Online Grocery System

A PROJECT REPORT

Submitted By

AADITYA SINGH
(2000290140001)
Jai Pratap Singh
(2000290140053)
KSHITIZ PANDEY
(2000290140060)
OMVEER SINGH
(2000290140082)

Submitted in partial fulfilment of the Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of
Ms. Shalika Arora
Assistant Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

(JAN 2022)

CERTIFICATE

Certified that **Aaditya Singh (2000290140001), Jai Pratap Singh (2000290140053), Kshitiz Pandey (2000290140060), Omveer Singh (2000290140082),** have carried out the project work having "**Title of Report- Online Grocery System**" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date: 12-01-2022

Aaditya Singh
(2000290140001)
Jai Pratap Singh
(2000290140053)
Kshitiz Pandey
(2000290140060)
Omveer Singh
(2000290140082)

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

Date: 12-01-2022

Ms. Shalika Arora
Assistant Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner **Signature of External Examiner**

Dr. Ajay Shrivastava
Head, Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

Objective of proposed system, is to provide Online Grocery Shopping solution to consumers and vendors. It will automate some of the basic operations of an online store. Scope would be to provide basic functionalities using a web application so that those manual process can be automated. It will include to provide administration access to vendors and admins and user specific access to customers.

This project is an attempt to provide the advantages of online shopping to customers of a real shop. It helps buying the products in the shop anywhere through internet by using an internet. Thus the customer will get the service of online shopping and home delivery from his favorite shop. This system can be implemented to any shop in the locality or to multinational branded shops having retail outlet chains.

If shops are providing an online portal where their customers can enjoy easy shopping from anywhere, the shops won't be losing any more customers to the trending online shops.

Proposed system will ease the shopping operations for customer of online store. It will provide vendor or administration functionality to manage categories and products. Consumer will be able to browse and search products under different categories. Selected products or items selected for purchase would be added into the virtual shopping cart. Which can be managed separately by customer. It can be examined at any time by customer for selected products, their quantity & price.

LIST OF FIGURES

ACKNOWLEDGEMENTS

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Ms. Shalika Arora** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Ajay Kumar Shrivastava**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

AADITYA SINGH
(2000290140001)
Jai Pratap Singh
(2000290140053)
Kshitiz Pandey
(2000290140060)
OMVEER SINGH
(2000290140082)

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

This project is a web based shopping system for an existing shop. The project objective is to deliver the online shopping application into android platform.

Online shopping is the process whereby consumers directly buy goods or services from a seller in real-time, without an intermediary service, over the Internet. It is a form of electronic commerce. This project is an attempt to provide the advantages of online shopping to customers of a real shop. It helps buying the products in the shop anywhere through internet by using an android device. Thus the customer will get the service of online shopping and home delivery from his favorite shop

1.1 PROJECT OBJECTIVE:

The objective of the project is to make an application in android platform to purchase items in an existing shop. In order to build such an application complete web support need to be provided. A complete and efficient web application which can provide the online shopping experience is the basic objective of the project. The web application can be implemented in the form of an android application with web view.

1.2 PROJECT OVER VIEW:

The central concept of the application is to allow the customer to shop virtually using the Internet and allow customers to buy the items and articles of their desire from the store. The information pertaining to the products are stores on an RDBMS at the server side (store).

The Server process the customers and the items are shipped to the address submitted by them. The application was designed into two modules first is for the customers who wish to buy the articles. Second is for the storekeepers who maintains and updates the information pertaining to the articles and those of the customers. The end user of this product is a departmental store where the application is hosted on the web and the administrator maintains the database. The application which is deployed at the customer database, the details of the items are brought forward from the database for the customer view based on the selection through the menu and the database of all the products are updated at the end of each transaction. Data entry into the application can be done through various screens designed for various levels of users. Once the authorized personnel feed the relevant data into the system, several reports could be generated as per the security.

1.3 PROJECT SCOPE:

This system can be implemented to any shop in the locality or to multinational branded shops having retail outlet chains. The system

recommends a facility to accept the orders 24*7 and a home delivery system which can make customers happy.

If shops are providing an online portal where their customers can enjoy easy shopping from anywhere, the shops won't be losing any more customers to the trending online shops such as flipcart or ebay. Since the application is available in the Smartphone it is easily accessible and always available.

1.4 STUDY OF THE SYSTEM

1.4.1 MODULES:

The system after careful analysis has been identified to be presented with the following modules and roles.

The modules involved are:

- Administrator
- Moderators
- Users

1.4.1.1 ADMINISTRATOR:

The administrator is the super user of this application. Only admin have access into this admin page. Admin may be the owner of the shop. The administrator has all the information about all the users and about all products.

This module is divided into different sub-modules.

1. Manage Moderators
2. Manage Products
3. Manage Users
4. Manage Orders

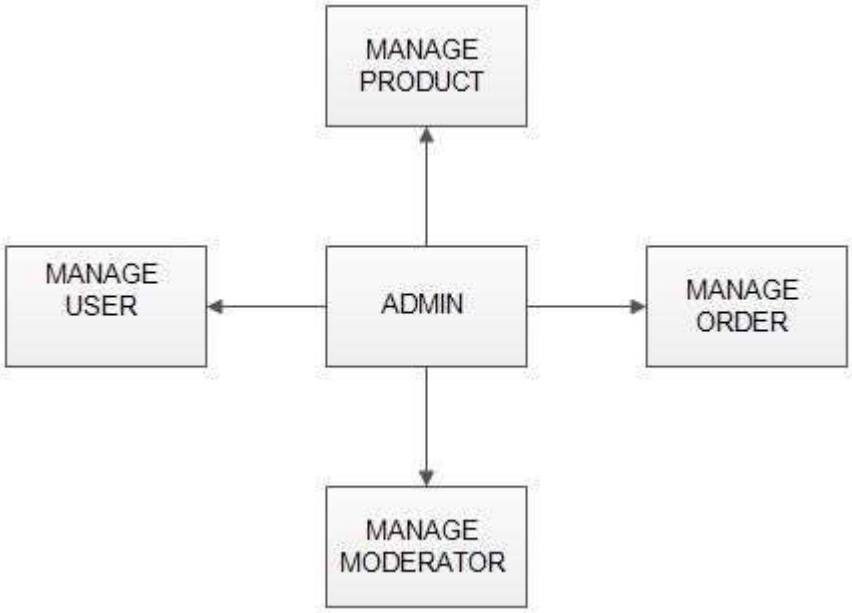


Fig 1.1: Admin module 1

MANAGE MODERATOR

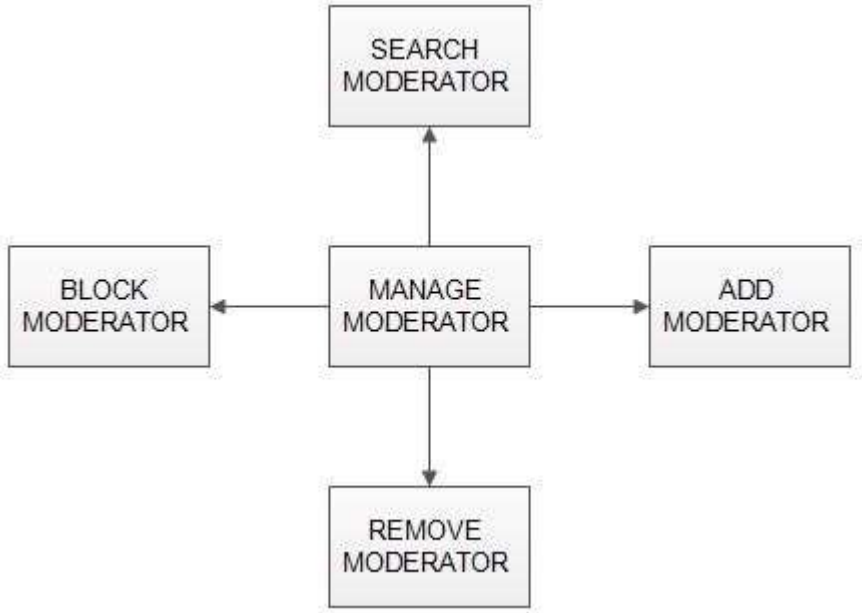


Fig 1.2: Manage Moderator

- Add Moderator

Only admin is having the privilege to add a moderator. A moderator can be considered as a staff who manages the orders or owner of a group of products.

- Block moderator

  Admin can restrict a moderator from managing the orders by blocking them. Admin can unblock a blocked user if needed.

- Remove Moderator

Admin has privilege to delete a moderator who was added.

- Search moderator:

All existing moderators can be viewed by the administrator as a list. If there is number of moderators and admin need to find one of them, the admin can search for a moderator by name.

