**Grocery Store Automation System**

Submitted in partial fulfilment of the requirements

of the degree of

**Bachelor of Technology**

by

**Mr. Shyam Ganesh Pradhan**

**(Reg. No. 2018BCS509 / A76)**

Project Guide:

**Prof. D. R. Bilolikar**

****

**Department of Computer Science and Engineering,**

**Shri Guru Gobind Singhji Institute of Engineering & Technology, Vishnupuri, Nanded, Maharashtra, India, 431606.**

**2020-21**

**DECLARATION**

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

**Shyam G. Pradhan**

**(2018bcs509 / A76)**

Date: \_\_\_\_\_\_\_\_\_\_

**CERTIFICATE**

This is to certify that project entitled “**Grocery Store Automation System**” which is submitted by Name of **Shyam G. Pradhan (2018bcs509 / A76)** in partial fulfilment of the requirement for the award of degree Bachelor of Technology in Computer Science and Engineering to Shri Guru Gobind Singhji Institute of Engineering & Technology, Vishnupuri, Nanded (M. S.) for Project during academic year 2020-21 and has been carried out under our supervision.

**Project Guide Department Head**

**Prof. D. R. Bilolikar Prof. S. S. Hatkar**

**ACKNOWLEDGEMENT**

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend our sincere thanks to all of them.

I am highly indebted to **Prof. D. R. Bilolikar** for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express our gratitude towards our parents & members of **SHRI GURU GOBINDSINGJI INSTITUTE OF ENGINEERING AND TECHONLOGY** for their kind co-operation and encouragement which help me in completion of this project.

I would like to express our special gratitude and thanks to industry persons for giving us such attention and time.

Our thanks and appreciations also go to our colleague in developing the project and people who have willingly helped me out with their abilities.

I would like to express our deepest appreciation to all those who provided us the possibility to complete this report.  A special gratitude to the project guide **Prof. D. R. Bilolikar.**

**Shyam G. Pradhan**

**(2018bcs509 / A76)**

**Abstract**

In this internet era, we all know the importance of ecommerce. Basically, Electronic Commerce is process of doing business through computer networks. A person sitting on his chair in front of a computer can access all the facilities of the Internet to buy or sell the products.

Due to pandemic situation, All the shop owners of small shop had faced lots of problem for buying, selling and storing products. The customers had faced the problems of buying essential products etc. for buying something they physically has to go to shop and buy essentials. Big ecommerce web sites also provide this kind of service but this takes a time to deliver products. The only option available to customer is that to go to shop to buy it. In this project I am trying to solve this problem. By taking surveys to the small grocery shop we come to know that they sell their products to the people who are living in 2km. circle. So, we are going to develop the website which can help shopkeeper to manage and sell their product. And it will be helpful for customer to buy products from the shop.

In this project we are concentrating on Grocery Shop automation system. Basically, I am developing the website which will help shopkeepers to sell their products online. They can manage their shop through online portal. I will add smart inventory system to manage the inventory, there are lots of modules in which we are going to use machine learning like recommendation system, efficient inventory management, etc.

**LIST OF CONTENTS**

1. **INTRODUCTION 1-2**
   1. Surveys 1-2
   2. Current Solutions 2
   3. How this Solution is More efficient? 2
2. **SOFTWARE APPROACH 3-4**

**2.1** Technology Stack 3-4

**2.2** Conceptual Diagram 4

1. **TECHNOLOGY USED 5-9**

**3.1** ReactJS 5

**3.2** Node JS 5-6

**3.3** Django 6

**3.4** MongoDB 6-7

**3.5** Machine learning (Content based Recommendation System) 7-9

1. **RESULT 10-18**

**4.1** Snapshots 10-17

**4.2** Future Scope 17-18

**4.3** Conclusion 18

1. **REFERENCES 19**

**CHAPTER 1**

**INTRODUCTION**

In this pandemic situation, buying and selling product online is safe. there will be fewer physical contacts if Someone buy or sell the product, which will help to reduce the risk on buyer and seller. This is online solution so that everyone can access from their remote location. The webapp is made in such a way so that it can cover some important features of e-commerce webapp.

The main idea behind this project is to make the web app or mobile app for selling groceries online. So that we will dividing modules in customer site and admin site. As this webapp can managed by the Admin/Owner of shop we are having two portals first portal is for admin and other one is like ecommerce webapp called customer site. At the admin site the admin can manage its product and inventory. Web app has smart indicator to show the status of stock available. Owner can manage orders they can see the stock report date wise and manage stock efficiently. Also, some of the graphical representation is given to show data graphically it boosts the efficiency in management of inventory. Same at the customer site we are having management of orders, profile management, place orders. In Modules section we will see each of them.

**1.1 Surveys**

Before starting work on the project, I started with survey I asked some questions to the grocery shop owners. Following are some question and answer that I got from surveys.

1. How grocery shop sells their products?

Local grocery shops don’t have any delivery system. Customer physically need to go to grocery shop to buy the groceries they want. We come to know that local grocery shops have customer in 2km. circle.

1. How they manage their inventory?

Some of grocery shops have their godown (Small Warehouse). They normally use book to manage product they have. Whenever the stock of some product is less, they order the same from different vendors. For that they have list of their merchants.

1. How they manage customers information?

Usually, the grocery shop doesn’t use to have any information about the customer.

1. Did they used online way to sell their products in pandemic?

Lots of grocery shop don’t use to have their business on internet. But some of them sell their product on WhatsApp or on mobile phone. These shops also have home delivery. For that they have hired delivery boy to deliver the products. For home delivery they are taking some cost.

**1.2 Current Solutions**

There is no such a solution to sell the product online. Although there are some online solutions through which customer can buy product online. But these cannot be helpful for local bodies. Like Amazon Pantry, Groofers are some of the online portal which sell the groceries online but it will not helpful for rural peoples. As we know most of Indian population has at least on smart phone in family. But still, they can’t buy groceries from these online solutions as these portals provide solution to some of the main metro cities. I did some research on them and I came to know that they take at least 2 to 4 days to deliver the groceries.

Some of local groceries shop provide delivery service but for that they don’t use any online solutions for this they mainly use WhatsApp Business app which is itself a third-party solution. And it doesn’t have that much features in it. Also, there are some grocery store which accept orders on phone call. Again, it is too inconvenient solution as it take too much time.

**1.3 How this solution is more efficient?**

From all surveys that I did and information that I gathered, I decided to make the web app which will solve most of the problems in the previous solutions. As the project is based on ecommerce web app, first I figured out all of the ecommerce features and tried to implement them in easiest way so that cost of making the app would be less. In the project I have mainly focused on the cost as it will used by the local grocery shop. Also, the main motive of this project is to make the app which will place the order and owner delivers it.

If we compare this solution with the current online solution like groofers and amazon pantry this is more sufficient in case of selling the products as we are buying products from local grocery shops which are very close to our current location so that it will take less amount of time. And at the owner’s end, they are getting complete solutions for their problems like managing stocks, accepting online orders. So, in this way it is more suitable and efficient way.

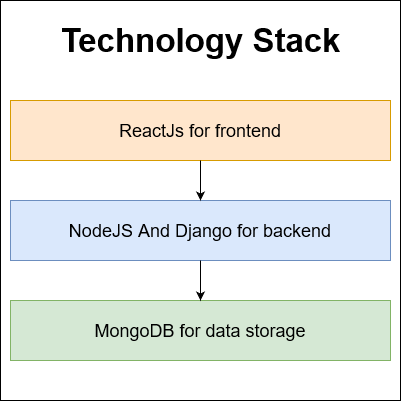
**CHAPTER 2**

**SOFTWARE APPROCH**

As this is ecommerce website for local grocery store, I have divided whole app into two different portal customer site and admin site. The main aim of customer site is to provide all ecommerce features like cart, product list, buying products, place orders, manage address, etc. these are some of the features that I needed to add in customer site. Now, if we consider admin site it is more on managing the product like managing stocks, adding and listing of products, managing orders etc.

**2.1 Technology Stack**

Following diagram is defining the technology stack of the project. All the modules are strictly followed while making the project.

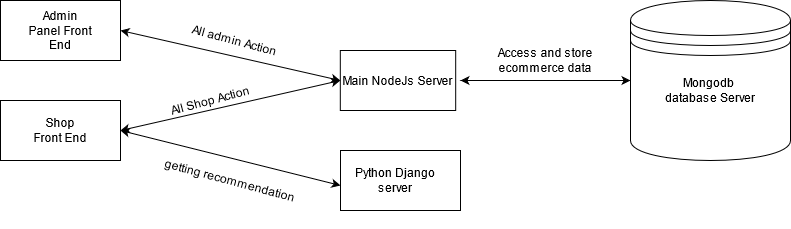


*Figure 1*

Mainly most of the front-end part is developed in the ReactJS. We will this technology in next chapter. And next is NodeJS and Django these technologies have been used for back-end of the project. As I am using REST API for making the app, I have developed end point which will be get and add the data into the server. Next is Database for data base I am using MongoDB as it more flexible and it has fast indexing to search data accurately.

**2.2 Conceptual Diagram**

Following diagram will show basic level of project design. It will elaborate the modules that have been used while making the project.

****

*Figure 2*

This is conceptual diagram of project. There are mainly three parts in it,

1. Front End
2. Servers
3. Database Servers

**Front End:** In the front end I am using ReactJS. This is web project so the html and CSS would be there. There are two parts of front end, first is Admin Panel which handles all operation of shopkeeper. It consists Management of product, Inventory, Orders Modules. And second module is for customer side. The basic task of this is to provide all e commerce features.

