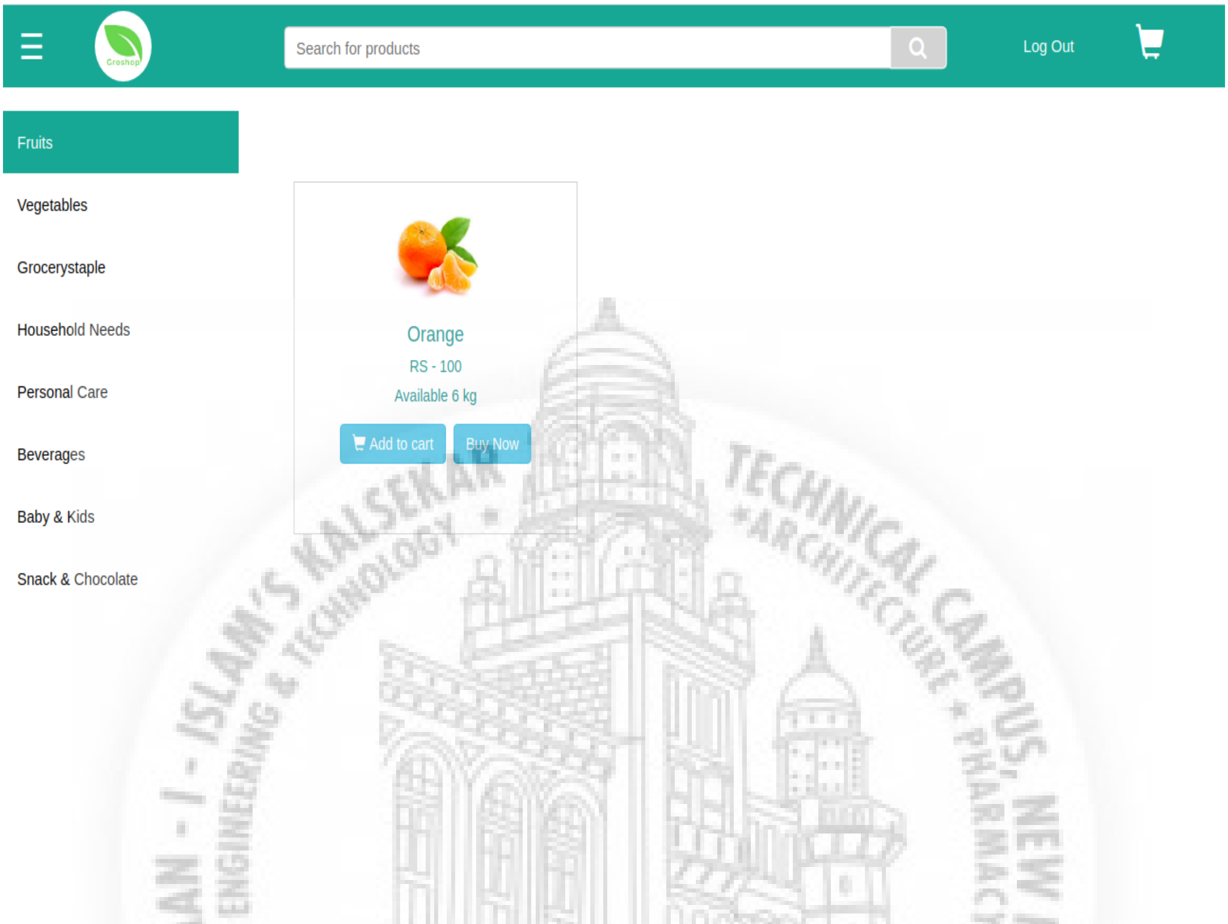
8.1.3 Home page

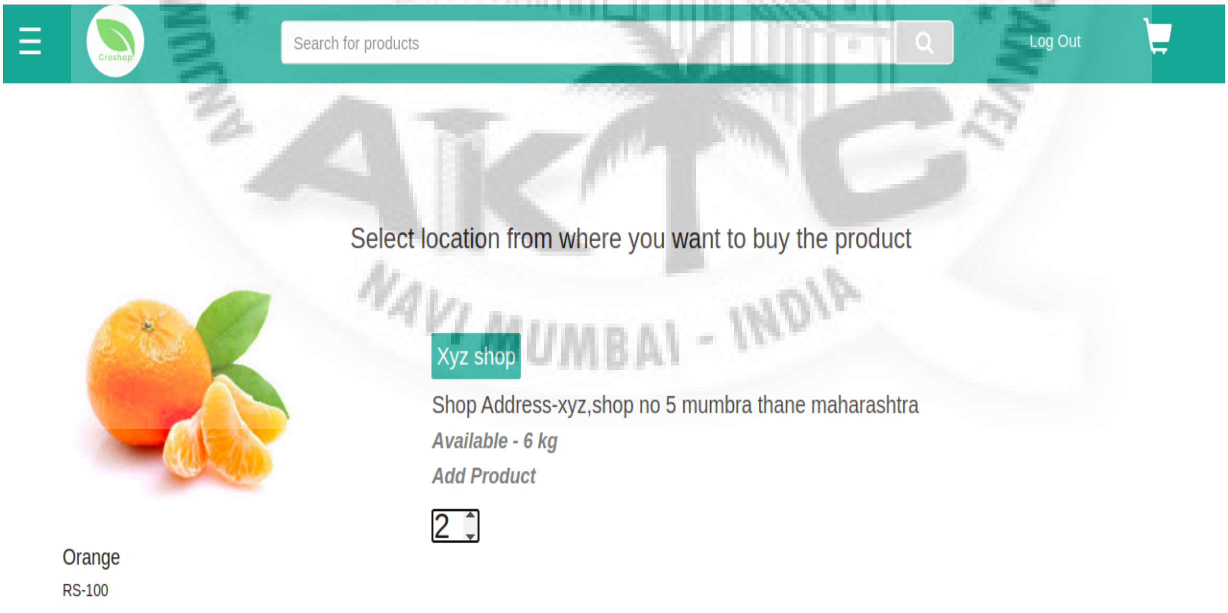


Scanfcode.com *CODE WANTS TO BE SIMPLE* is an initiative to help the upcoming programmers with the code. Scanfcode focuses on providing the most efficient code or snippets as the code wants to be simple. We will help programmers build up concepts in different programming languages that include C, C++, Java, HTML, CSS, Bootstrap, JavaScript, PHP, Android, SQL and Algorithm.

Copyright © 2017 All Rights Reserved by Scanfcode.