MANAGE PRODUCTS

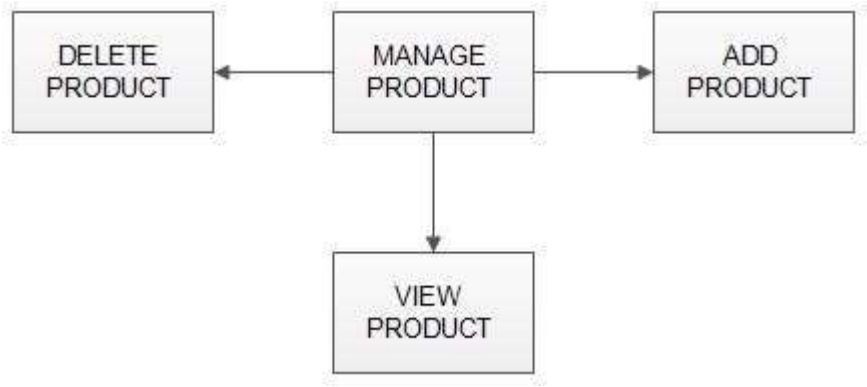


Fig 1.3: Manage Products 1

- Add Products

The shopping cart project contains different kind of products. The products can be classified into different categories by name. Admin can add new products into the

existing system with all its details including an image.

- Delete Products

Administrator can delete the products based on the stock of that particular product.

- Search products

  Admin will have a list view of all the existing products. He can also search

for a particular product by name.

MANAGE USER

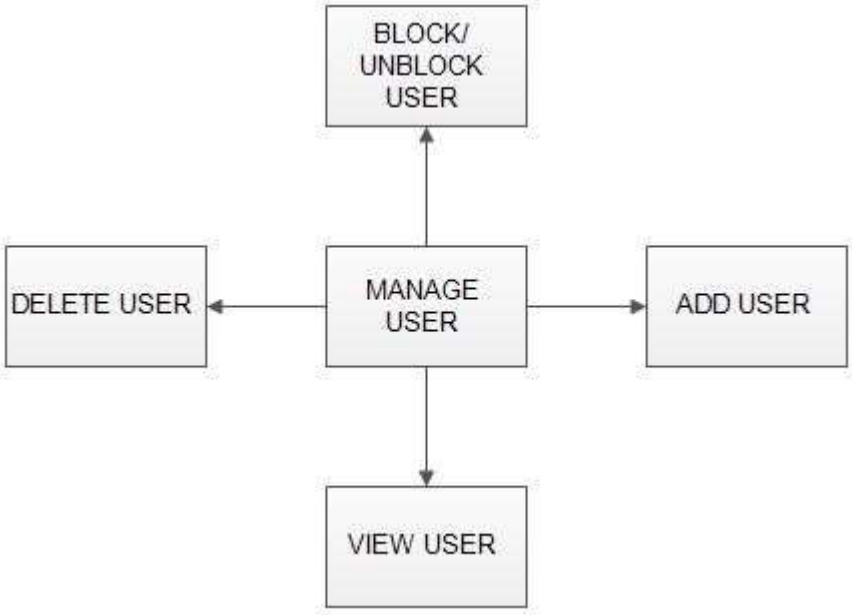


Fig 1.4: Manage User

- View Users

The admin will have a list view of all the users registered in the system. Admin can view all the details of each user in the list except password.

- Add Users

Admin has privileges to add a user directly by providing the details.

- Delete &Block Users

Administrator has a right to delete or block a user. The default status of a new user registered is set as blocked. The admin must accept the new user by unblocking him.

MANAGE ORDERS

Fig 1.5: Manage Orders

- View Order

Administrator can view the Orders which is generated by the users. He can verify the details of the purchase.

- Delete order

Admin can delete order from the orders list when the product is taken for delivery.

1.4.1.2 MODERATORS

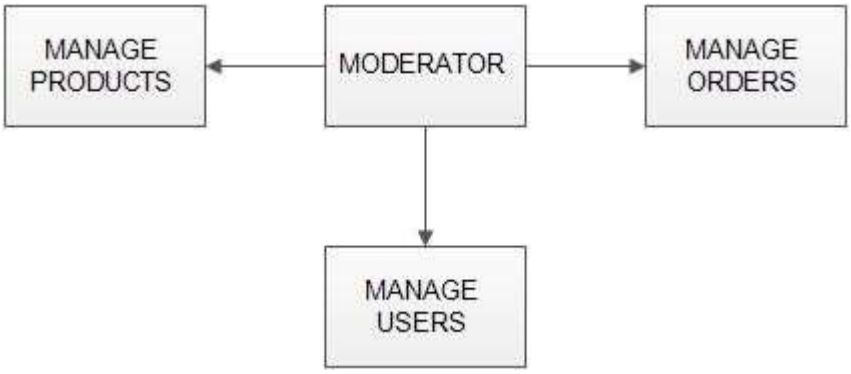


Fig 1.6: Moderator Module

A moderator is considered as a staff who can manage orders for the time being. As a future update moderator may give facility to add and manage his own products . Moderators can reduce the work load of admin. Now

moderator has all the privilege an admin having except managing other moderators. He can add products and users. He can also check the orders and edit his profile.

- Manage products

- Manage users
- Manage orders

1.4.1.3 USERS

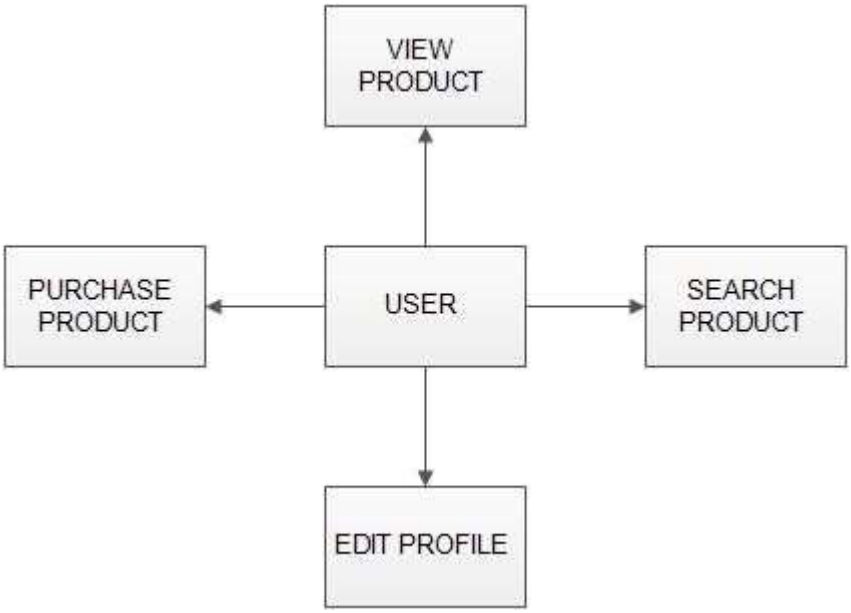


Fig 1.7: User Module

- Registration

  A new user will have to register in the system by providing essential details in order to view the products in the system. The admin must accept a new user by unblocking him.

- Login

  A user must login with his user name and password to the system after registration.

- View Products

User can view the list of products based on their names after successful login. A detailed description of a particular product with product name, products details, product image, price can be viewed by users.

- Search Product

Users can search for a particular product in the list by name.

- Add to cart:

The user can add the desired product into his cart by clicking add to cart option on the product.

He can view his cart by clicking on the cart button. All products added by cart can be viewed in the cart. User can remove an item from the cart by clicking remove.

- Submit Cart:

After confirming the items in the cart the user can submit the cart by providing a delivery address. On successful submitting the cart will become empty.

- History

In the history the user will have a view of pending orders.

- Edit

The user can view and edit the profile.

CHAPTER 2

SYSTEM ANALYSIS

System analysis is the process of gathering and interpreting facts, diagnosing problems and using the information to recommend improvements on the system. System analysis is

a problem solving activity that requires intensive communication between the system users and system developers.

System analysis or study is an important phase of any system development process. The system is viewed as a whole, the inputs are identified and the system is subjected to close study to identify the problem areas. The solutions are given as a proposal. The proposal is reviewed on user request and suitable changes are made. This loop ends as soon as the user is satisfied with the proposal.

2.1 EXISTING SYSTEM

The current system for shopping is to visit the shop manually and from the available product choose the item customer want and buying the item by payment of the price of the item .

1. It is less user-friendly.

2. User must go to shop and select products.
3. It is difficult to identify the required product.
4. Description of the product limited.
5. It is a time consuming process
6. Not in reach of distant users.

2.2 PROPOSED SYSTEM

In the proposed system customer need not go to the shop for buying the products. He can order the product he wish to buy through the application in his Smartphone. The shop owner will be admin of the system. Shop owner can appoint moderators who will help owner in managing the customers and product orders. The system also recommends a home delivery system for the purchased products.

2.3 SYSTEM REQUIREMENT SPECIFICATION

2.3.1 GENERAL DESCRIPTION

Product Description:

The system consists of two parts .A web application which can provide the online shopping service and an android application for the customer to

access the web service from his Smartphone. Web application should be able to help the customer for selecting his item and to help the owner in managing the orders from the customers.

Problem Statement:

As online shopping became a trend nowadays the regular shops are losing their customers to online brands. Customers have effortless shopping experience and saving time through shopping online. For competing with those online brands , If shops are providing an online portal where their customers can shop through internet and get the products at their doors it will increase the number of customers.

2.3.2 *SYSTEM OBJECTIVES*

- To provide an android application for online shopping of products in an existing shop.
- To provide a online shopping web site for the same shop.