**Servers:** The next thing is server, I am using NODEJS for performing all the task of managing the ecommerce functions. And the DJANGO is for Recommending the product. As it uses a python, it is convenient to use for Machine Learning algorithms.

**Database Servers:** for storing the data I am using both relational and non-relational databases. For general storing of data, I am using MongoDB.

**CHAPTER 3**

**TECHNOLOGY USED**

Following are technologies that I have used to make the project,

**3.1 ReactJS**

ReactJS is an [open-source](https://en.wikipedia.org/wiki/Open-source), [front end](https://en.wikipedia.org/wiki/Front_end_and_back_end), [JavaScript library](https://en.wikipedia.org/wiki/JavaScript_library) for building [user interfaces](https://en.wikipedia.org/wiki/User_interfaces) or UI components. It is maintained by [Facebook](https://en.wikipedia.org/wiki/Facebook) and a community of individual developers and companies. React can be used as a base in the development of [single-page](https://en.wikipedia.org/wiki/Single-page_application) or mobile applications. However, ReactJS is only concerned with state management and rendering that state to the [DOM](https://en.wikipedia.org/wiki/Document_Object_Model), so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

**Notable features of react:**

1. Component**:** React code is made of entities called components. Components can be rendered to a particular element in the [DOM](https://en.wikipedia.org/wiki/Document_Object_Model) using the React DOM library. When rendering a component, one can pass in values that are known as props.
2. VirtualDOM**:** Another notable feature is the use of a virtual [Document Object Model](https://en.wikipedia.org/wiki/Document_Object_Model), or virtual DOM. React creates an [in-memory](https://en.wikipedia.org/wiki/In-memory_processing) data-structure cache, computes the resulting differences, and then updates the browser's displayed DOM efficiently. This process is called reconciliation. This allows the programmer to write code as if the entire page is rendered on each change, while the React libraries only render subcomponents that actually change. This selective rendering provides a major performance boost. It saves the effort of recalculating the CSS style, layout for the page and rendering for the entire page.
3. JSX: JSX, or JavaScript [XML](https://en.wikipedia.org/wiki/XML), is an extension to the JavaScript language syntax. Similar in appearance to HTML, JSX provides a way to structure component rendering using syntax familiar to many developers. React components are typically written using JSX, although they do not have to be (components may also be written in pure JavaScript). JSX is similar to another extension syntax created by Facebook for [PHP](https://en.wikipedia.org/wiki/PHP) called [XHP](https://en.wikipedia.org/wiki/XHP).
4. Lifecycle methods:Lifecycle methods use a form of [hooking](https://en.wikipedia.org/wiki/Hooking) that allows the execution of code at set points during a component's lifetime.
5. shouldComponentUpdate allows the developer to prevent unnecessary re-rendering of a component by returning false if a render is not required.
6. componentDidMount is called once the component has "mounted" (the component has been created in the user interface, often by associating it with a [DOM](https://en.wikipedia.org/wiki/Document_Object_Model) node). This is commonly used to trigger data loading from a remote source via an [API](https://en.wikipedia.org/wiki/API).
7. componentWillUnmount is called immediately before the component is torn down or "unmounted". This is commonly used to clear resource-demanding dependencies to the component that will not simply be removed with the unmounting of the component (e.g., removing any setInterval() instances that are related to the component, or an "[eventListener](https://en.wikipedia.org/wiki/Event_(computing)" \o "Event (computing))" set on the "document" because of the presence of the component)
8. render is the most important lifecycle method and the only required one in any component. It is usually called every time the component's state is updated, which should be reflected in the user interface

**3.2 NodeJS**

Node.js is an [open-source](https://en.wikipedia.org/wiki/Open-source_software), [cross-platform](https://en.wikipedia.org/wiki/Cross-platform), [back-end](https://en.wikipedia.org/wiki/Front_end_and_back_end) [JavaScript](https://en.wikipedia.org/wiki/JavaScript) [runtime environment](https://en.wikipedia.org/wiki/Runtime_system) that runs on the [V8 engine](https://en.wikipedia.org/wiki/V8_(JavaScript_engine)) and executes JavaScript code outside a [web browser](https://en.wikipedia.org/wiki/Web_browser). Node.js lets developers use JavaScript to write command line tools and for [server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting)—running scripts server-side to produce [dynamic web page](https://en.wikipedia.org/wiki/Dynamic_web_page) content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying [web-application](https://en.wikipedia.org/wiki/Web_application) development around a single programming language, rather than different languages for server-side and client-side scripts.

Though .js is the standard [filename extension](https://en.wikipedia.org/wiki/Filename_extension) for JavaScript code, the name "Node.js" doesn't refer to a particular file in this context and is merely the name of the product. Node.js has an [event-driven architecture](https://en.wikipedia.org/wiki/Event-driven_architecture) capable of [asynchronous I/O](https://en.wikipedia.org/wiki/Asynchronous_I/O). These design choices aim to optimize [throughput](https://en.wikipedia.org/wiki/Throughput) and [scalability](https://en.wikipedia.org/wiki/Scalability) in web applications with many input/output operations, as well as for [real-time Web](https://en.wikipedia.org/wiki/Real-time_Web) applications (e.g., [real-time communication](https://en.wikipedia.org/wiki/Real-time_communication) programs and [browser games](https://en.wikipedia.org/wiki/Browser_game)).

The Node.js [distributed development](https://en.wikipedia.org/wiki/Distributed_development) project was previously governed by the Node.js Foundation, and has now merged with the [JS Foundation](https://en.wikipedia.org/wiki/JS_Foundation) to form the [OpenJS Foundation](https://en.wikipedia.org/wiki/OpenJS_Foundation), which is facilitated by the [Linux Foundation](https://en.wikipedia.org/wiki/Linux_Foundation)'s Collaborative Projects program.

**3.3 Django**

Django is a [Python](https://en.wikipedia.org/wiki/Python_(programming_language))-based [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source_software) [web framework](https://en.wikipedia.org/wiki/Web_framework) that follows the model-template-views (MTV) [architectural pattern](https://en.wikipedia.org/wiki/Architectural_pattern_(computer_science)). It is maintained by the [Django Software Foundation](https://en.wikipedia.org/wiki/Django_Software_Foundation) (DSF), an American independent organization established as a [501(c)(3)](https://en.wikipedia.org/wiki/501(c)(3)) non-profit.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes [reusability](https://en.wikipedia.org/wiki/Reusability) and "pluggability" of components, less code, low coupling, rapid development, and the principle of [don't repeat yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself). Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative [create, read, update and delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete) interface that is generated dynamically through [introspection](https://en.wikipedia.org/wiki/Type_introspection) and configured via admin models.

**3.4 MongoDB**

MongoDB is a [source-available](https://en.wikipedia.org/wiki/Source-available) [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) [document-oriented database](https://en.wikipedia.org/wiki/Document-oriented_database) program. Classified as a [NoSQL](https://en.wikipedia.org/wiki/NoSQL) database program, MongoDB uses [JSON](https://en.wikipedia.org/wiki/JSON)-like documents with optional [schemas](https://en.wikipedia.org/wiki/Database_schema). MongoDB is developed by [MongoDB Inc.](https://en.wikipedia.org/wiki/MongoDB_Inc.) and licensed under the [Server Side Public License](https://en.wikipedia.org/wiki/Server_Side_Public_License) (SSPL).

1. Ad-hoc queries: MongoDB supports field, [range query](https://en.wikipedia.org/wiki/Range_query_(database)), and [regular-expression](https://en.wikipedia.org/wiki/Regular_expression) searches. Queries can return specific fields of documents and also include user-defined [JavaScript](https://en.wikipedia.org/wiki/JavaScript) functions. Queries can also be configured to return a random sample of results of a given size.
2. Indexing: Fields in a MongoDB document can be indexed with primary and secondary indices.
3. Replication: MongoDB provides high availability with replica sets. A replica set consists of two or more copies of the data. Each replica-set member may act in the role of primary or secondary replica at any time. All writes and reads are done on the primary replica by default. Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically conducts an election process to determine which secondary should become the primary. Secondaries can optionally serve read operations, but that data is only eventually consistent by default. If the replicated MongoDB deployment only has a single secondary member, a separate [daemon](https://en.wikipedia.org/wiki/Daemon_(computing)) called an *arbiter* must be added to the set. It has a single responsibility, which is to resolve the election of the new primary. As a consequence, an idealized distributed MongoDB deployment requires at least three separate servers, even in the case of just one primary and one secondary.
4. Load balancing: MongoDB scales horizontally using [sharding](https://en.wikipedia.org/wiki/Sharding). The user chooses a shard key, which determines how the data in a collection will be distributed. The data is split into ranges (based on the shard key) and distributed across multiple shards. (A shard is a master with one or more replicas.). Alternatively, the shard key can be hashed to map to a shard – enabling an even data distribution. MongoDB can run over multiple servers, [balancing the load](https://en.wikipedia.org/wiki/Load_balancing_(computing)) or duplicating data to keep the system up and running in case of hardware failure.
5. File storage: MongoDB can be used as a [file system](https://en.wikipedia.org/wiki/File_system), called [GridFS](https://en.wikipedia.org/w/index.php?title=GridFS&action=edit&redlink=1), with load balancing and data replication features over multiple machines for storing files. This function, called [grid file system](https://en.wikipedia.org/wiki/Grid_file_system), is included with MongoDB drivers. MongoDB exposes functions for file manipulation and content to developers. GridFS can be accessed using mongofiles utility or plugins for [Nginx](https://en.wikipedia.org/wiki/Nginx) and [lighttpd](https://en.wikipedia.org/wiki/Lighttpd). GridFS divides a file into parts, or chunks, and stores each of those chunks as a separate document.
6. Aggregation: MongoDB provides three ways to perform aggregation: the aggregation pipeline, the map-reduce function, and single-purpose aggregation methods. [Map-reduce](https://en.wikipedia.org/wiki/Map-reduce) can be used for batch processing of data and aggregation operations. But according to MongoDB's documentation, the Aggregation Pipeline provides better performance for most aggregation operations. The aggregation framework enables users to obtain the kind of results for which the [SQL](https://en.wikipedia.org/wiki/SQL) GROUP BY clause is used. Aggregation operators can be strung together to form a pipeline – analogous to [Unix pipes](https://en.wikipedia.org/wiki/Pipeline_(Unix)). The aggregation framework includes the $lookup operator which can join documents from multiple collections, as well as statistical operators such as standard deviation. Server-side JavaScript execution: JavaScript can be used in queries, aggregation functions (such as [MapReduce](https://en.wikipedia.org/wiki/MapReduce)), and sent directly to the database to be executed.
7. Capped collections: MongoDB supports fixed-size collections called capped collections. This type of collection maintains insertion order and, once the specified size has been reached, behaves like a [circular queue](https://en.wikipedia.org/wiki/Circular_queue).
8. Transactions: MongoDB claims to support multi-document ACID transactions since the 4.0 release in June 2018. This claim was found to not be true as MongoDB violates [snapshot isolation](https://en.wikipedia.org/wiki/Snapshot_isolation).