8.1.4 Cart product





Select location from where you want to buy the product

Xyz shop

Shop Address-xyz,shop no 5 mumbra thane maharashtra

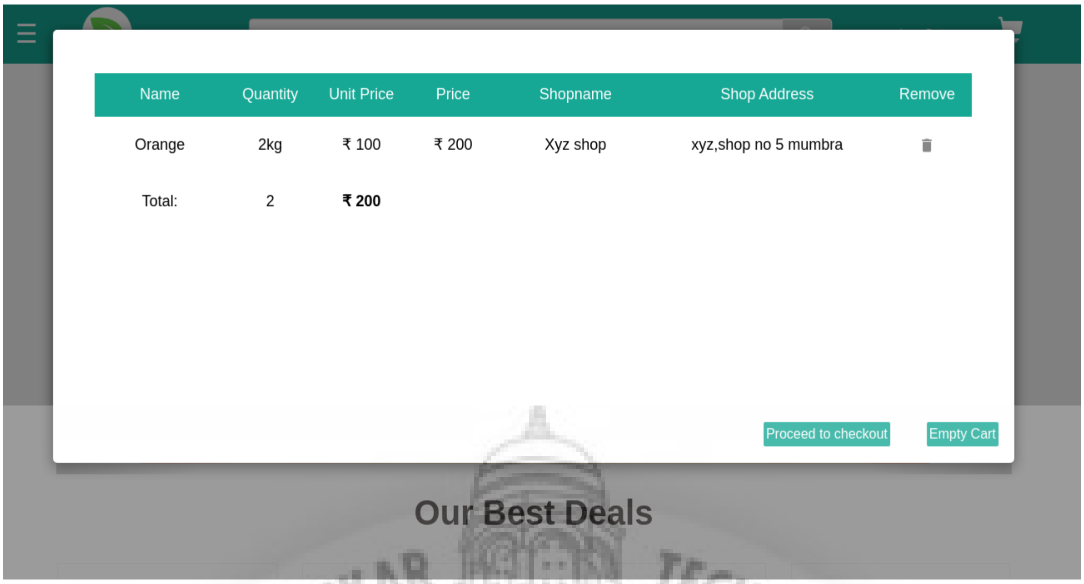
Available - 6 kg

Add Product

2

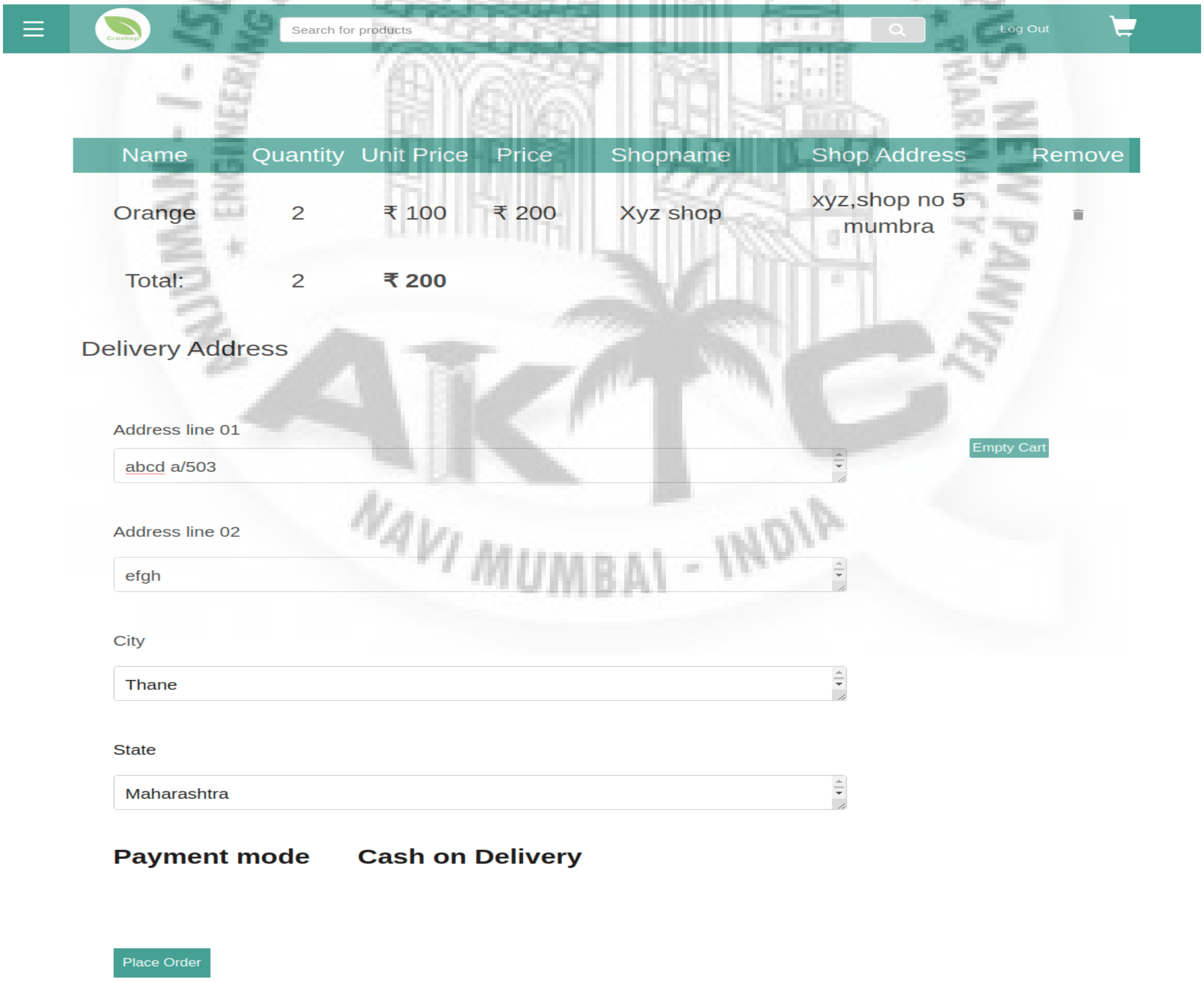
Orange

RS-100



Name	Quantity	Unit Price	Price	Shopname	Shop Address	Remove
Orange	2kg	₹ 100	₹ 200	Xyz shop	xyz,shop no 5 mumbra	🗑
Total:	2	**₹ 200**				

8.1.5 Checkout product



Name	Quantity	Unit Price	Price	Shopname	Shop Address	Remove
Orange	2	₹ 100	₹ 200	Xyz shop	xyz,shop no 5 mumbra	🗑
Total:	2	**₹ 200**				