2.3.3 *SYSTEM REQUIREMENTS*

2.3.3.1 NON FUNCTIONAL REQUIREMENTS

i. EFFICIENCY REQUIREMENT

When an online shopping cart android application implemented customer can purchase product in an efficient manner.

ii. RELIABILITY REQUIREMENT

The system should provide a reliable environment to both customers and owner. All orders should be reaching at the admin without any errors.

iii. USABILITY REQUIREMENT

The android application is designed for user friendly environment and ease of use.

iv. IMPLEMENTATION REQUIREMENT

Implementation of the system using css and html in front end with jsp as back end and it will be used for database connectivity. And the database part is developed by mysql. Responsive web designing is used for making the website compatible for any type of screen.

v. DELIVERY REQUIREMENT

The whole system is expected to be delivered in four months of time with

a weekly evaluation by the project guide.

2.3.3.2 FUNCTIONAL REQUIREMENTS

USER

- USER LOGIN

Description of feature

This feature used by the user to login into system. A user must login with his user name and password to the system after registration. If they are invalid, the user not allowed to enter the system.

Functional requirement

- Username and password will be provided after user registration is confirmed.
- Password should be hidden from others while typing it in the field

- REGISTER NEW USER

Description of feature

A new user will have to register in the system by providing essential details in order to view the products in the system. The admin must accept a new user by unblocking him.

Functional requirement

- System must be able to verify and validate information.
- The system must encrypt the password of the customer to provide security.

➢ PURCHASING AN ITEM

Description of feature

The user can add the desired product into his cart by clicking add to cart option on the product. He can view his cart by clicking on the cart button. All products added by cart can be viewed in the cart. User can remove an item from the cart by clicking remove. After confirming the items in the cart the user can submit the cart by providing a delivery address. On successful submitting the cart will become empty.

Functional requirement

- System must ensure that, only a registered customer can purchase items.

ADMIN

- MANAGE USER

Description of feature

The administrator can add user, delete user, view user and block user.

- MANAGE MODERATOR

Description of feature

The administrator can add moderator, delete moderator, block moderator and search for a moderator.

- MANAGE PRODUCTS

Description of feature

The administrator can add product, delete product and view product.

- MANAGE ORDERS

Description of feature

The administrator can view orders and delete orders. Functional

requirements

-The system must identify the login of the admin.

-Admin account should be secured so that only owner of the shop can access that account

MODERATOR

Description of features

A moderator is considered as a staff who can manage orders for the time being. As a future update moderator may give facility to add and manage his own products. Moderators can reduce the work load of admin. Now moderator has all the privilege of an admin having except managing other moderators. He can manage users and manage products. He can also check the orders and edit his profile.

Functional requirement

-The system must identify the login of a moderator.

CHAPTER 3

SYSTEM DESIGN

System design is the solution for the creation of a new system. This phase focuses on the detailed implementation of the feasible system. It emphasis on translating design. Specifications to performance specification. System design has two phases of development

- Logical design
- Physical design

During logical design phase the analyst describes inputs (sources), output s(destinations), databases (data sores) and procedures (data flows) all in a format that meets the user requirements. The analyst also specifies the needs of the user at a level that virtually determines the information flow in and out of the system and the data resources. Here the logical design is done through data flow diagrams and database design. The physical design is followed by physical design or coding. Physical design produces the working system by defining the design specifications, which specify

exactly what the candidate system must do. The programmers write the necessary programs that accept input from the user, perform

necessary processing on accepted data and produce the required report on a hard copy or display it on the screen.

3.1 INPUT AND OUTPUT DESIGN

3.1.1 INPUT DESIGN:

Input design is the link that ties the information system into the world of its

users. The input design involves determining the inputs, validating the data, minimizing the data entry and provides a multi-user facility. Inaccurate inputs are the most common cause of errors in data processing. Errors entered by the data entry operators can be controlled by input design. The user-originated inputs are converted to a computer based format in the input design. Input data are collected and organized into groups of similar data. Once identified, the appropriate input media are selected for processing. All the input data are validated and if any data violates any conditions, the user is warned by a message. If the data satisfies all the conditions, it is transferred to the appropriate tables in the database. In this project the student details are to be entered at the time of registration. A page is designed for this purpose which is user friendly and easy to use. The design is done such that users get appropriate messages when exceptions occur.

3.1.2 OUTPUT DESIGN:

Computer output is the most important and direct source of information to the user. Output design is a very important phase since the output needs to be in an efficient manner. Efficient and intelligible output design improves the system relationship with the user and helps in decision making. Allowing the user to view the sample screen is important because the user is the ultimate judge of the quality of output. The output module of this system is the selected notifications.

3.2 SYSTEM TOOLS

The various system tools that have been used in developing both the front end and the back end of the project are being discussed in this chapter.

3.3.1.FRONT END:

Edge template engine and CSS are utilized to implement the frontend.

Edge template engine

Edge is a logical and batteries included template engine for Node.js. It can render any text-based format, whether is **HTML**, **Markdown** or **plain text** files.

CSS (Cascading Style Sheets)

CSS is a style sheet language used for describing the look and formatting of a document written in a markup language

3.3.2 BACK END

The back end is implemented using MongoDb which is used to design the databases.

MongoDb

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License (SSPL).