**3.5 Machine learning (Content based Recommendation System)**

**Machine learning (ML):** ML is the study of computer [algorithms](https://en.wikipedia.org/wiki/Algorithm) that improve automatically through experience and by the use of data. It is seen as a part of [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence). Machine learning algorithms build a model based on sample data, known as "[training data](https://en.wikipedia.org/wiki/Training_data)", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, [email filtering](https://en.wikipedia.org/wiki/Email_filtering), and [computer vision](https://en.wikipedia.org/wiki/Computer_vision), where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

A subset of machine learning is closely related to [computational statistics](https://en.wikipedia.org/wiki/Computational_statistics), which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of [mathematical optimization](https://en.wikipedia.org/wiki/Mathematical_optimization) delivers methods, theory and application domains to the field of machine learning. [Data mining](https://en.wikipedia.org/wiki/Data_mining) is a related field of study, focusing on [exploratory data analysis](https://en.wikipedia.org/wiki/Exploratory_data_analysis) through [unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning). In its application across business problems, machine learning is also referred to as [predictive analytics](https://en.wikipedia.org/wiki/Predictive_analytics).

**Recommender systems**: Recommender systems are the systems that are designed to recommend things to the user based on many different factors. These systems predict the most likely product that the users are most likely to purchase and are of interest to. Companies like [Netflix](https://www.analyticssteps.com/blogs/using-data-handling-and-digital-marketing-maximise-customer-experience-netflix-case-study), Amazon, etc. use recommender systems to help their users to identify the correct product or movies for them.

The recommender system deals with a large volume of information present by filtering the most important information based on the data provided by a user and other factors that take care of the user’s preference and interest. It finds out the match between user and item and imputes the similarities between users and items for recommendation.

Both the users and the services provided have benefited from these kinds of systems. The quality and decision-making process has also improved through these kinds of systems.

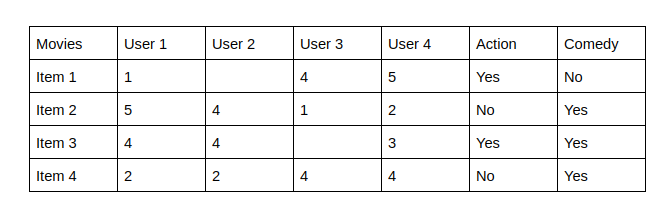
**Content Based Recommendation System:** Content-based filtering uses item features to recommend other items similar to what the user likes, based on their previous actions or explicit feedback.

To demonstrate content-based filtering, let’s hand-engineer some features for the Google Play store. The following figure shows a feature matrix where each row represents an app and each column represents a feature. Features could include categories (such as Education, Casual, Health), the publisher of the app, and many others. To simplify, assume this feature matrix is binary: a non-zero value means the app has that feature.

You also represent the user in the same feature space. Some of the user-related features could be explicitly provided by the user. For example, a user selects "Entertainment apps" in their profile. Other features can be implicit, based on the apps they have previously installed. For example, the user installed another app published by Science R Us.

The model should recommend items relevant to this user. To do so, you must first pick a similarity metric (for example, dot product). Then, you must set up the system to score each candidate item according to this similarity metric. Note that the recommendations are specific to this user, as the model did not use any information about other users.

Content-Based recommender system tries to guess the features or behaviour of a user given the item’s features, he/she reacts positively to.



*Figure 3*

The last two columns Action and Comedy Describe the Genres of the movies. Now, given these genres, we can know which users like which genre, as a result, we can obtain features corresponding to that particular user, depending on how he/she reacts to movies of that genre.

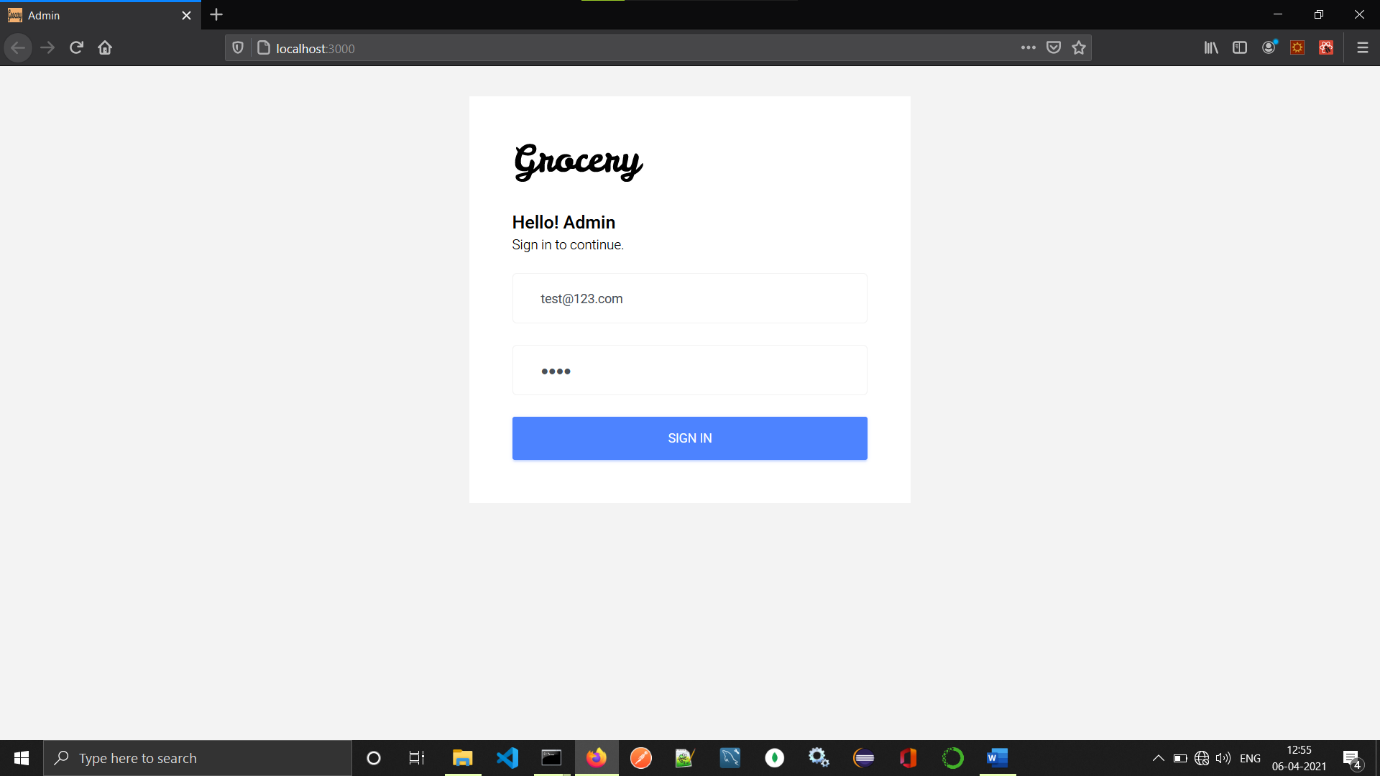
Once, we know the likings of the user we can embed him/her in an embedding space using the feature vector generated and recommend him/her according to his/her choice. During recommendation, the similarity metrics (We will talk about it in a bit) are calculated from the item’s feature vectors and the user’s preferred feature vectors from his/her previous records. Then, the top few are recommended. Content-based filtering does not require other users' data during recommendations to one user.