Delivery Address

Address line 01: abcd a/503

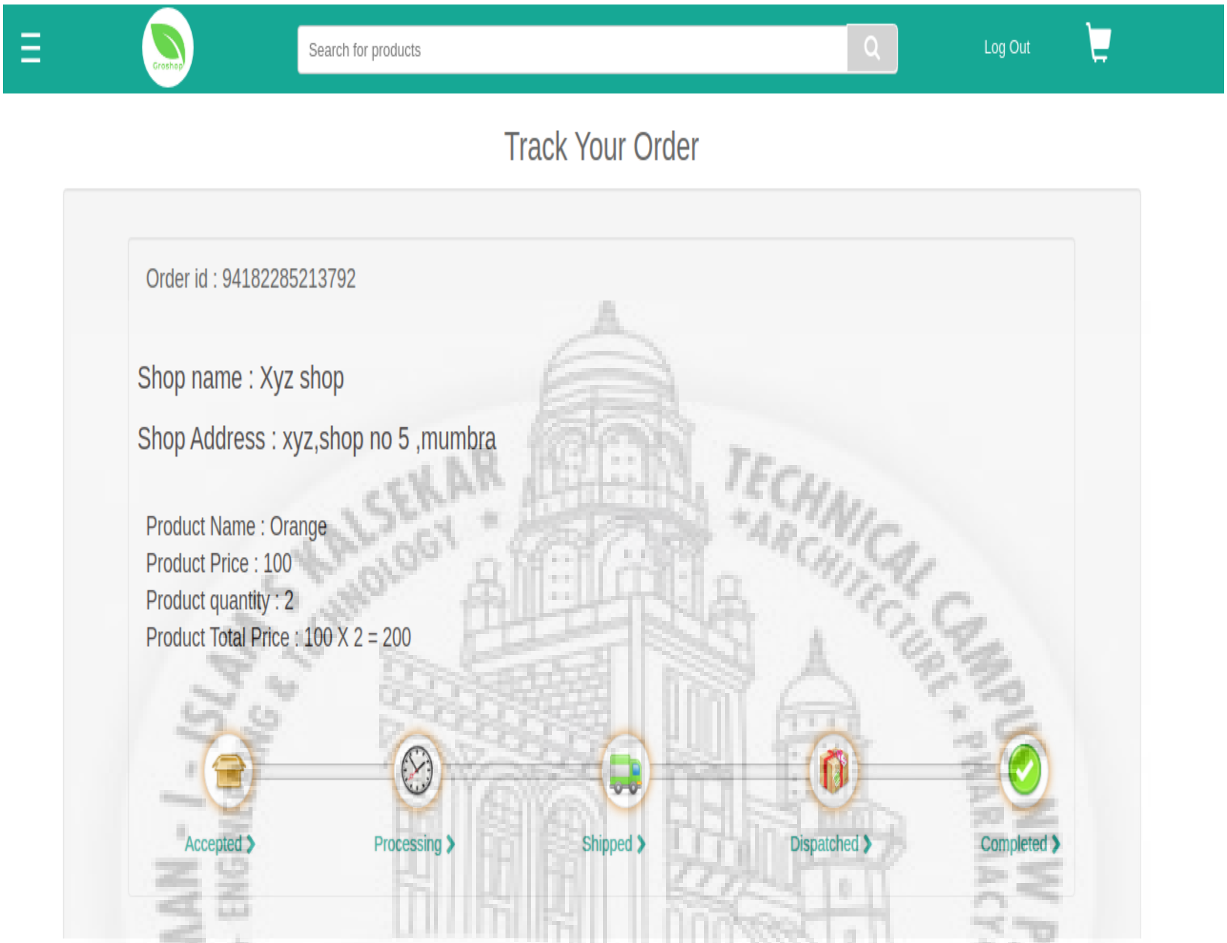
Address line 02: efgh

City: Thane

State: Maharashtra

Payment mode **Cash on Delivery**

8.1.6 Track product



Track Your Order

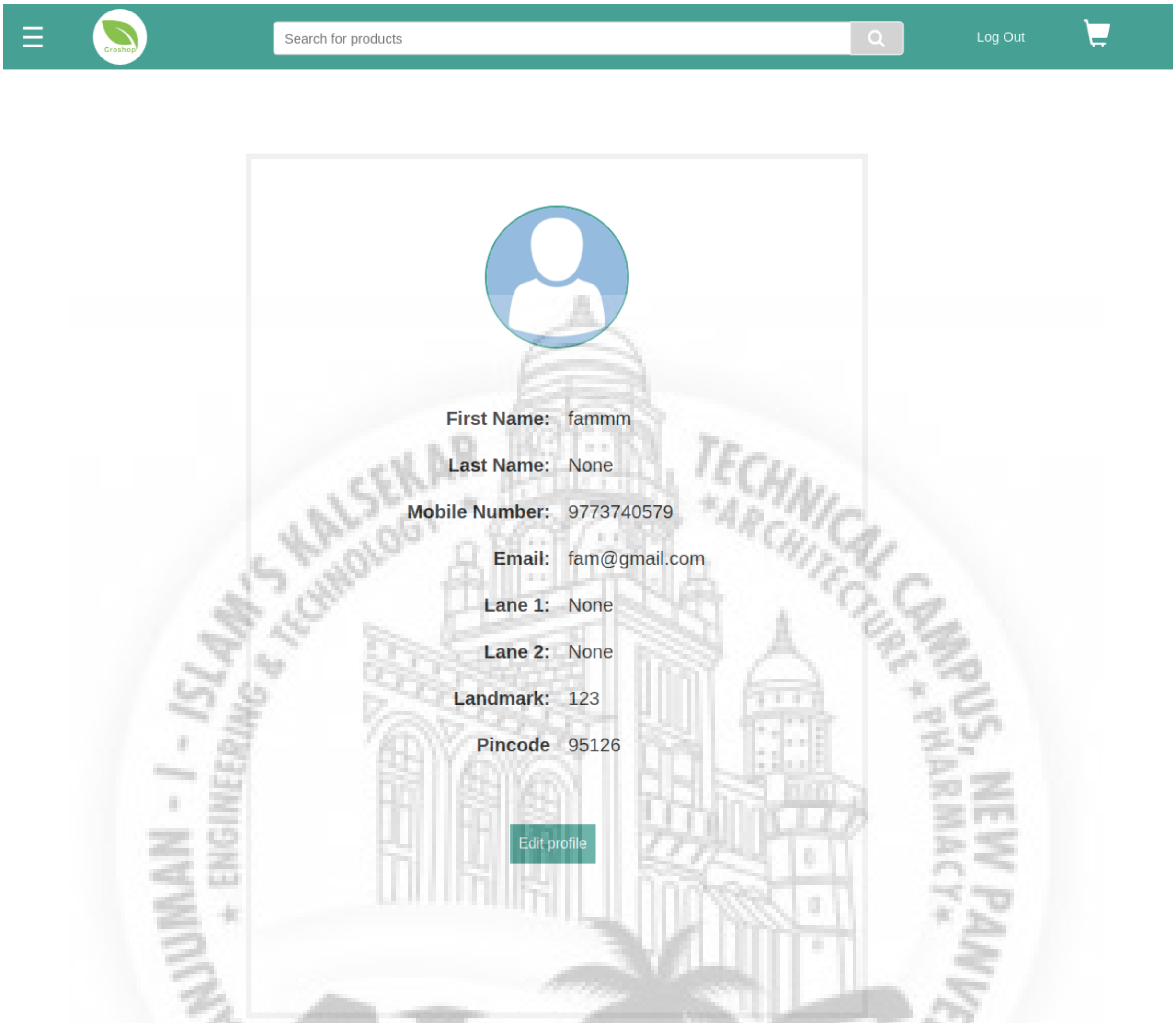
Order id : 941822285213792

Shop name : Xyz shop

Shop Address : xyz,shop no 5 ,mumbra

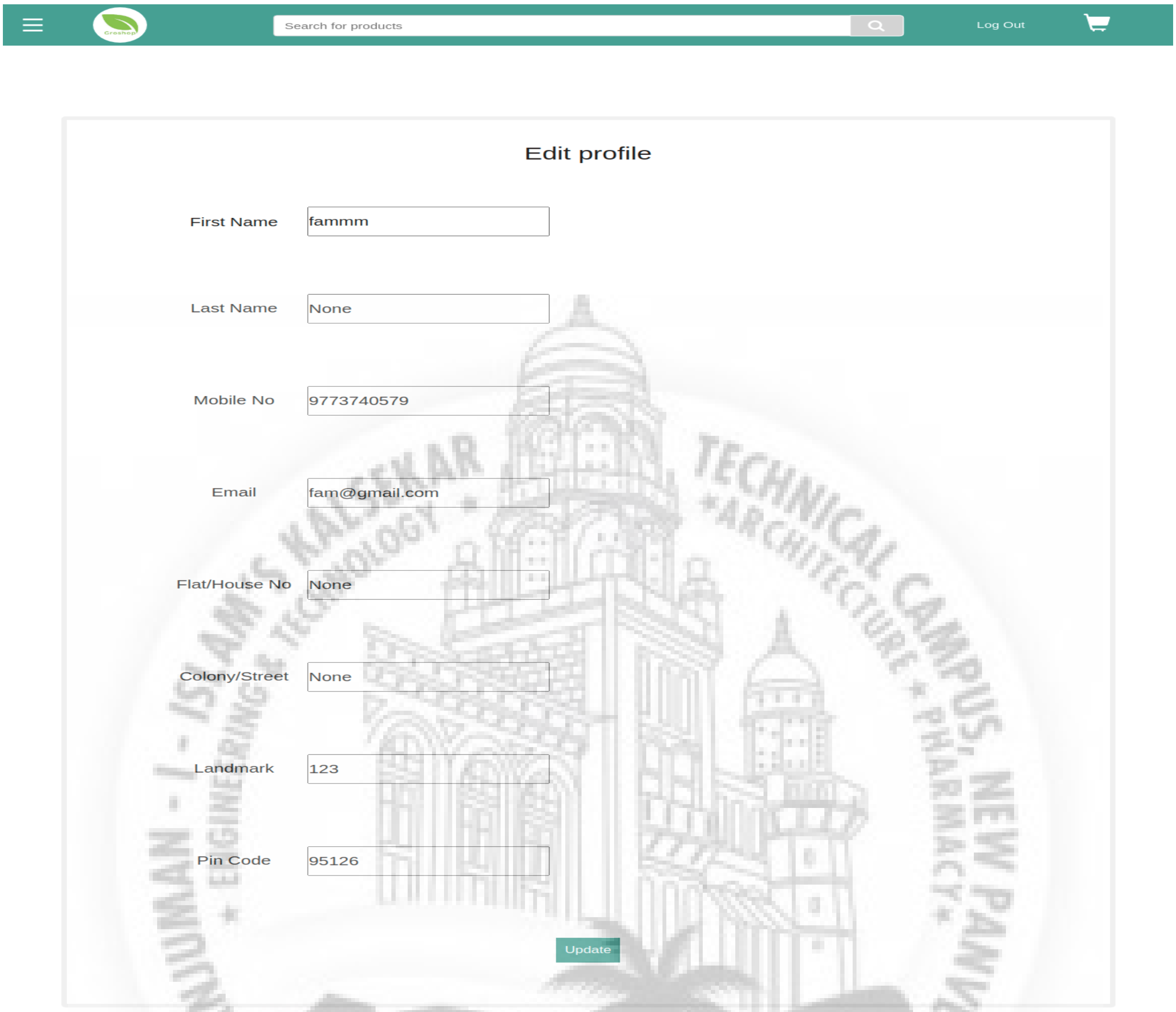
Product Name : Orange
Product Price : 100
Product quantity : 2