3.3 E-R DIAGRAMS

- LOGIN

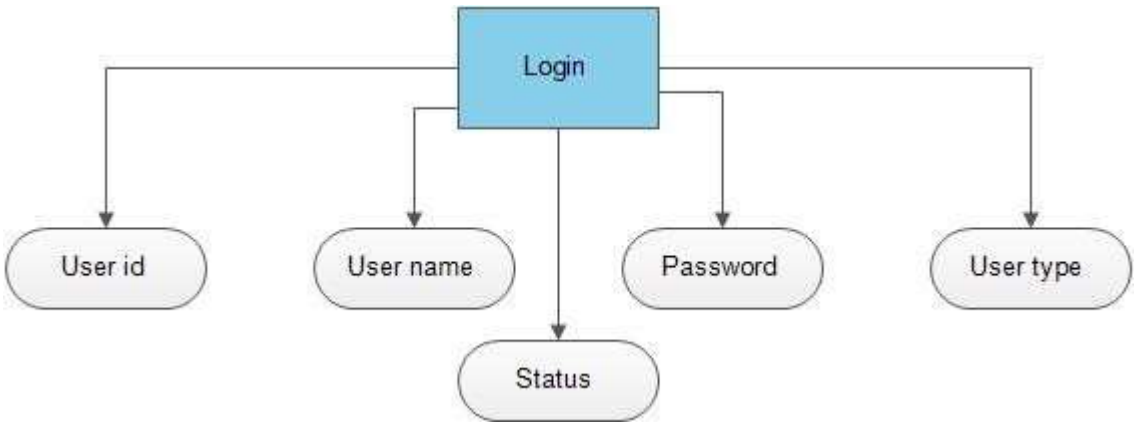


Fig 3.5: Login

- USER DETAILS

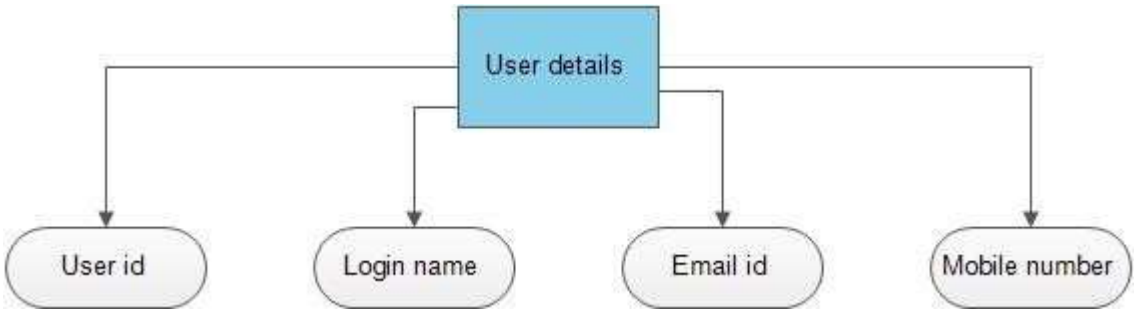


Fig 3.6: User Details

- PRODUCT DETAILS

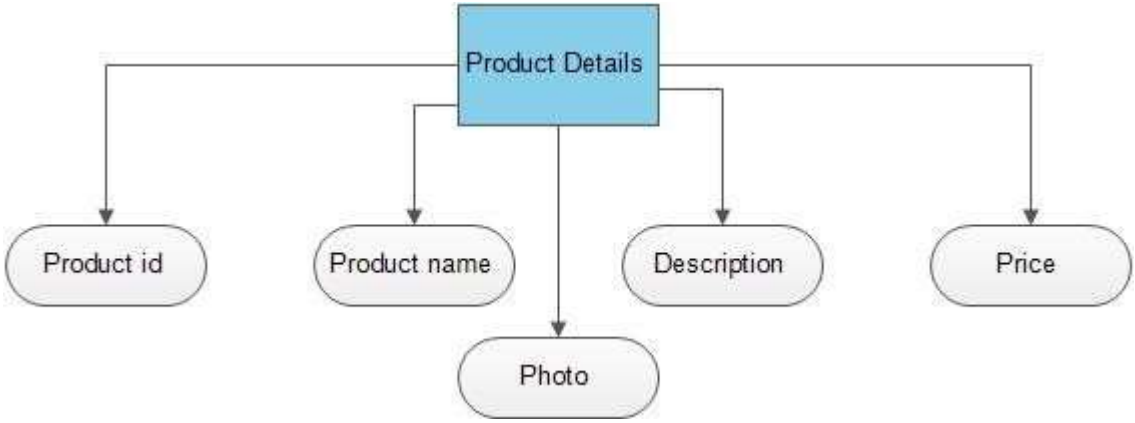


Fig 3.7: Product Details

- PRODUCT ORDERS

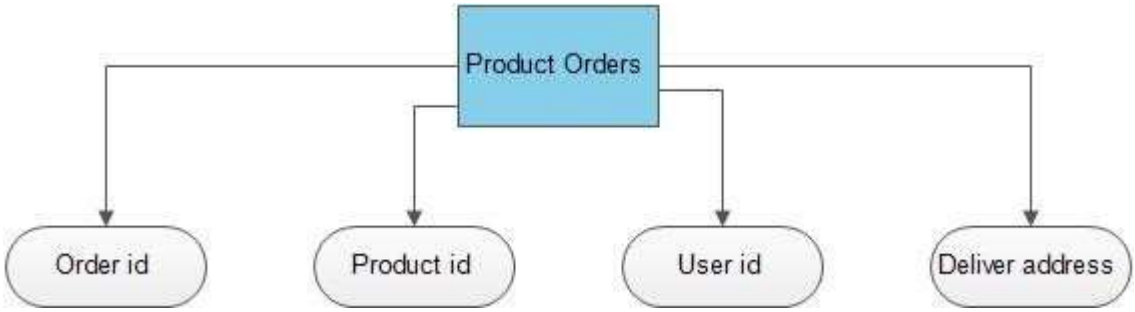


Fig 3.8: Product Orders

- COMPLETE DIAGRAM

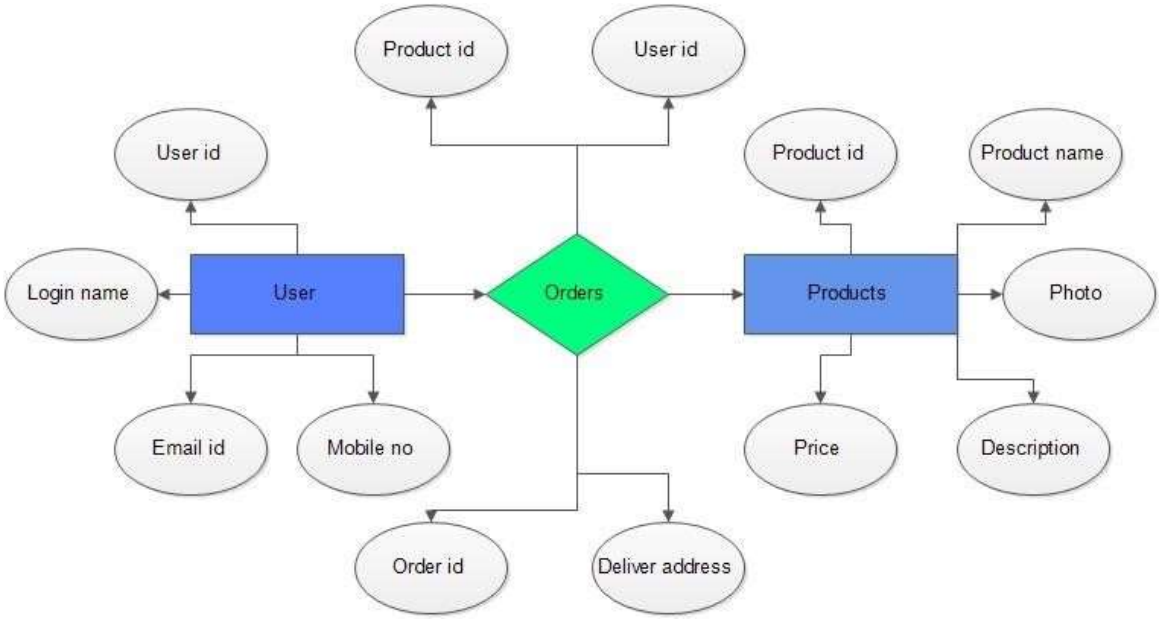


Fig 3.9: Complete Diagram

3.4 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a structured analysis and design tool that can be used for flowcharting. A DFD is a network that describes the flow of data and the processes that change or transform the data throughout a system. This network is constructed by using a set of symbols that do not imply any physical implementation. It has the purpose of clarifying system

requirements and identifying major transformations. So it is the starting point of the design phase that functionally decomposes the requirements specifications down to the lowest level of detail. DFD can be considered to an abstraction of the logic of an information-oriented or a process-oriented system flow-chart. For these reasons DFD's are often referred to as logical data flow diagrams.

EXTERNAL ENTITY

An external entity is a source or destination of a data flow. Only those entities which originate or receive data are represented on a data flow diagram. The symbol used is a rectangular box.

PROCESS

A process shows a transformation or manipulation of data flow within the system. The symbol used is an oval shape.

DATAFLOW

The data flow shows the flow of information from a source to its destination. Data flow is represented by a line, with arrowheads showing the direction of flow. Information always flows to or from a process and may be written, verbal or electronic. Each data flow may be referenced by the processes or data stores at its head and tail, or by a description of its contents.

DATA STORE

A data store is a holding place for information within the system: It is represented by an open ended narrow rectangle. Data stores may be long-term files such as sales ledgers, or may be short-term accumulations: for example batches of documents that are waiting to be processed. Each data store should be given a reference followed by an arbitrary number.