**CHAPTER 4**

**RESULT**

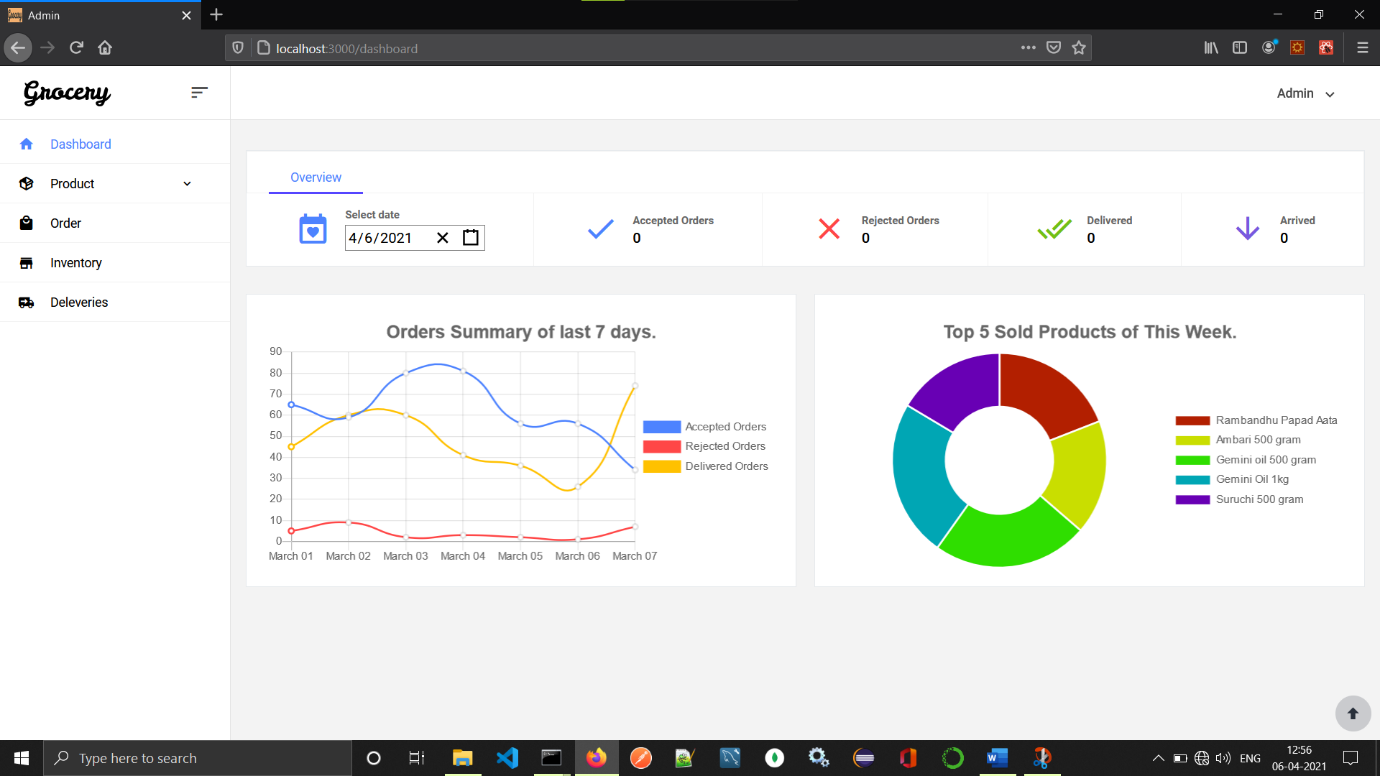
**4.1 ADMIN SITE SNAPSHOTS**

1. Login Window



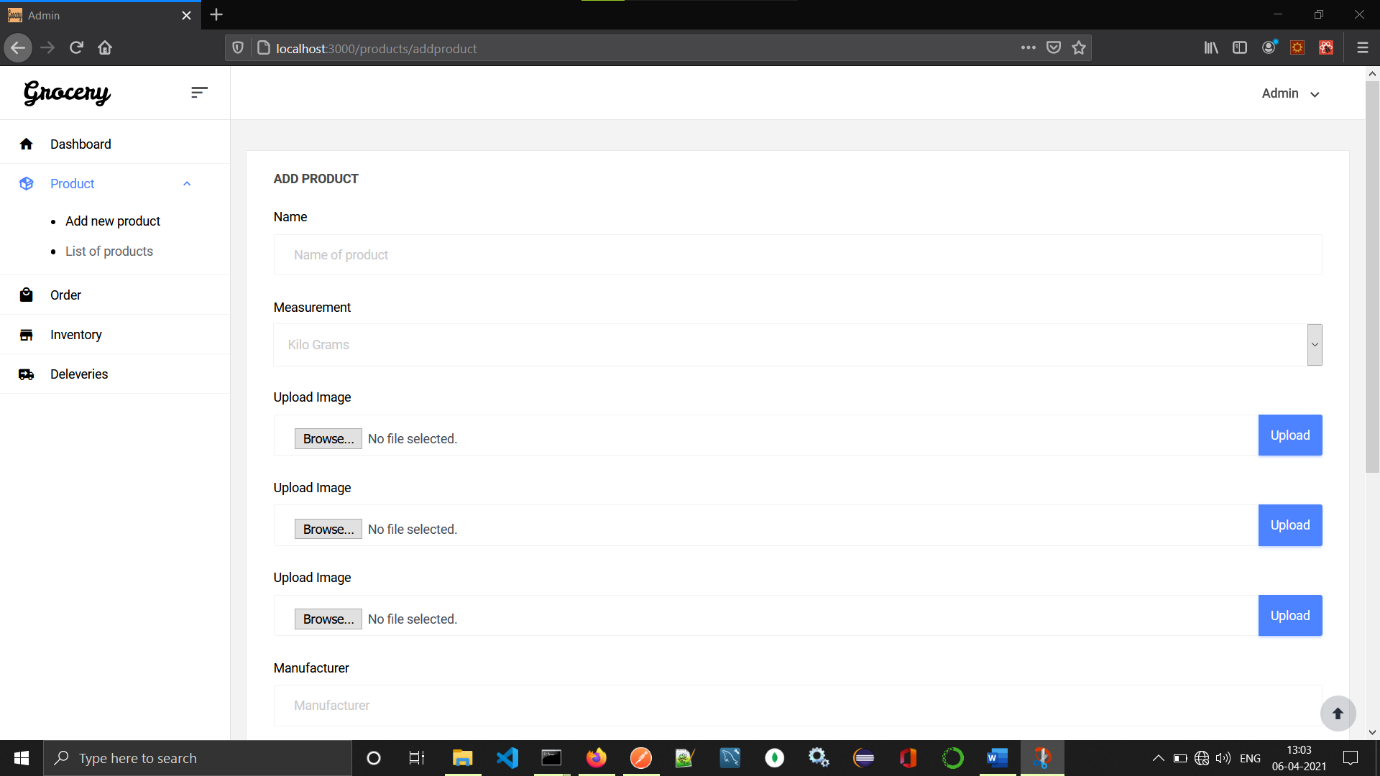
This is login window for admin. It requires registered admin email and password.

2. Dashboard

****

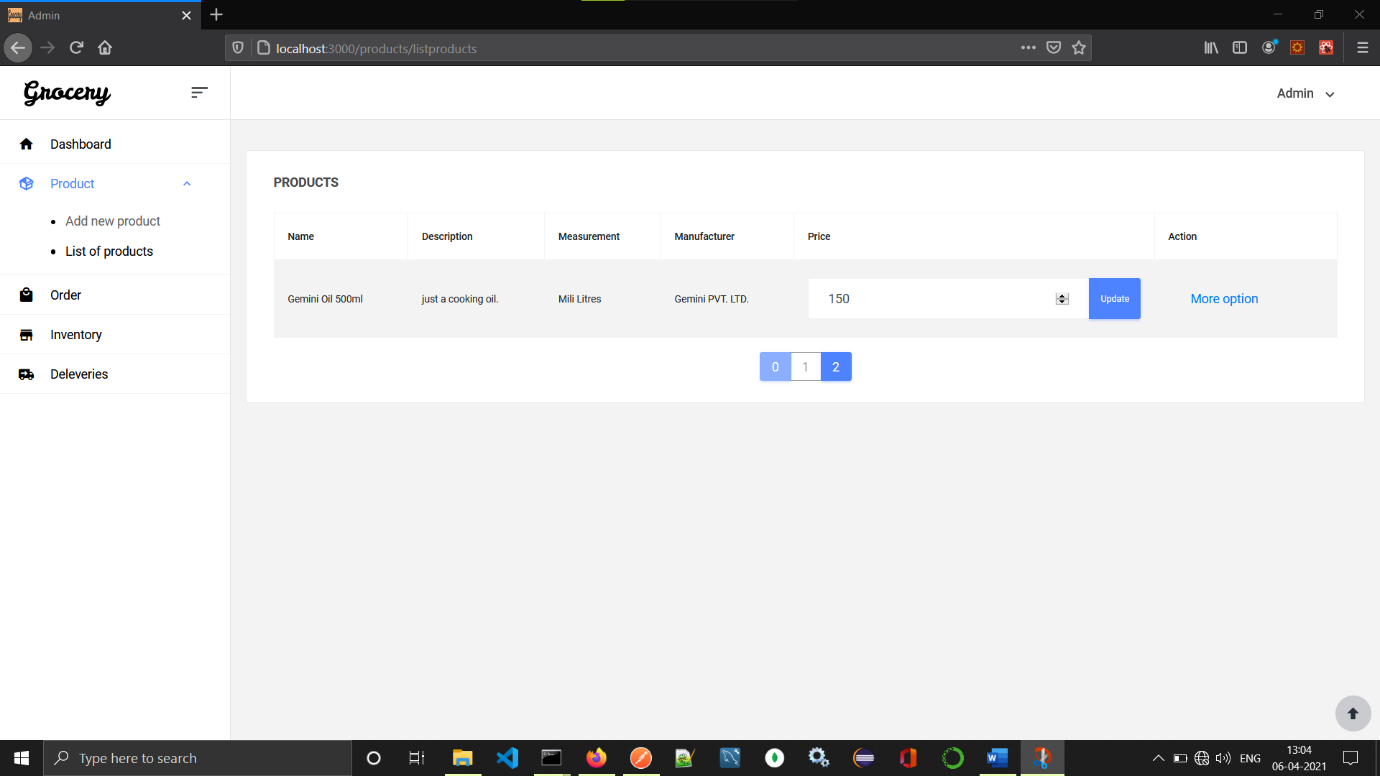
This is Dashboard of admin panel. It mainly consists pie chart line chart and the date wise report catalog. In Overview menu we can see the information about todays orders. It shows number of accepted orders, Rejected Orders, Delivered Orders, And Arrived Orders. And next is graph component it shows summary of last seven days in graphical manner.