Product Total Price : 100 X 2 = 200

8.1.7 User Profile



First Name:	fammm
Last Name:	None
Mobile Number:	9773740579
Email:	fam@gmail.com
Lane 1:	None
Lane 2:	None
Landmark:	123
Pincode	95126

8.1.8 User Profile Edit

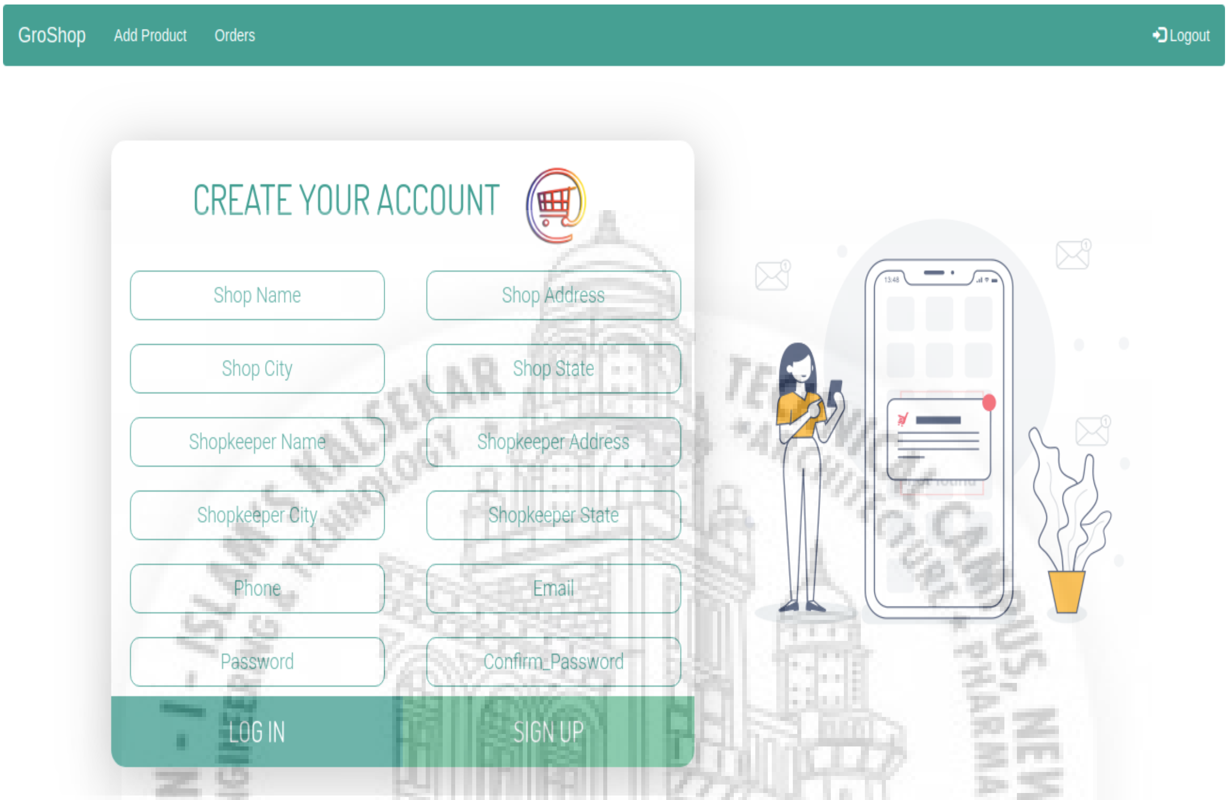


Edit profile

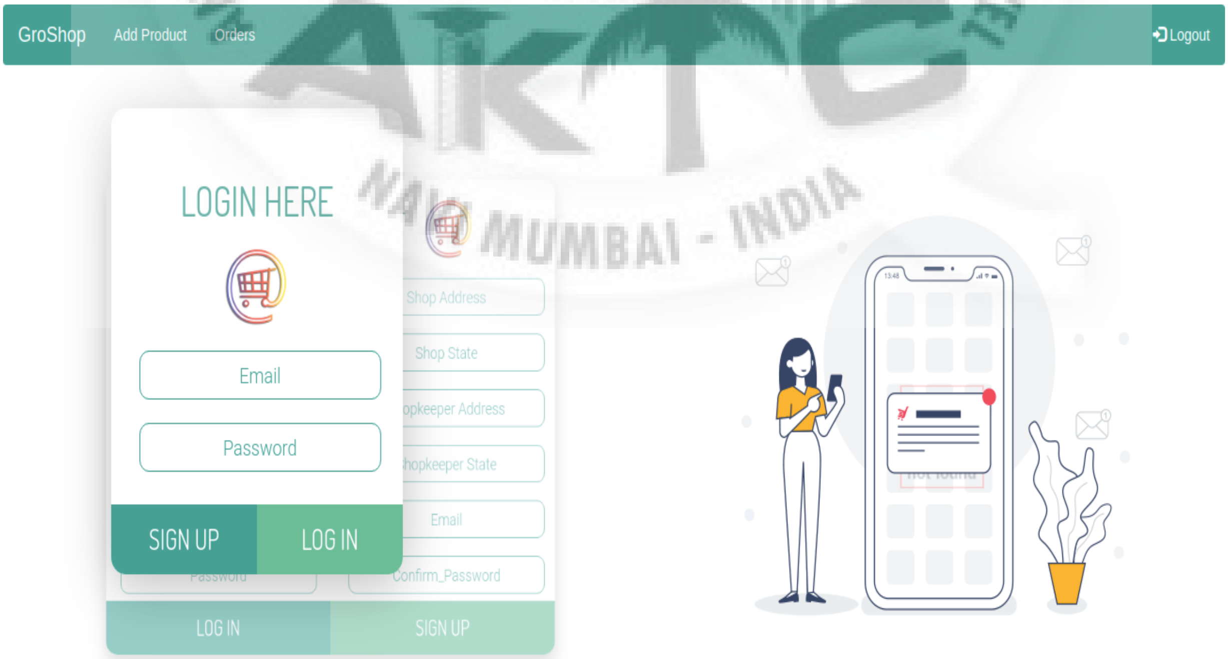
First Name	fammm
Last Name	None
Mobile No	9773740579
Email	fam@gmail.com
Flat/House No	None
Colony/Street	None
Landmark	123
Pin Code	95126