- LOGIN DFD

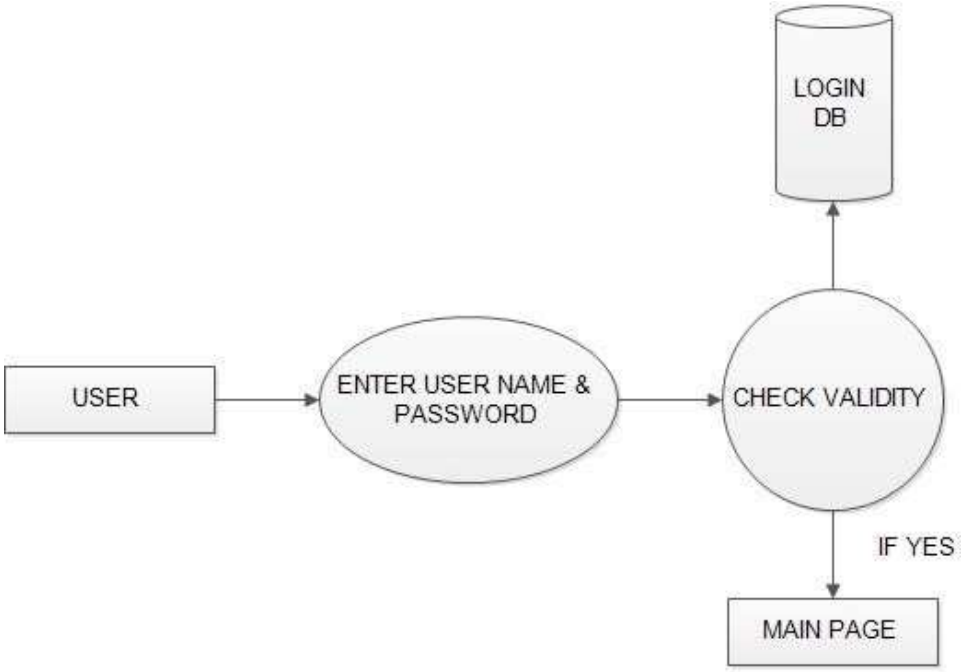


Fig 3.10: Login DFD

- REGISTRATION DFD

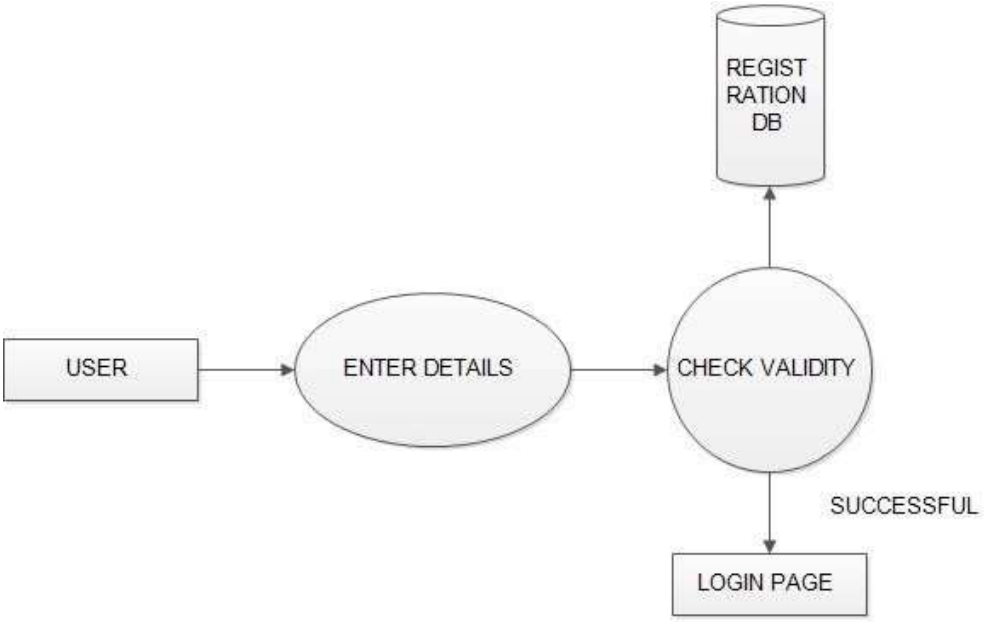


Fig 3.11: Registration DFD

- ADMIN DFD

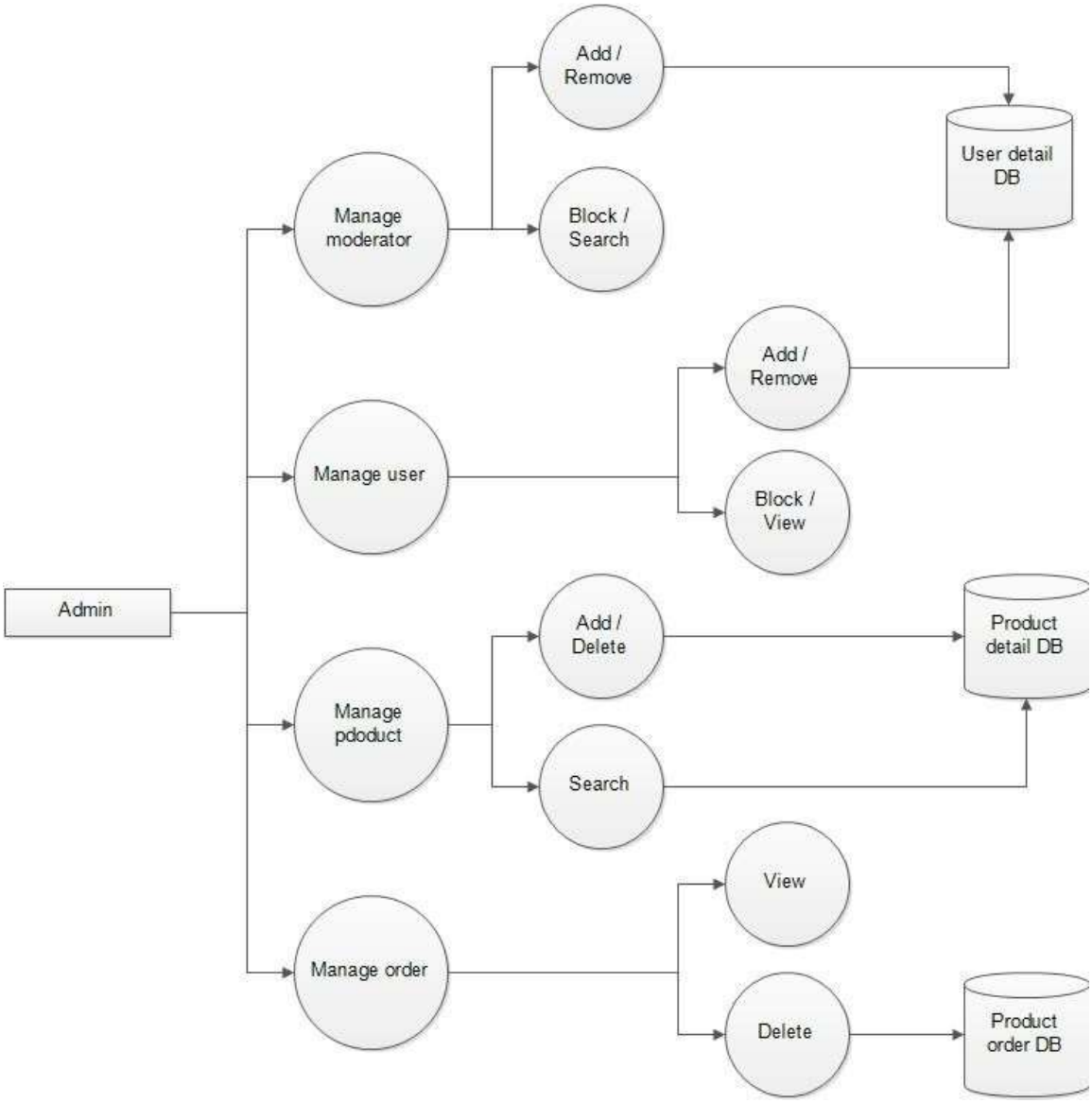
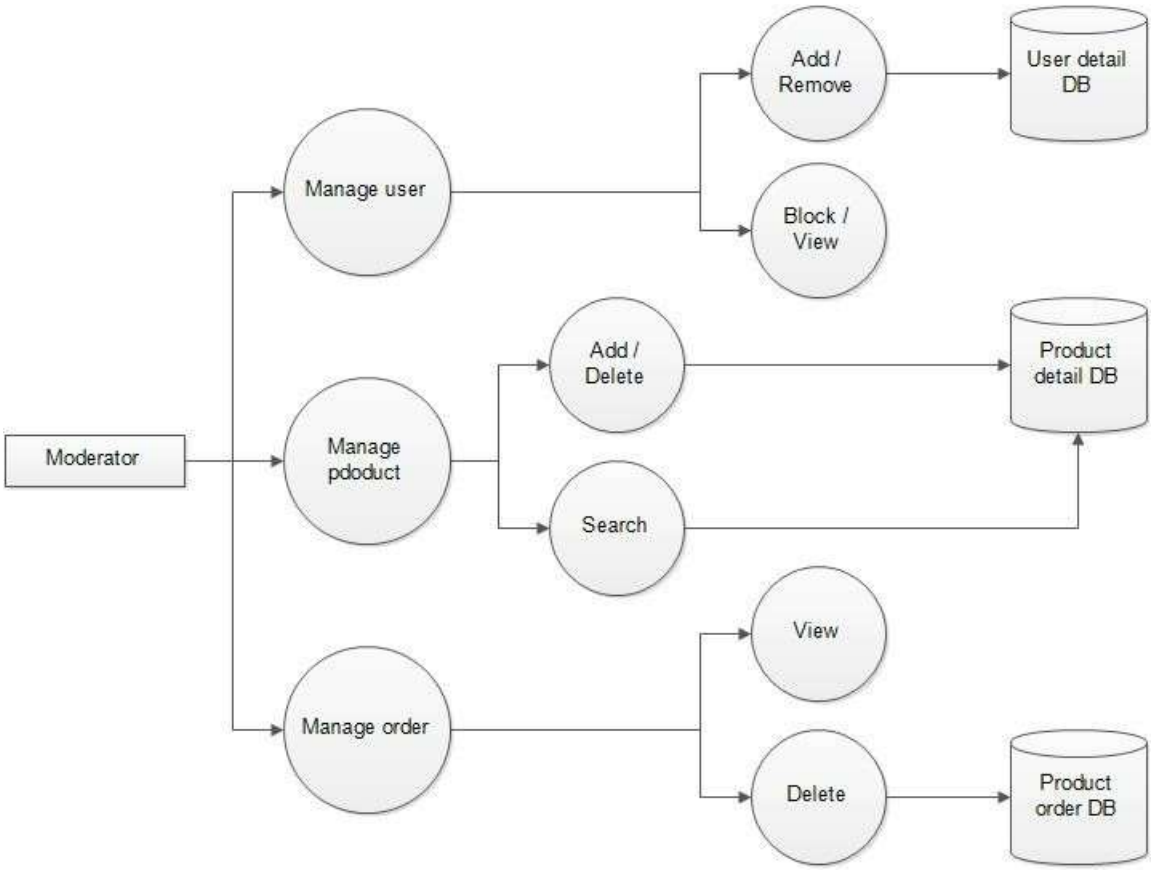