* 1. Add Product Form

****

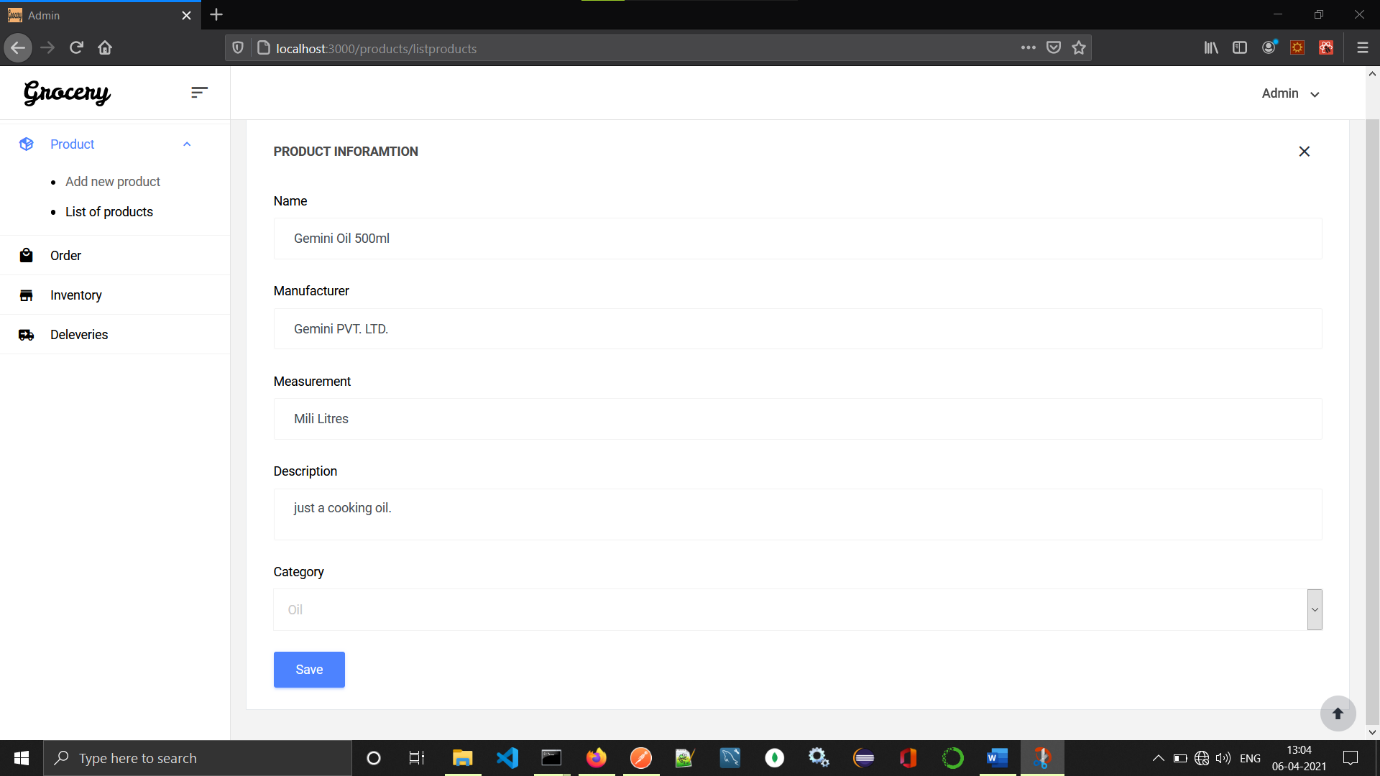
Add product form will use for adding new product in shop. In this project the product has following attributes, Name, Measurement, Images, Manufacturer, Keywords, Category. Plus, we are having autogenerated id for each product.

* 1. List of Products

****

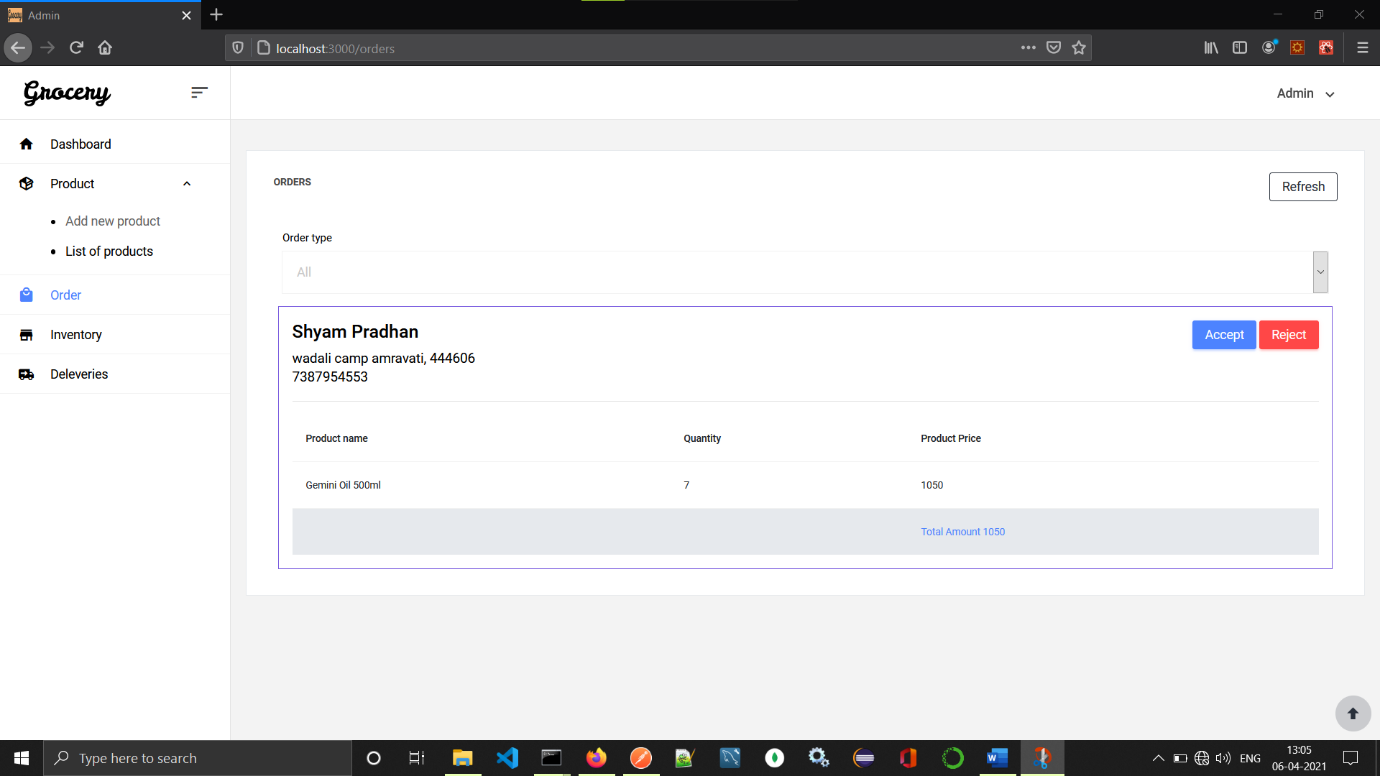
It has List of products form this list one can edit the price of product and even edit the product by clicking more option button.