8.2 Shopkeeper View

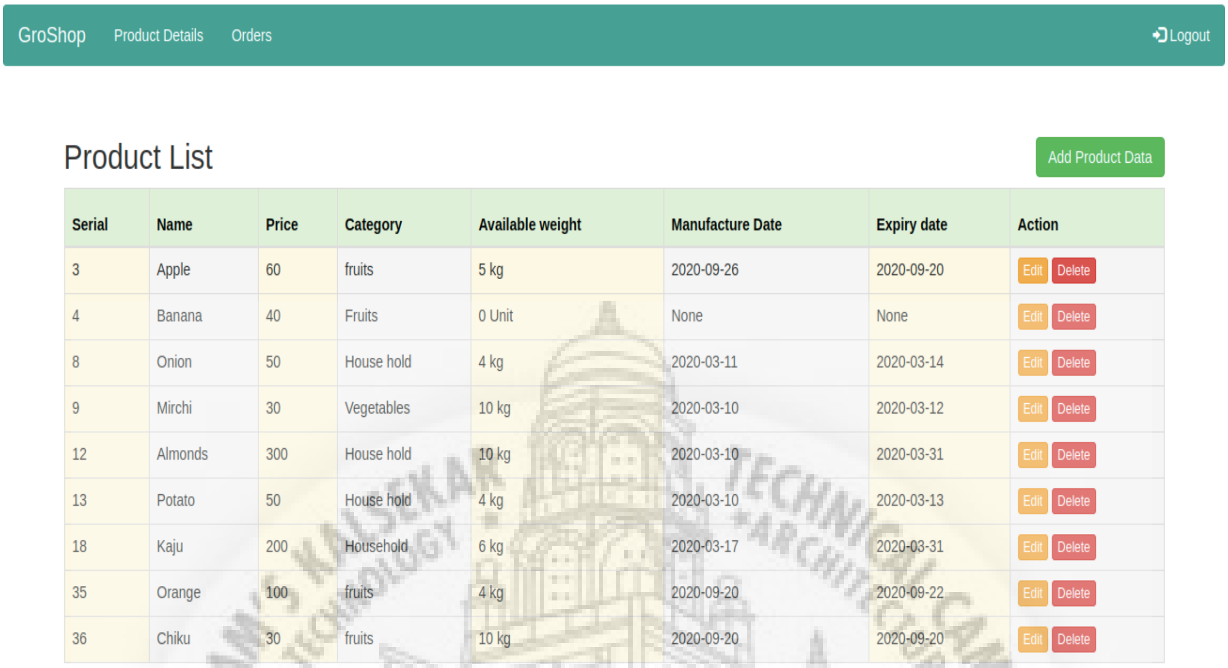
8.2.1 Registration



8.2.2 Login



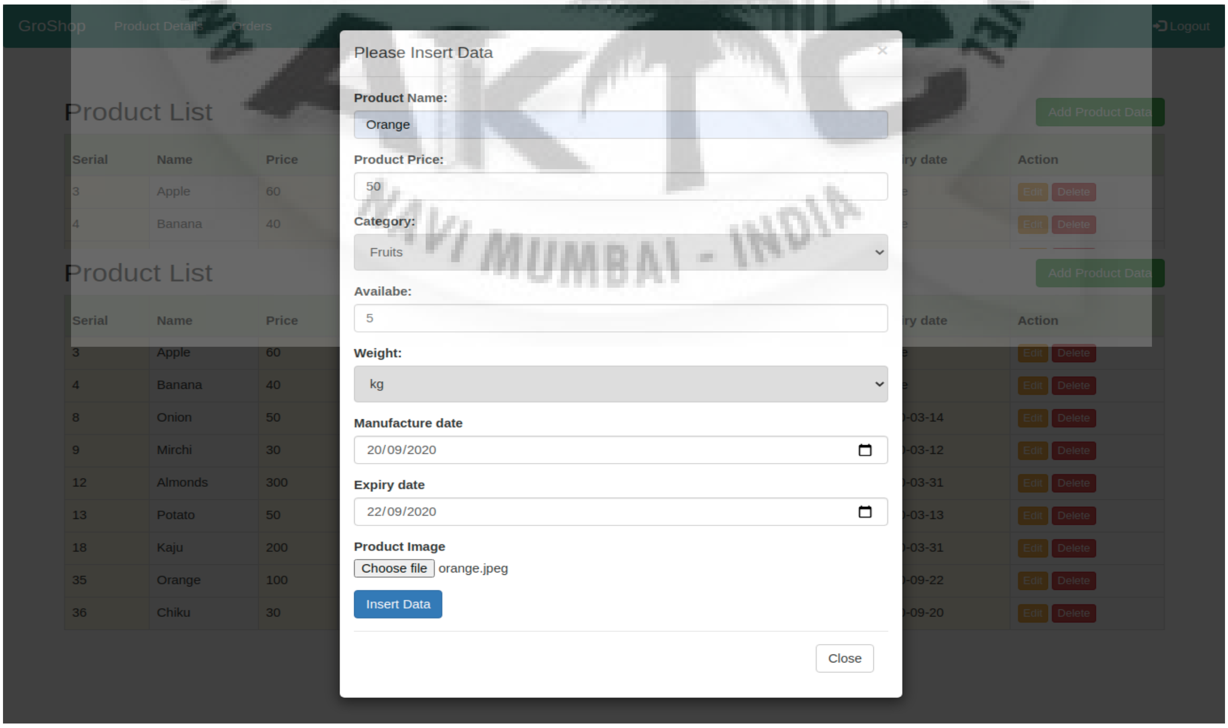
8.2.3 Product List of shopkeeper



Product List

Serial	Name	Price	Category	Available weight	Manufacture Date	Expiry date	Action
3	Apple	60	fruits	5 kg	2020-09-26	2020-09-20	Edit Delete
4	Banana	40	Fruits	0 Unit	None	None	Edit Delete
8	Onion	50	House hold	4 kg	2020-03-11	2020-03-14	Edit Delete
9	Mirchi	30	Vegetables	10 kg	2020-03-10	2020-03-12	Edit Delete
12	Almonds	300	House hold	10 kg	2020-03-10	2020-03-31	Edit Delete
13	Potato	50	House hold	4 kg	2020-03-10	2020-03-13	Edit Delete
18	Kaju	200	Household	6 kg	2020-03-17	2020-03-31	Edit Delete
35	Orange	100	fruits	4 kg	2020-09-20	2020-09-22	Edit Delete
36	Chiku	30	fruits	10 kg	2020-09-20	2020-09-20	Edit Delete