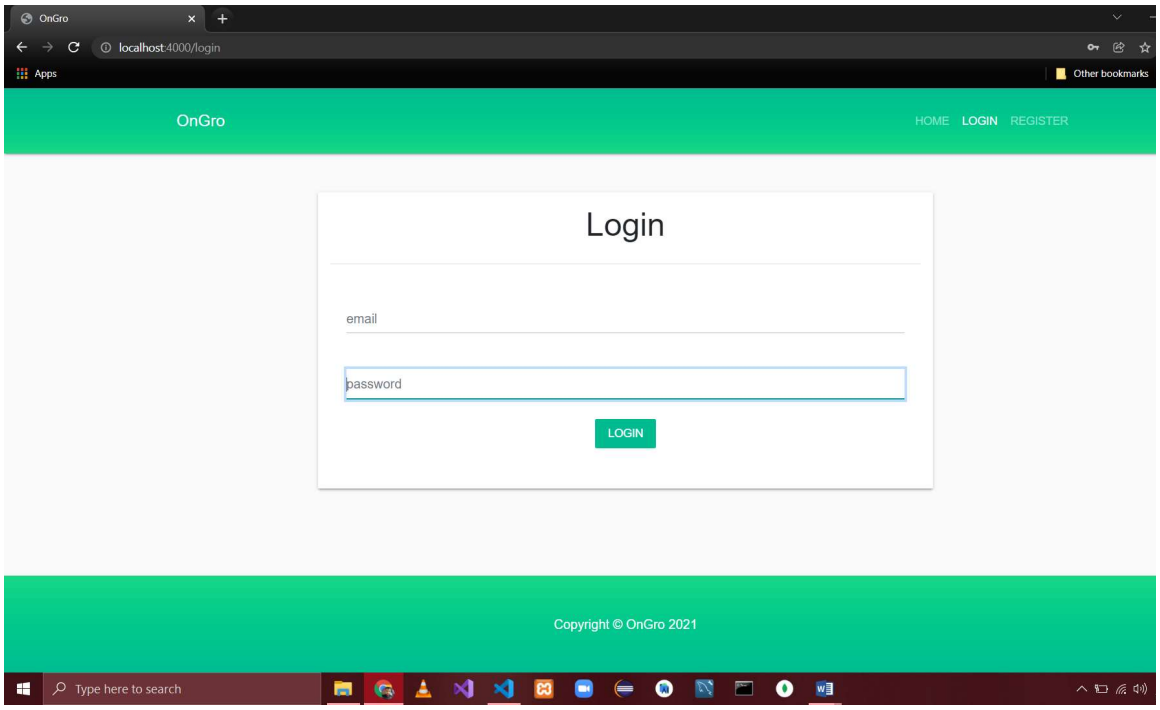
Fig 3.12: Admin DFD

- MODERATOR DFD

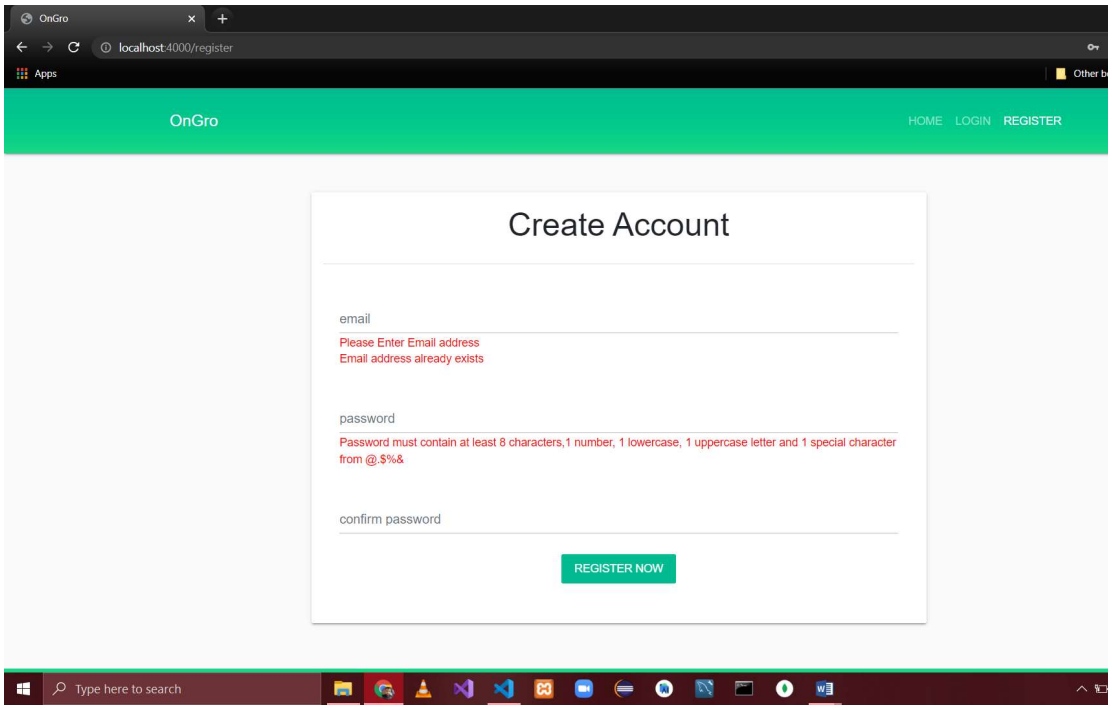


Screenshots:

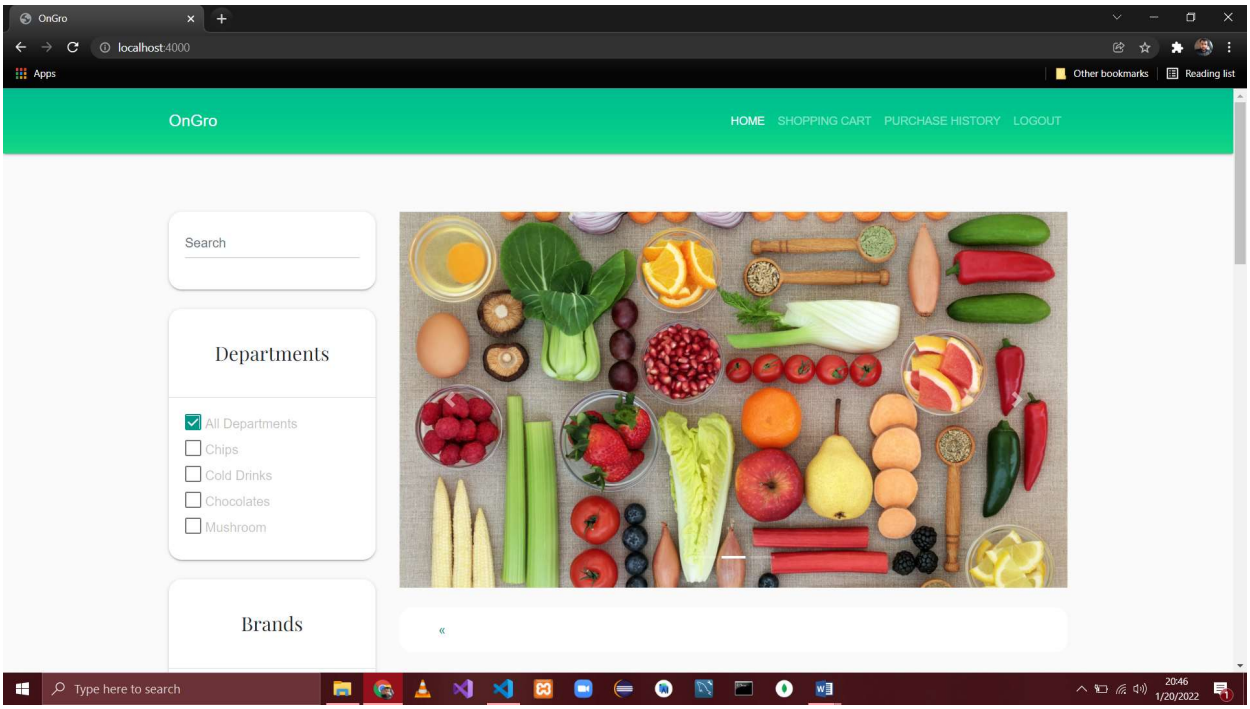
- LOGIN:



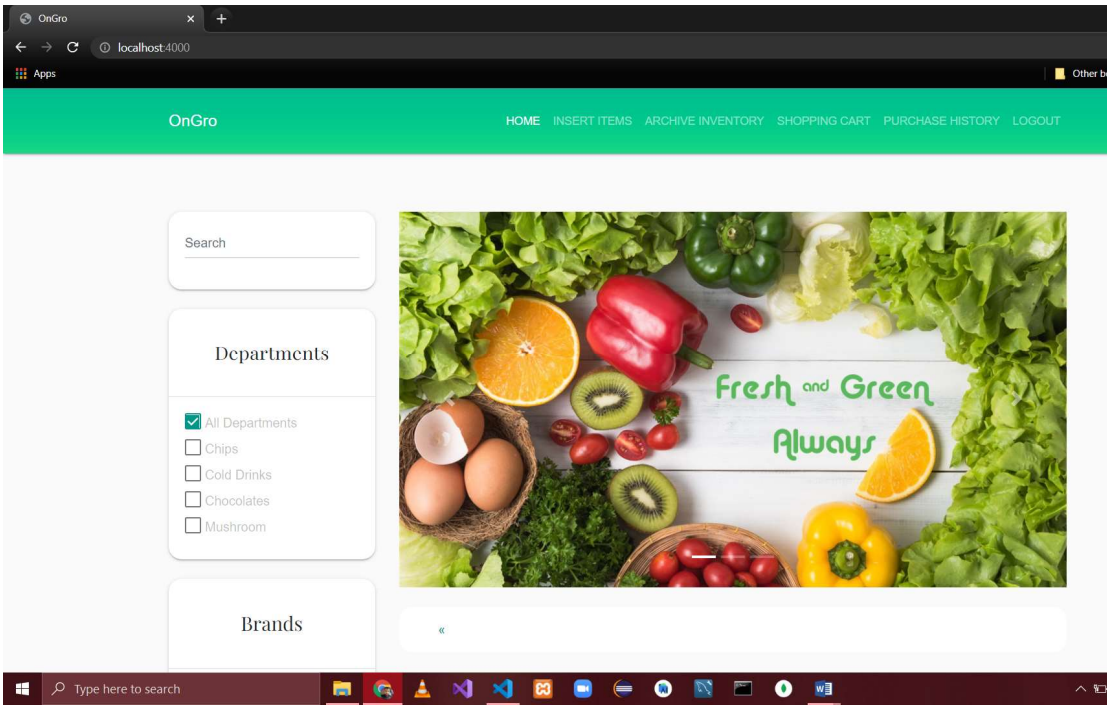
- REGISTRATION


OnGro
localhost:4000/register
Apps
OnGro
HOME LOGIN REGISTER
Create Account
email
Please Enter Email address
Email address already exists
password
Password must contain at least 8 characters,1 number, 1 lowercase, 1 uppercase letter and 1 special character from @.$%&
confirm password
REGISTER NOW
Type here to search