* 1. Edit Product Form

****

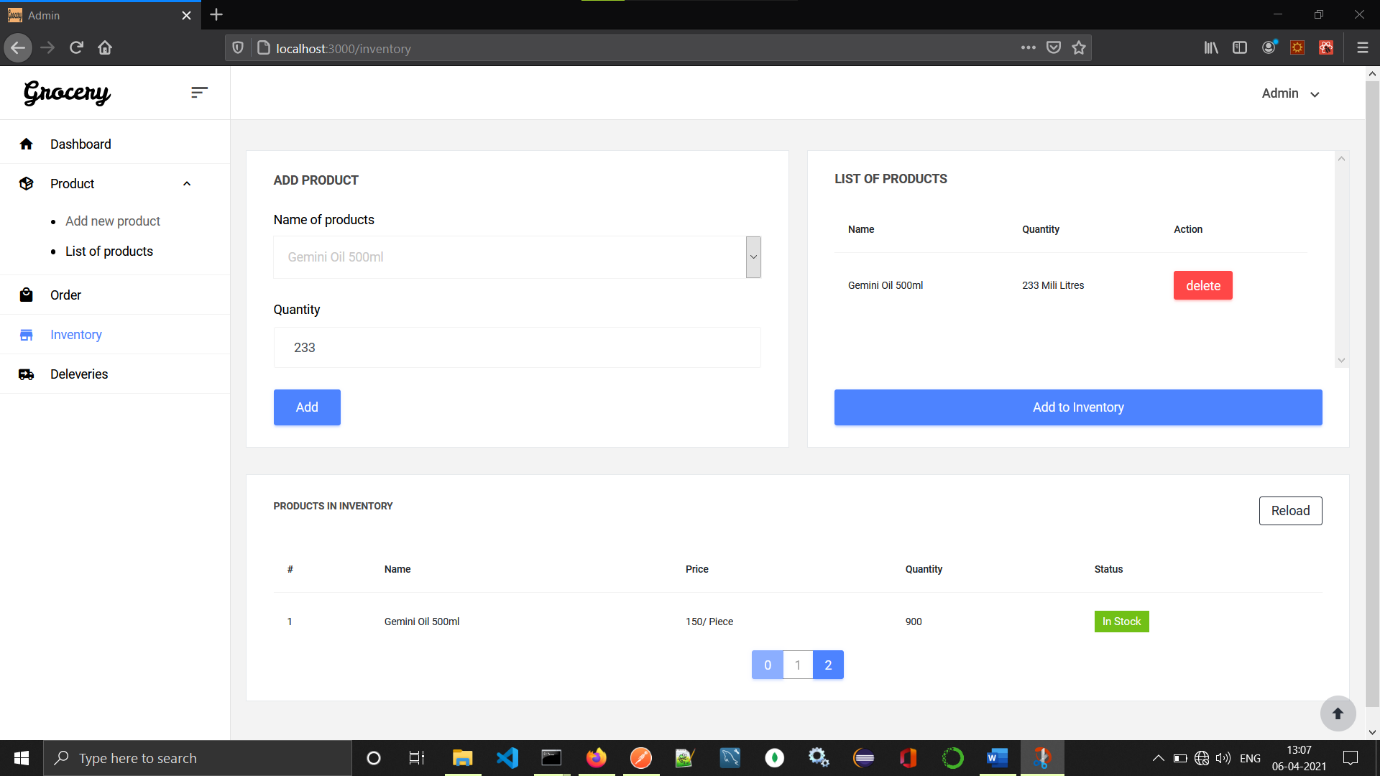
After clicking more options from list, we will get this form in which we can see the information of it. And product information can be edited from this form.

* 1. List of Orders

****

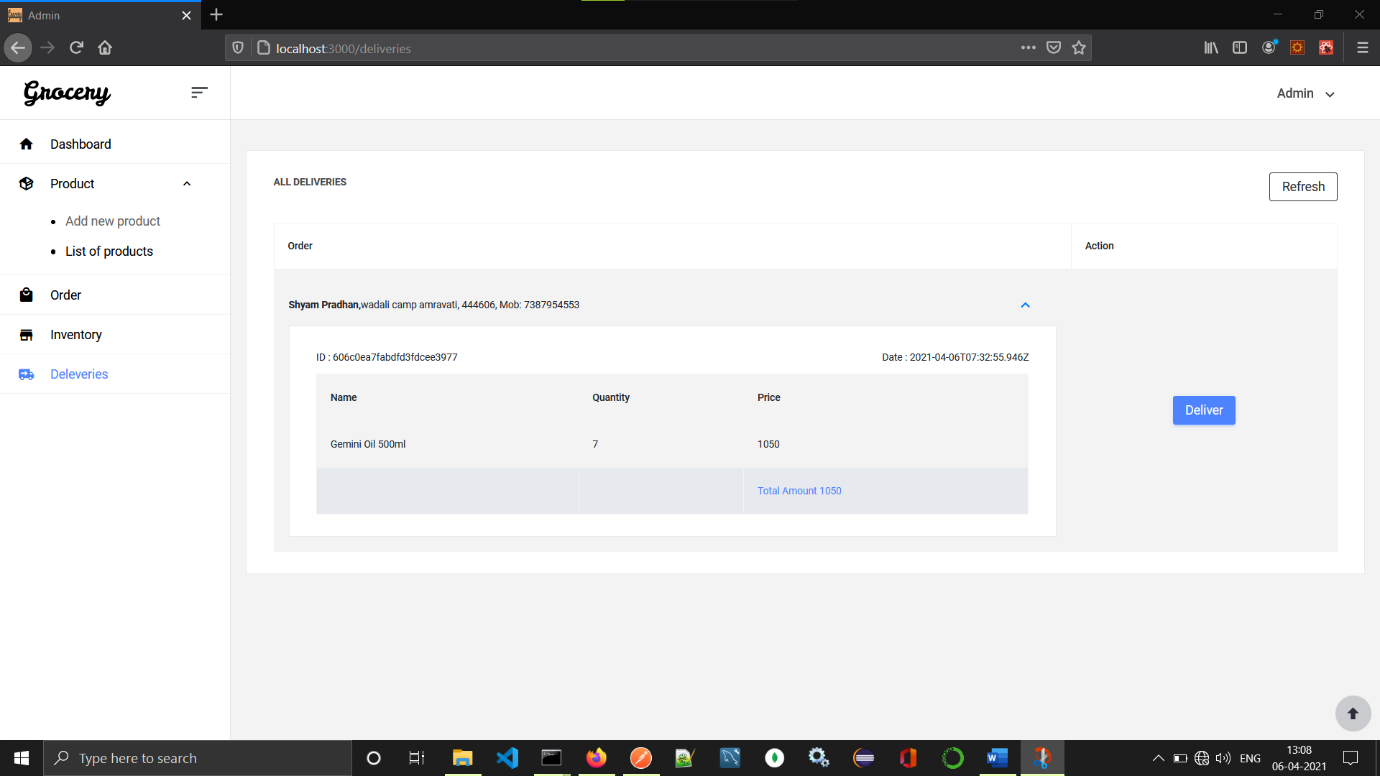
This page consists list of all the todays orders. Order having different status like, Arrived, Being Delivered, Rejected, Delivered. From this admin can accept reject order or even track the orders. It also consists the filter which allow admin to filter orders.

* 1. Inventory

****

Adding new items to inventory can be handle by this module. It shows status of each product whether it is in stock or not is provided by this module.

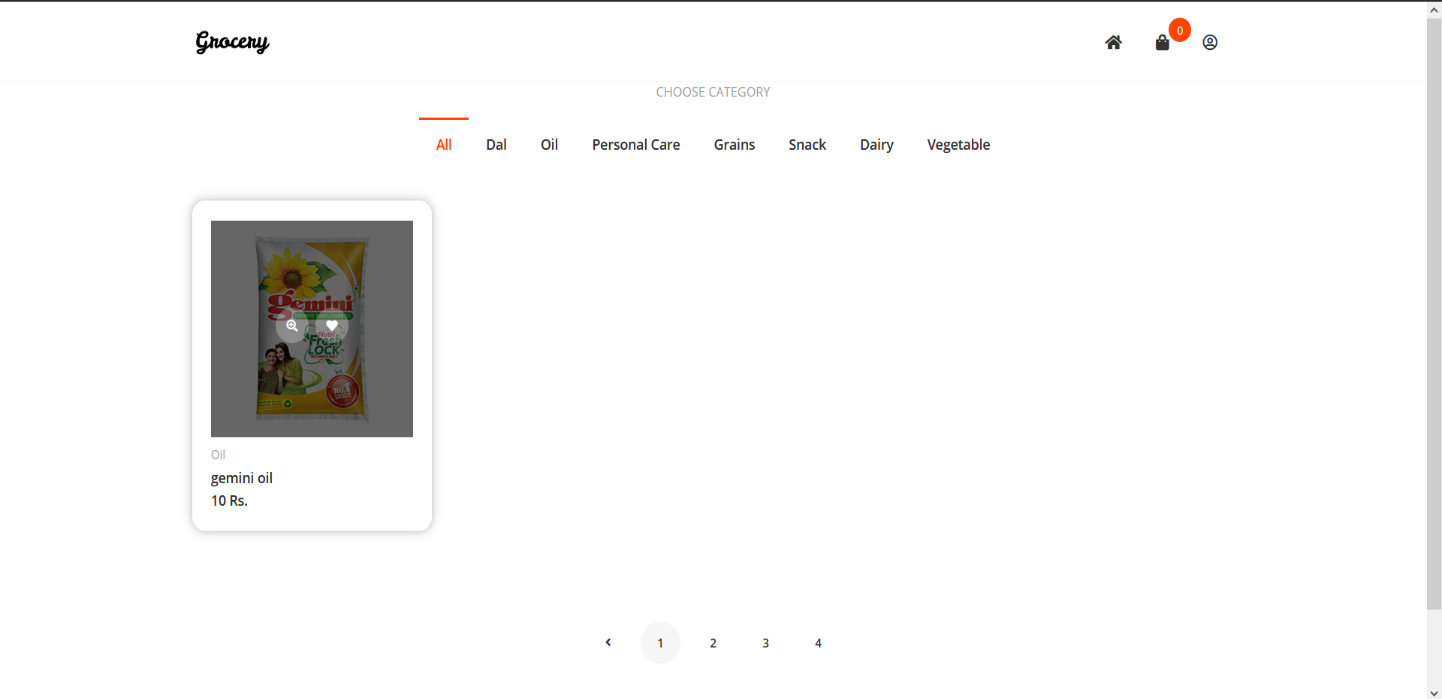
* 1. Deliveries

****

This page mainly shows the deliveries that has to delivered to customer. Deliver button will be clicked when item is delivered successfully to the customer.