8.2.4 Add Product On Shopkeeper side



Please Insert Data

Product Name: Orange

Product Price: 50

Category: Fruits

Availabe: 5

Weight: kg

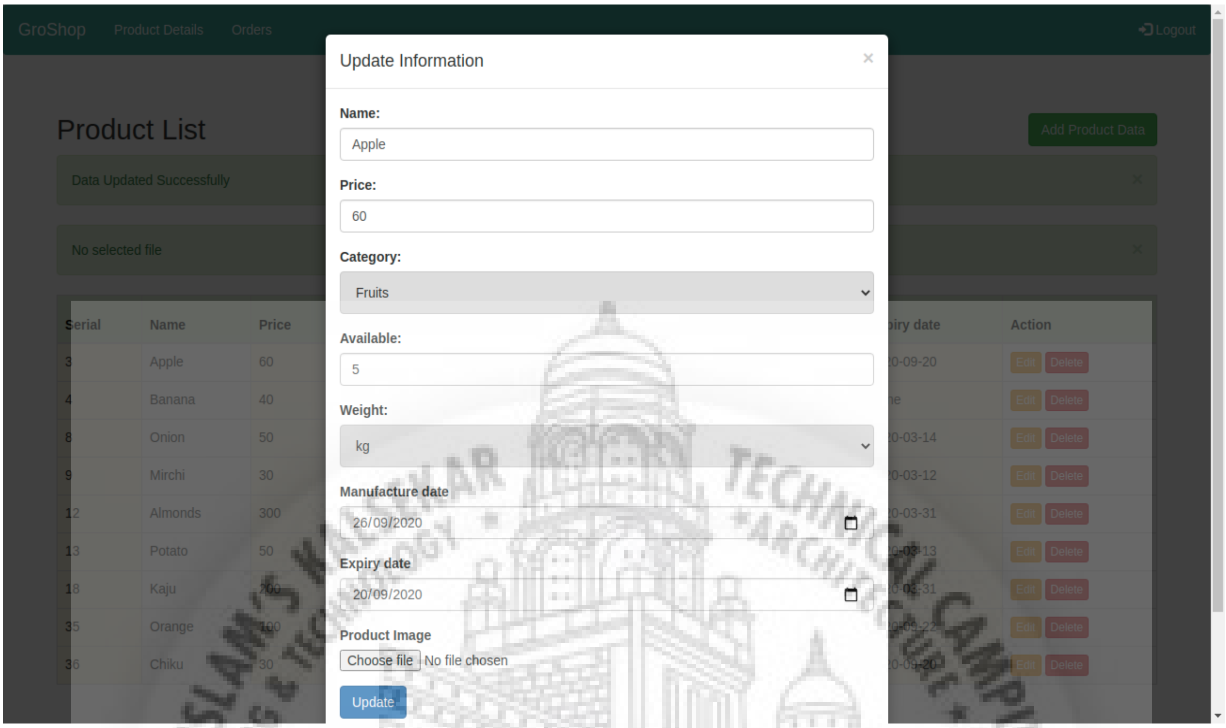
Manufacture date 20/09/2020

Expiry date 22/09/2020

Product Image Choose file orange.jpeg

Insert Data

8.2.5 Edit/Update Product On Shopkeeper side



Update Information ×

Name: Apple

Price: 60

Category: Fruits

Available: 5

Weight: kg

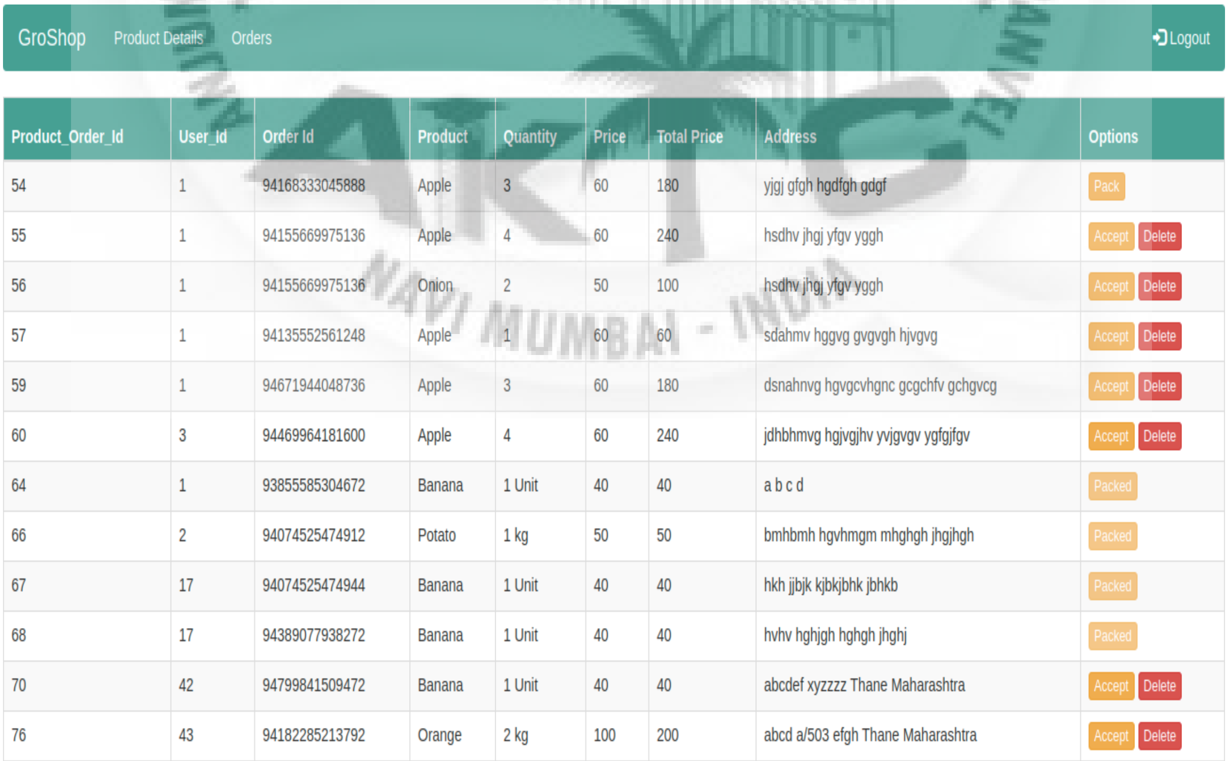
Manufacture date: 26/09/2020

Expiry date: 20/09/2020

Product Image: Choose file No file chosen

Update

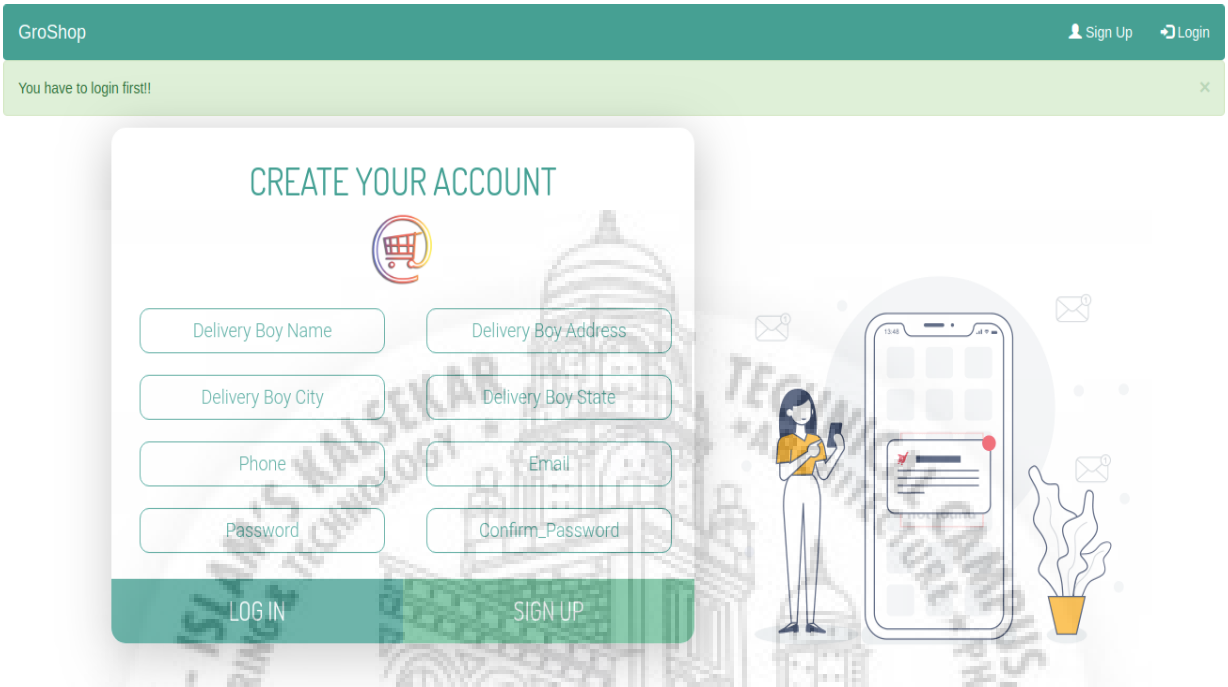
8.2.6 Ordered Product Accept/ Packed/ Reject