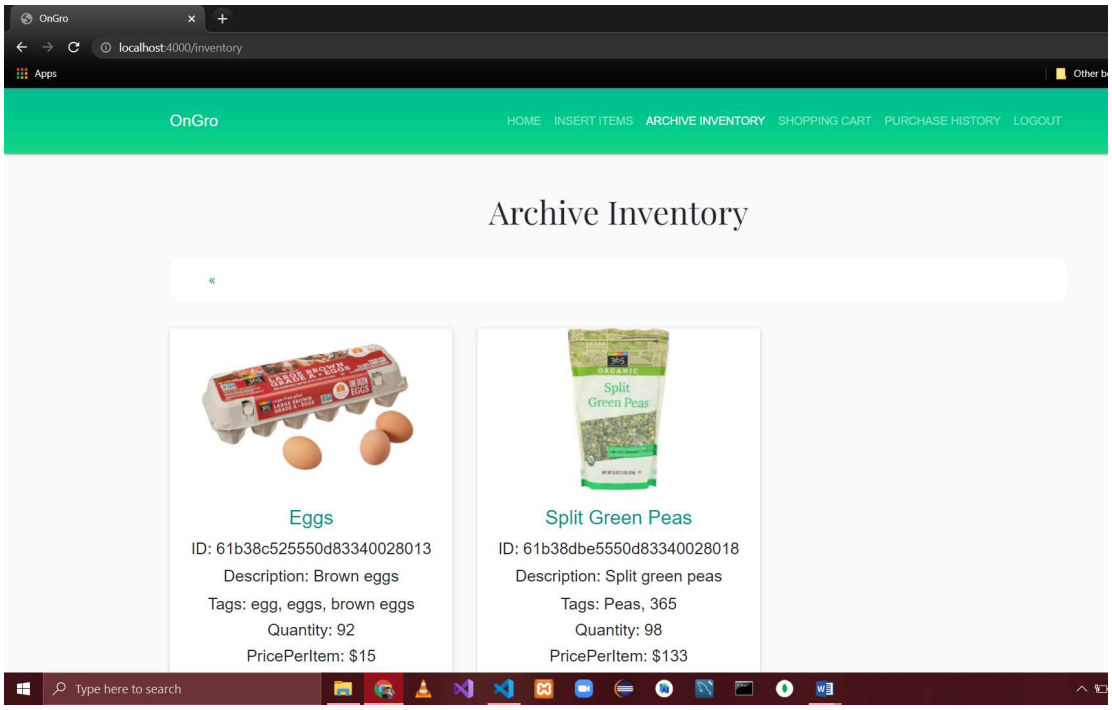
- HOME


OnGro
localhost:4000
Apps
Other bookmarks
Reading list
OnGro
HOME
SHOPPING CART
PURCHASE HISTORY
LOGOUT
Search
Departments
All Departments
Chips
Cold Drinks
Chocolates
Mushroom
Brands
«
Type here to search
20:46
1/20/2022