**4.2 Customer Site Snapshots**

1 Home Page

****

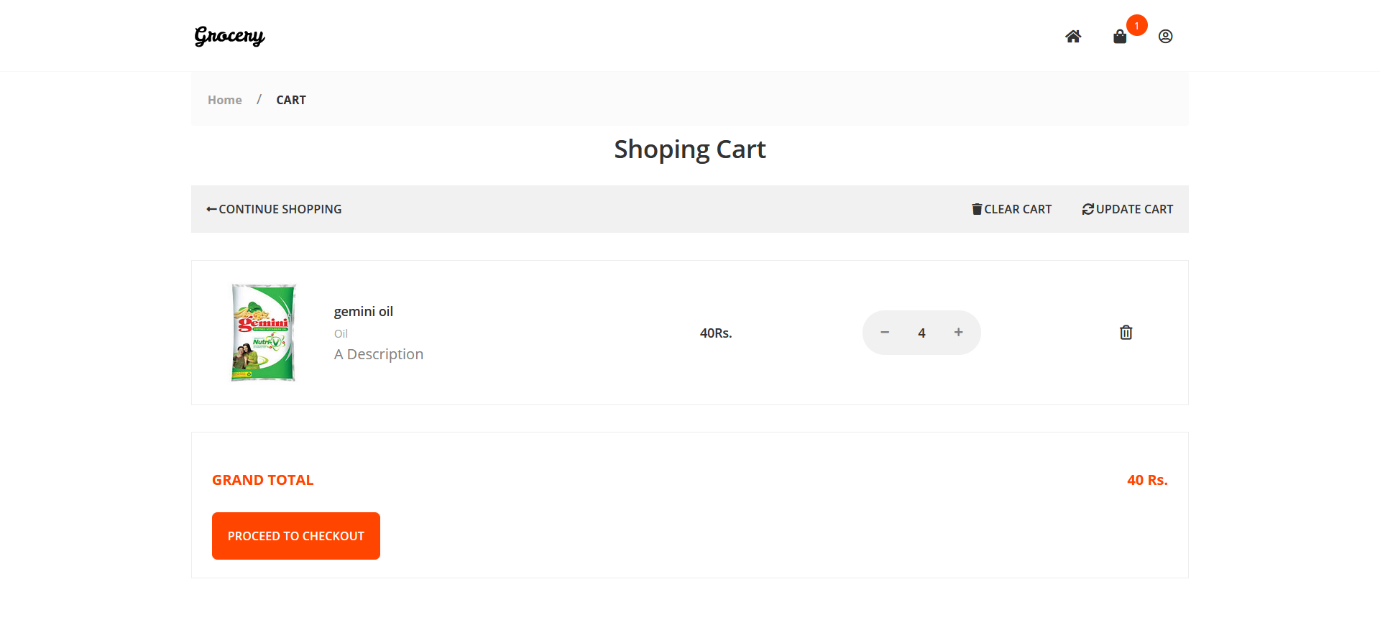
This is homepage for the shop. This mainly consists list of the products. And one can search product and it also consist category from which customer can search products.

2 Add to Cart



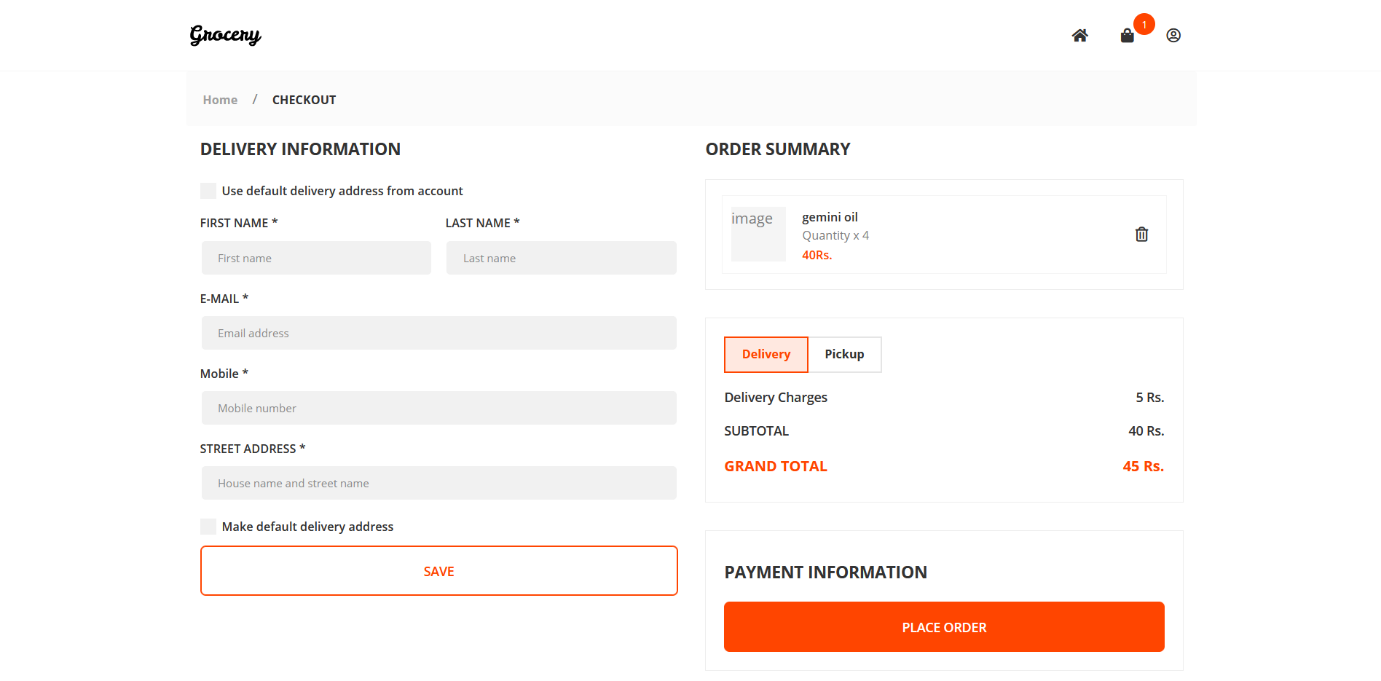
This popup shows description of product name of product and add cart button. Add to cart button will add this product to cart with quantity and same will be shown in checkout.

3 Shopping Cart



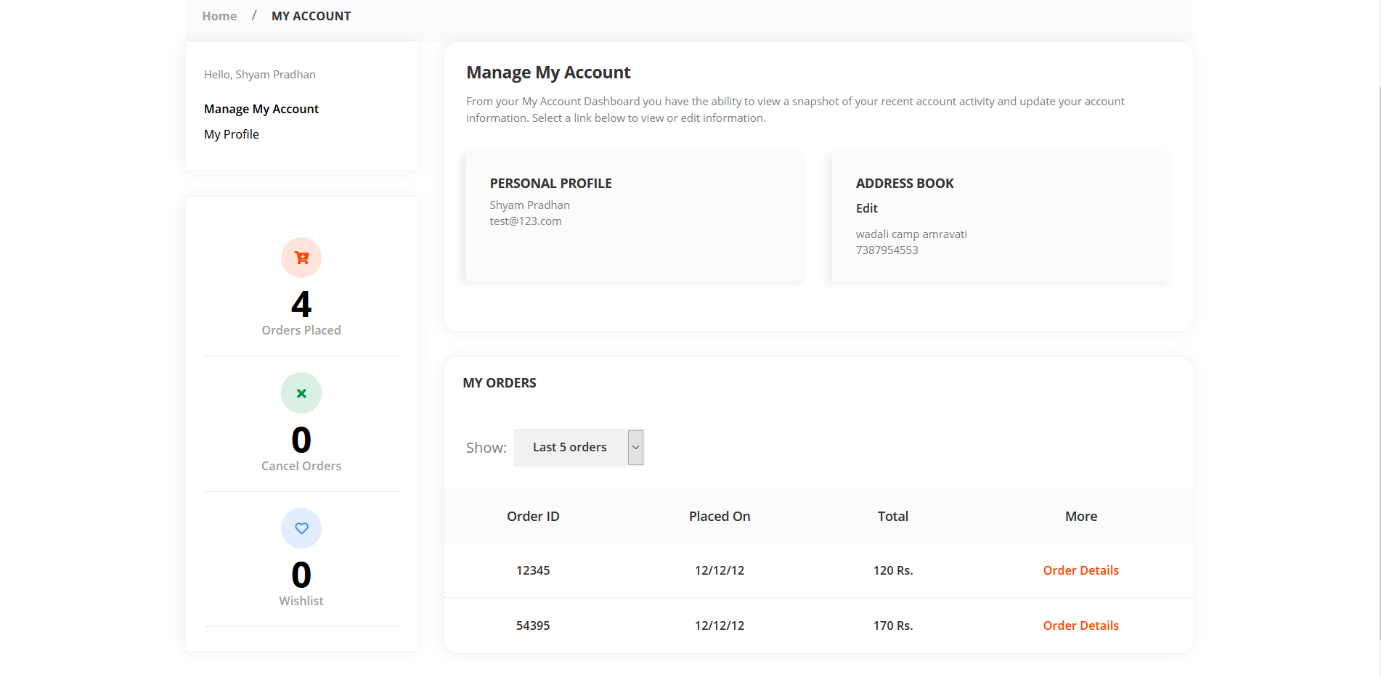
This page will consist list of the product in the cart. From which cart can be updated and cart can be cleared. From this page quantity of product in cart can be updated. And finally we can see the grand total. It has button for going to checkout page.