Product_Order_Id	User_id	Order Id	Product	Quantity	Price	Total Price	Address	Options
54	1	94168333045888	Apple	3	60	180	yjgj gfgh hgdfgh gdgf	Pack
55	1	94155669975136	Apple	4	60	240	hsdhv jhgj yfgv yggh	Accept Delete
56	1	94155669975136	Onion	2	50	100	hsdhv jhgj yfgv yggh	Accept Delete
57	1	94135552561248	Apple	1	60	60	sdahmv hggvg gvgvgh hjvgvg	Accept Delete
59	1	94671944048736	Apple	3	60	180	dsnahnvg hgvgcvhgnc gcgchfv gchgvcg	Accept Delete
60	3	94469964181600	Apple	4	60	240	jdhbhmvg hgjvgjhv yvjgvgv ygfgjfgv	Accept Delete
64	1	93855585304672	Banana	1 Unit	40	40	a b c d	Packed
66	2	94074525474912	Potato	1 kg	50	50	bmhbmh hgvhmgm mhghgh jhgjhgh	Packed
67	17	94074525474944	Banana	1 Unit	40	40	hkh jjbjk kjbkjbhk jbhkb	Packed
68	17	94389077938272	Banana	1 Unit	40	40	hvhv hghjgh hghgh jhghj	Packed
70	42	94799841509472	Banana	1 Unit	40	40	abcdef xyzzzz Thane Maharashtra	Accept Delete
76	43	94182285213792	Orange	2 kg	100	200	abcd a/503 efgh Thane Maharashtra	Accept Delete

8.3 Delivery Boy View

8.3.1 Registration



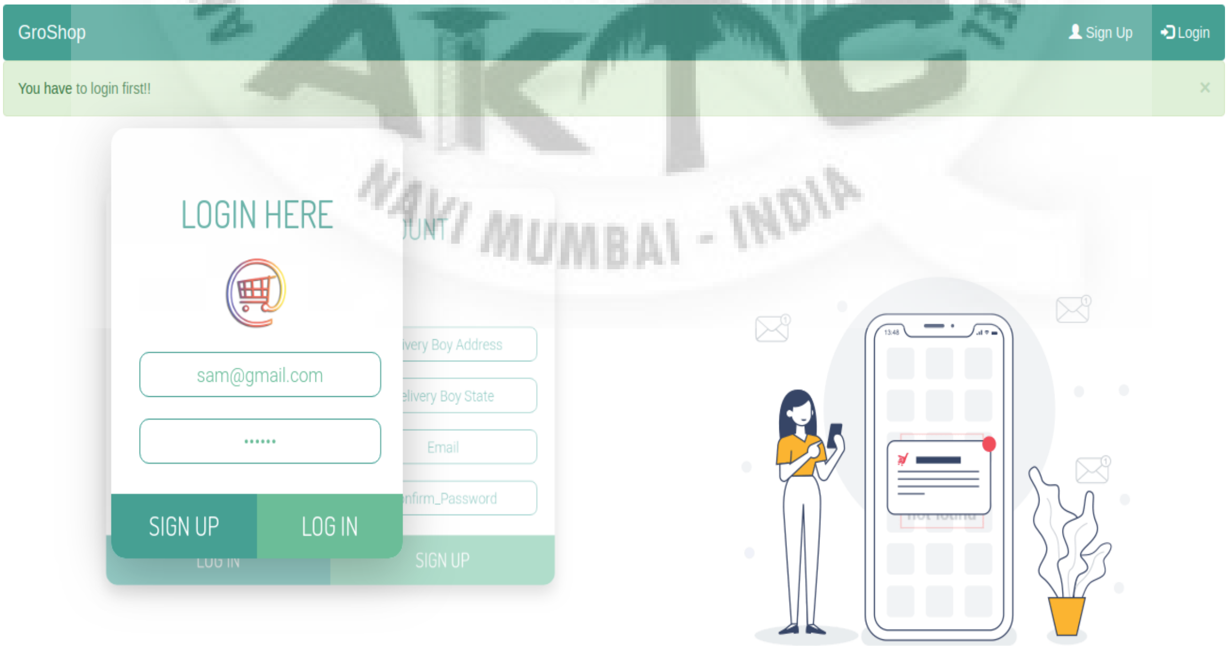
GroShop

Sign Up Login

You have to login first!!

CREATE YOUR ACCOUNT

8.3.2 Login



GroShop

Sign Up Login

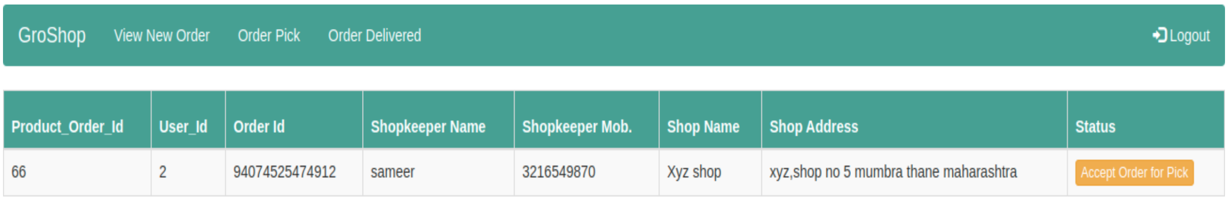
You have to login first!!