4 Checkout



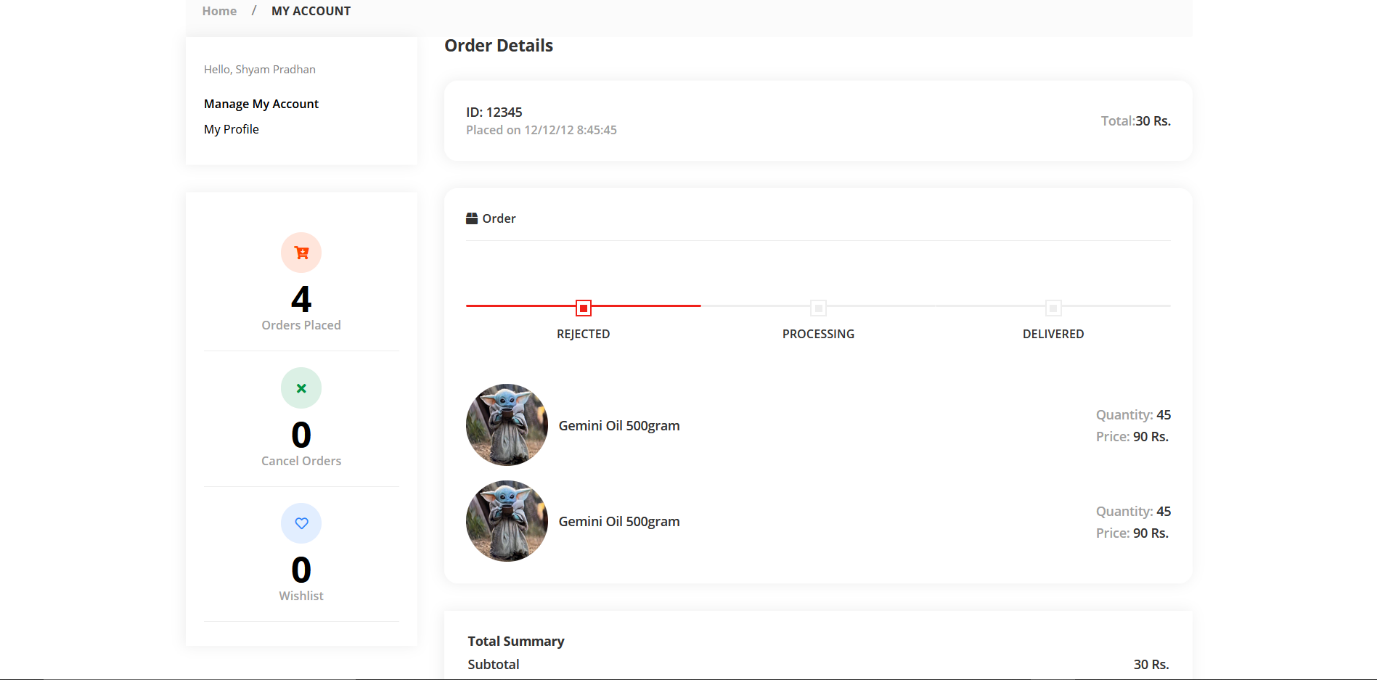
This is checkout page which consist the address info and user info in one side which will previously loaded (Not Completed). And in right side we can again see the list of products in cart we can also delete the product in checkout. Now user needs to choose delivery or pickup option delivery will add extra charges of deliver and pickup don’t have any.

5 My Account



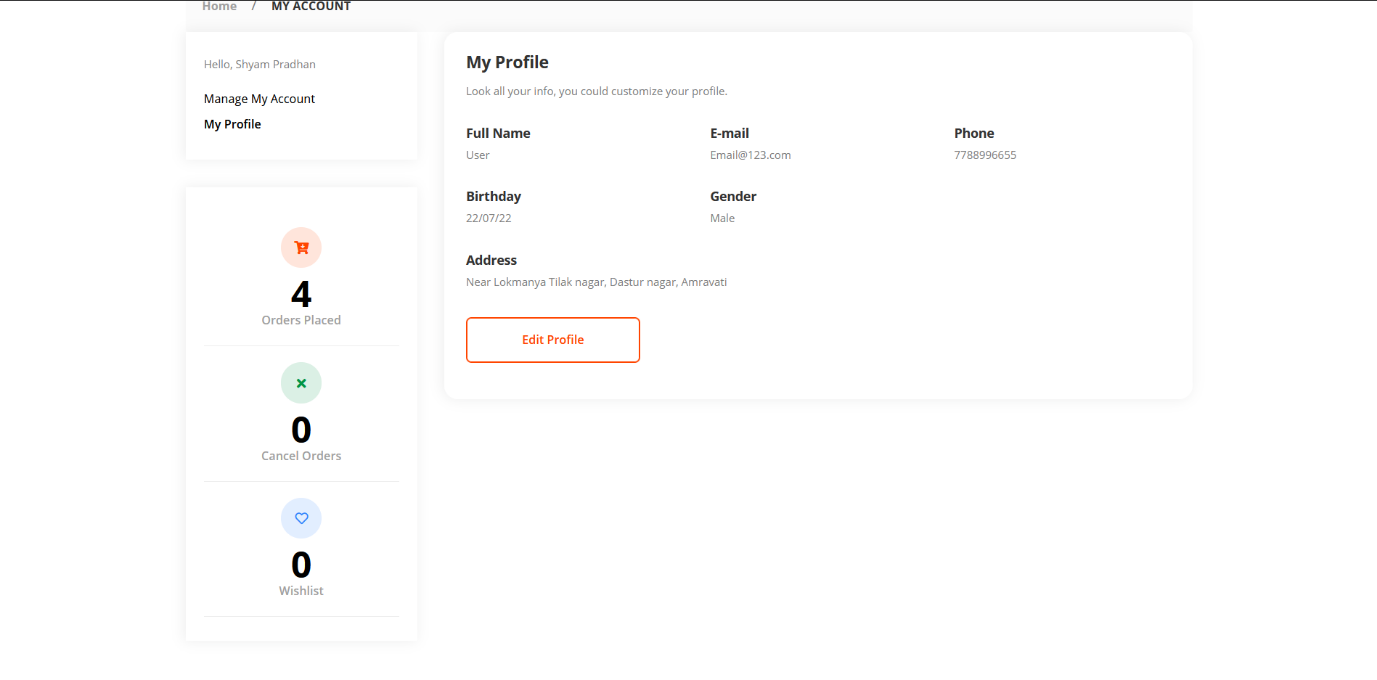
This page consists account info such name and address also it has list of orders thar are placed before. It also gives summary of orders in left side.

6 Order Details



This is order details page which has info about the orders by clicking order details button in list we can get to it. Progress bar is given to show progress of orders.

7 My Profile



This is My Profile which consists basic information about profile can also be edited from this page.

**4.2 FUTURE SCOPE**

This project can easily used and managed by one who have some basic understanding in the mobile and internaet technology. One can easily manage the grocery product through this app. Also it would be easily used by the grocery shop owners to place the orders. This application can easily implemented in various situations. We can add new features as and when we require. There is flexibility in all the models. The reusability can be achived as two or more grocery shop can use this simulteniously. As of now there is no notification system is implemented either in customer or admin site so that we can add those features in it. The UI/UX is quite good in this application but it has some issues in it like real time data reload is biggest problem from what this app is suffering so we can add real time socket connection with server. Back-end of app has some issues like load-balacing, communication so these can also be solved.

More modules like online payment, smart inventory management by using machine learning algorithm. Also it can be easily integrated with cloud. Also it can be deployed on cloud by using dockers and containers.

Reusability is possible as and when we require in this application. We can update it future version of this application.reusability of software can reduce design,coding efforts. This app can be configured for other local shooping stores. Like cloth stors, general stores, even medical store can be us this app by changing some modules.

**4.3 CONCLUSION**

Grocery store automation system is an web based ecommerce solution for grocery stores. It can be easily used by customer and admin to manage and order the product online. This can best solution for buying products online. In this pandemic solution no one wants/allowed to go out of the house. The main reason behind making this project is to make the system to start the local grocery shop online. In this internet era not only big enterprizes but small local grocery shop should be online. There must online solution to sell/buy product online. This automation system provide greate solution to manage inventory and product stock. Also it can proper report of product selling.

**CHAPTER 5**

**REFERENCES**

* [*https://en.wikipedia.org/wiki/Django\_(web\_framework)*](https://en.wikipedia.org/wiki/Django_(web_framework))
* [*https://en.wikipedia.org/wiki/Machine\_learning*](https://en.wikipedia.org/wiki/Machine_learning)
* [*https://en.wikipedia.org/wiki/MongoDB*](https://en.wikipedia.org/wiki/MongoDB)
* [*https://en.wikipedia.org/wiki/Node.js*](https://en.wikipedia.org/wiki/Node.js)
* [*https://en.wikipedia.org/wiki/React\_(JavaScript\_library)*](https://en.wikipedia.org/wiki/React_(JavaScript_library))
* [*https://www.kdnuggets.com/2019/09/machine-learning-recommender-systems.html*](https://www.kdnuggets.com/2019/09/machine-learning-recommender-systems.html)
* [*https://scikit-learn.org/stable/modules/generated/sklearn.feature\_extraction.text.TfidfVectorizer.html*](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html)