LOGIN HERE

sam@gmail.com

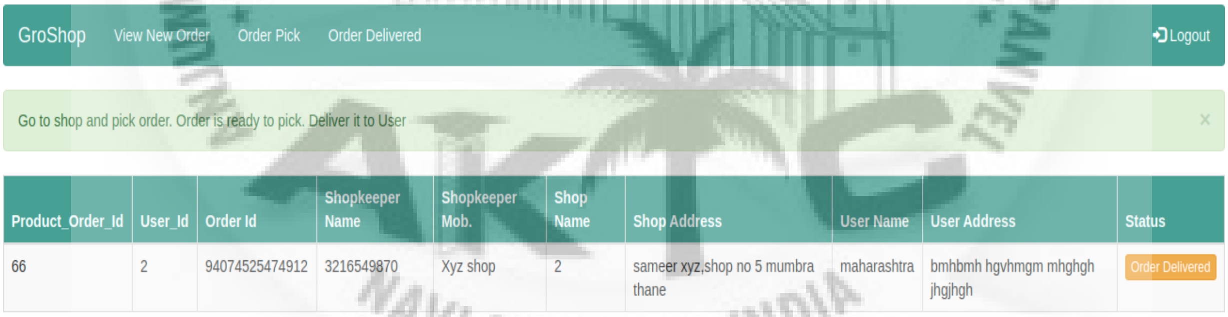
••••••

8.3.3 View New Orders



Product_Order_Id	User_Id	Order Id	Shopkeeper Name	Shopkeeper Mob.	Shop Name	Shop Address	Status
66	2	94074525474912	sameer	3216549870	Xyz shop	xyz,shop no 5 mumbra thane maharashtra	Accept Order for Pick

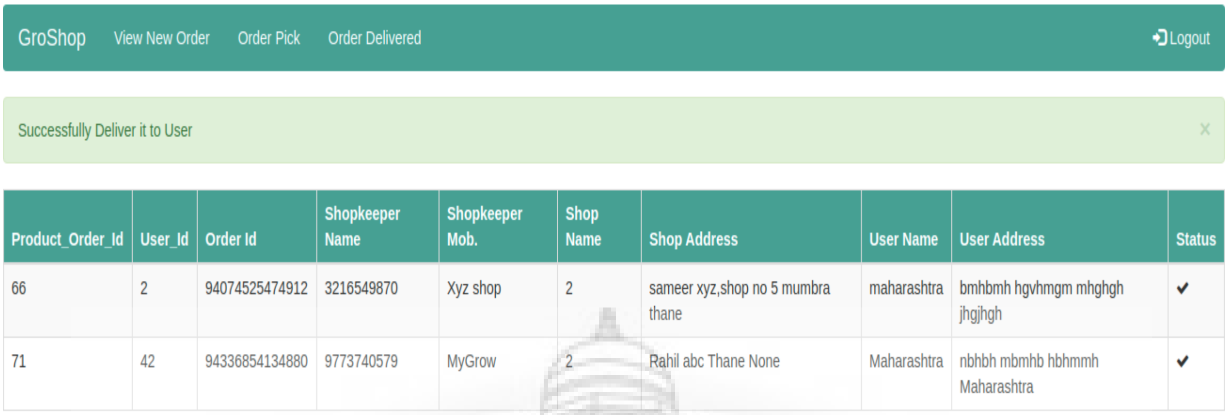
8.3.4 Order Pick



Go to shop and pick order. Order is ready to pick. Deliver it to User

Product_Order_Id	User_Id	Order Id	Shopkeeper Name	Shopkeeper Mob.	Shop Name	Shop Address	User Name	User Address	Status
66	2	94074525474912	3216549870	Xyz shop	2	sameer xyz,shop no 5 mumbra thane	maharashtra	bmhbmh hgvhmgm mhghgh jhgjhgh	Order Delivered

8.3.5 Delivered Orders



Successfully Deliver it to User

Product_Order_Id	User_Id	Order Id	Shopkeeper Name	Shopkeeper Mob.	Shop Name	Shop Address	User Name	User Address	Status
66	2	94074525474912	3216549870	Xyz shop	2	sameer xyz,shop no 5 mumbra thane	maharashtra	bmhbmh hgvhmgm mhghgh jhgjhgh	✔
71	42	94336854134880	9773740579	MyGrow	2	Rahil abc Thane None	Maharashtra	nbhbh mbmhb hbhmmh Maharashtra	✔

Chapter 9

Conclusion and Future Scope

9.1 Conclusion

Now, this is the era of internet revolution, human activities are changing very fast. Online shopping is considered one of the most necessary part in our day to day life. In the world of online shopping buying groceries is still less popular.

Online Shopping is undoubtedly on an upward trajectory. Now, 450 billion dollar in revenues per year takes in by super market industry and the Internet groceries should increase to take a large portion of that amount. These online grocery stores should give benefit to Consumers because of how convenient, they are but price benefits may also evolve.

Though, there are already several applications built, but each one has their pros cons. The features of this product will make it a better system to stand out. With numerous features we are trying to build a website with latest technologies which will overcome the flaws of previous systems.

Being a connecting link between the sellers and buyers, we need to make sure the order gets delivered on time, because that's the key of our trading model.

The important characteristics of Products Recommender are reminding the user for any item which is possibly missing that they can be interested in, Automated smart list generation to show the list of products in one place, voice- over text gives user facility to control modern devices by talking rather than using buttons and keyboards for typing. In future different filtering can be applied to make it more flexible.

9.2 Future Scope

- Currently proposed system aim to deliver product recommendation, voice search, smart list features and so many other features which described above.
- But it doesn't deal with visual perception and chat bot for customers to interact while shopping.
- These features can be included in future to make the system more powerful.

References

[1] *The Smart Shopping List An effective mobile solution for grocery list-creation process*; Harsha Jayawilal W.A. and Saminda Premeratne,2017 IEEE 13th Malaysia International Conference on Communications (MICC), 28-30 Nov. 2017

[2] *Smart Shopping Technologies for Indoor Markets*; Ioana Rogojanu, George Suciu,Maria-Cristina Ditua nnd Adrian Pasat,2018 IEEE International Conference on Computational Science and Engineering, 2018

[3] *Assessing the Profitable Conditions of Online Grocery Using Simulation*; Ahmed Alzubairi 1 and Abdullah Alrabghi 2,Proceedings of the 2017 IEEE IEEM, 2017

[4] *A Method to Study How Older Adults Navigate in an Online Grocery Shopping Site*; Harsha Jayawilal W.A. and Saminda Premeratne,2016 4th International Conference on User Science and Engineering (i-USEr), 2016

[5] *Artificial Intelligence - Creating Automated Insights for Customer Relationship Management*; Suman Kumar deb and Ruchi Jain,2018 8th International Conference on Cloud Computing, Data Science Engineering (Confluence), 2018

[6] WRITE A SHORT-NAME WITHOUT SPACE *NAME OF site*; NAME OF AUTHORS,`http://EXAMPLE.com`,month and year of access

[7] Introduction to Flask *pymbook*;`https://pymbook.readthedocs.io/en/latest/flask.html`

[8] Why should we use Flask for web development?*Quora*;https://www.quora.com/Why-should-we-use-Flask-for-web-development , August 2015

[9] Explain what Flask is and its benefits?*i2 tutorials*;https://www.i2tutorials.com/technology/explain-what-flask-is-and-its-benefits/,November 2018

[10] Machine Learning*GeeksforGeeks*;https://www.geeksforgeeks.org/machine-learning/

[11] HTML Tutorials*GeeksforGeeks* https://www.geeksforgeeks.org/html-tutorials/

[12] CSS Tutorials*GeeksforGeeks* https://www.geeksforgeeks.org/css-tutorials/

[13] Python (programming language)*En.wikipedia.org.* https://en.wikipedia.org/wiki/Python(programminglanguage), Oct 2018

Achievements

1. Publication

   (a) *Online Grocery Delivery System*; Nooras Fatima Ansari, Amina Sarang, Mariyam Shaikh, BJIT – “International Journal of Information Technology” 2020 (https://www.springer.com/journal/41870)

2. Conferences

   (a) *Online Grocery Delivery System*; Nooras Fatima Ansari, Amina Sarang, Mariyam Shaikh, 5th INTERNATIONAL CONFERENCE ON ENGINEERING RESEARCH INNOVATION, 19TH to 20TH March, 2020 (Venue : Terna Engineering College, Navi Mumbai)

3. Project Competitions

   (a) *Online Grocery Delivery System*; Nooras Fatima Ansari, Amina Sarang, Mariyam Shaikh, 6th National Level Project Exhibition cum Poster Presentation , 13th March 2020 (Venue : Universal College of Engineering)