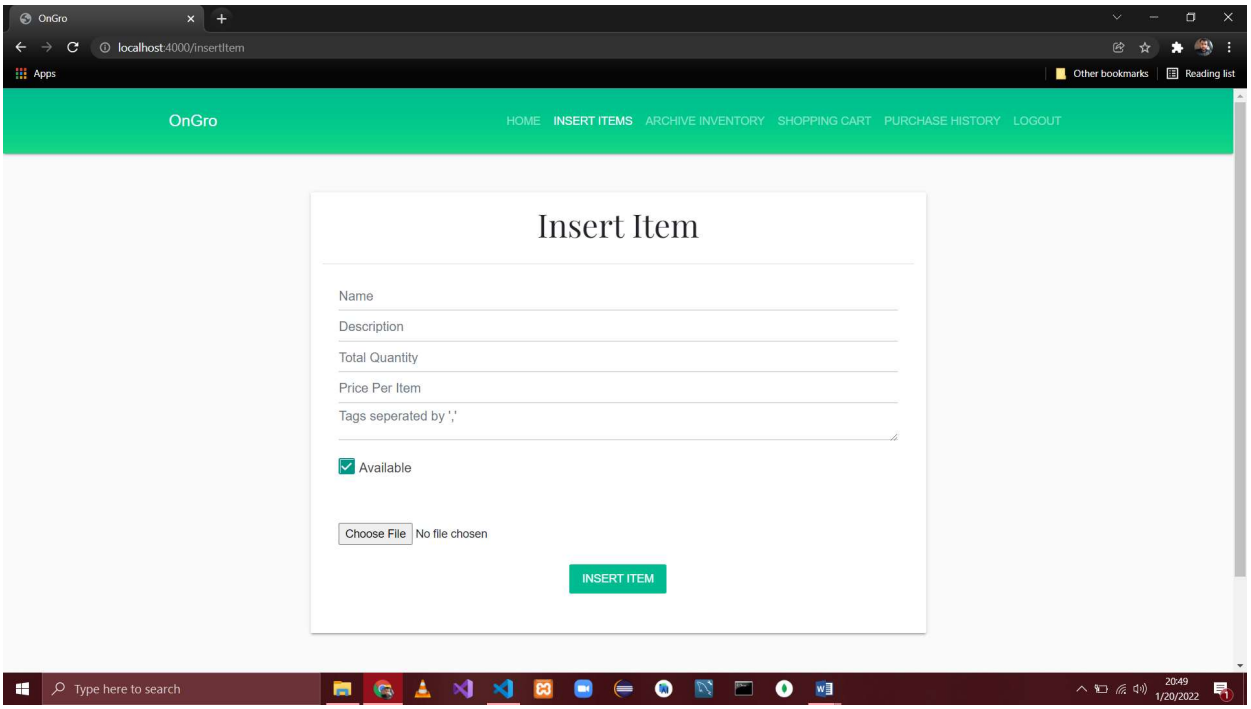
- ADMIN HOME



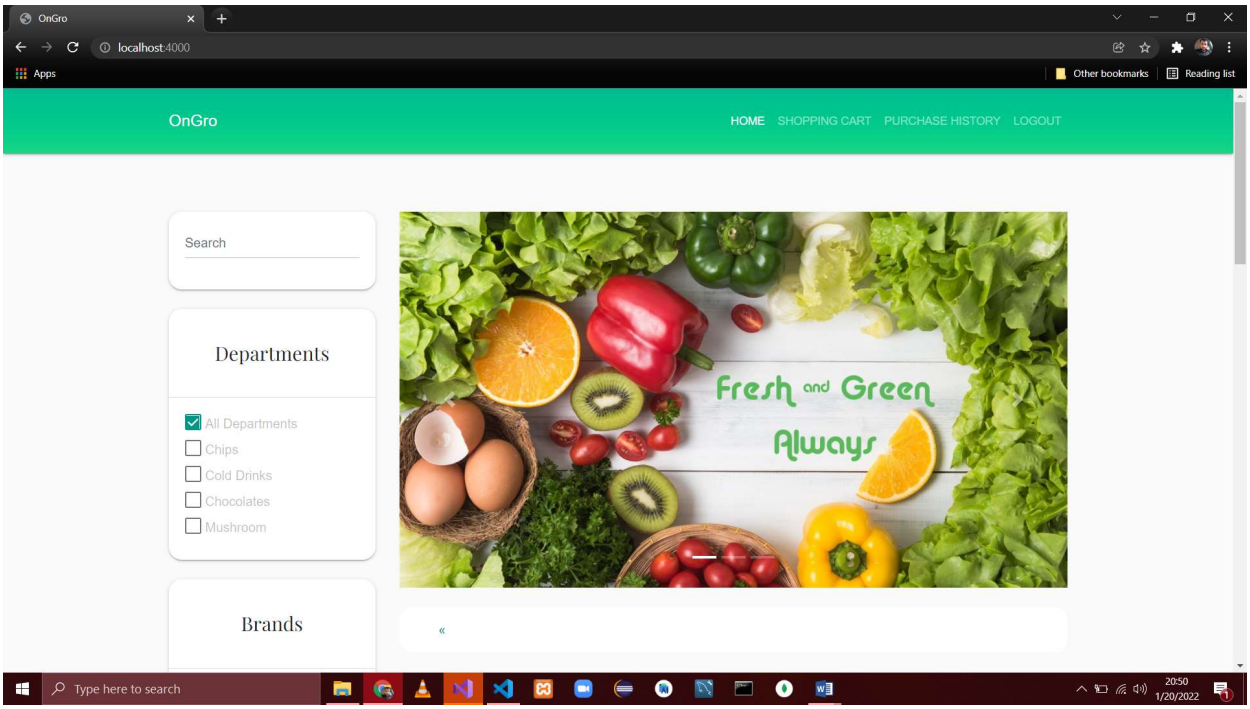
- ADMIN PRODUCT



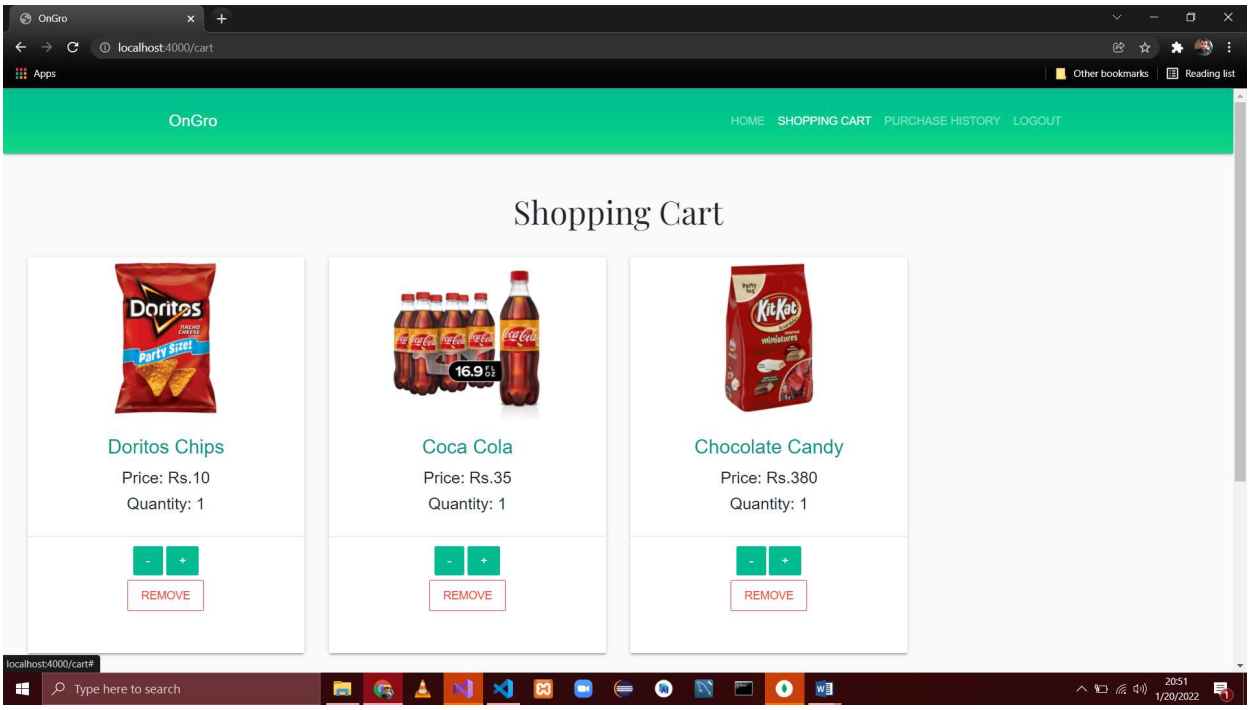
- ADMIN INSERT ITEM


OnGro
localhost:4000/insertItem
Apps
Other bookmarks
Reading list
OnGro
HOME
INSERT ITEMS
ARCHIVE INVENTORY
SHOPPING CART
PURCHASE HISTORY
LOGOUT
Insert Item
Name
Description
Total Quantity
Price Per Item
Tags seperated by ','
Available
Choose File
No file chosen
INSERT ITEM
Type here to search
20:49
1/20/2022

- USER HOME


OnGro
localhost:4000
Apps
Other bookmarks
Reading list
OnGro
HOME
SHOPPING CART
PURCHASE HISTORY
LOGOUT
Search
Departments
All Departments
Chips
Cold Drinks
Chocolates
Mushroom
Brands
Fresh and Green
Always
«
Type here to search
20:50
1/20/2022

- USER CART



SAMPLE CODE:

INDEX.JS

```
const expressEdge = require('express-edge')
const mongoose = require('mongoose')
const bodyParser = require('body-parser')
const fileUpload = require('express-fileupload')
const express = require('express')
const expressSession = require('express-session')
const connectMongo = require('connect-mongo')
const connectFlash = require('connect-flash')
const edge = require('edge.js')

const insertItemController = require('./controllers/insertItem')
const homePageController = require('./controllers/homePage')
const postItemController = require('./controllers/postItem')
const getItemController = require('./controllers/getItem')
const registerController = require('./controllers/register')
const createUserController = require('./controllers/createUser')
const loginController = require('./controllers/login')
const signInUserController = require('./controllers/signInUser')
const logoutController = require('./controllers/logout')
const addToCartController = require('./controllers/addToCart')
const storeTransactionController = require('./controllers/storeTransaction')
const cartController = require('./controllers/cart')
const isRegisteredController = require('./controllers/isRegistered')
const removeFromCartController = require('./controllers/removeFromCart')
const updateCartController = require('./controllers/updateCart')
const purchaseHistoryController = require('./controllers/purchaseHistory')
const updateItemController = require('./controllers/updateItem')
const editItemController = require('./controllers/editItem')
const deleteItemController = require('./controllers/deleteItem')
const inventoryController = require('./controllers/inventory')
const searchNfilterController = require('./controllers/searchNfilter')
const getFilterController = require('./controllers/getFilters')


const app = new express()
```

```
mongoose.connect('mongodb://localhost/FreshNGreen')

app.use(fileUpload())
app.use(express.static('public'))
app.use(expressEdge)
app.set('views', `${__dirname}/views`)
app.use(bodyParser.json())
app.use(bodyParser.urlencoded({ extended: true }))
app.use(connectFlash())


const mongoStore = connectMongo(expressSession)
app.use(expressSession({
    secret: 'secret',
    store: new mongoStore({
        mongooseConnection: mongoose.connection
    })
}))

app.use('*', (req, res, next) => {
    edge.global('auth', req.session.userId)
    next()
})

const auth = require('./middleware/auth')
const redirectOnAuth = require('./middleware/redirectOnAuth')
const postItem = require('./middleware/postItem')

app.get('/', homePageController)
app.get('/insertItem', auth, insertItemController)
app.get('/item/:id', getItemController)


app.get('/register', redirectOnAuth, registerController)
app.get('/login', redirectOnAuth, loginController)
```

```
app.get('/logout', auth, logoutController)
app.get('/cart', cartController)
app.get('/isRegistered/:id', isRegisteredController)
app.get('/purchaseHistory', purchaseHistoryController)
app.get('/updateItem/:id', updateItemController)
app.get('/deleteItem/:id', deleteItemController)
app.get('/inventory', inventoryController)
app.get('/searchNfilter/:search/:brand/:category/:minPrice/:maxPrice',
searchNfilterController)
app.get('/getFilters', getFilterController)


app.post('/postItem', postItem, postItemController)
app.post('/createUser', redirectOnAuth, createUserController)
app.post('/signInUser', redirectOnAuth, signInUserController)
app.post('/addToCart', addToCartController)
app.post('/storeTransaction', storeTransactionController)
app.post('/removeFromCart', removeFromCartController)
app.post('/updateCart', updateCartController)
app.post('/editItem', editItemController)


app.use((req, res) => res.render('notFound'))

app.listen(4000, () => {
    console.log("App Listening")
})
```

APP.EDGE:

```
<!DOCTYPE html>
<html lang="en">

<head>
```

```
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta name="description" content="">
  <meta name="author" content="">

  <title>OnGro</title>

  <!-- Bootstrap CSS -->
  <link href="/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

  <!-- Custom CSS -->
  <link href="/css/shop-homepage.css" rel="stylesheet">
  <link rel="stylesheet"
    href="https://unpkg.com/bootstrap-material-design@4.1.1/dist/css/bootstrap-material-design.min.css"
    integrity="sha384-wXznGJNEXNG1NFsbm0ugrLFMQPWswR3lds2VeinahP8N0zJw9VWSopbjv2x7WCvX"
crossorigin="anonymous">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
  <link
href="https://fonts.googleapis.com/css?family=Playfair+Display&display=swap"
rel="stylesheet">

  <!-- Bootstrap JavaScript -->
  <script src="/vendor/jquery/jquery.min.js"></script>
  <script src="/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

</head>

<body>

  <!-- Navigation -->
```

```
  <nav class="navbar navbar-expand-lg navbar-dark main_nav">
    <div class="container">
      <a class="navbar-brand" href="/">OnGro</a>
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarResponsive"
        aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle
navigation">
        <i class="material-icons">menu</i>
      </button>
      <div class="collapse navbar-collapse" id="navbarResponsive">
        <ul class="navbar-nav ml-auto">
          <li class="nav-item active">
            <a class="nav-link" href="/">Home
            </a>
          </li>
          @if(!auth)
          <li class="nav-item login">
            <a class="nav-link" href="/login">Login</a>
          </li>
          <li class="nav-item register">
            <a class="nav-link" href="/register">Register</a>
          </li>

          @else
          @if(auth=='admin@admin.com')
          <li class="nav-item insert">
            <a class="nav-link" href="/insertItem">Insert Items</a>
          </li>
          <li class="nav-item inventory">
            <a class="nav-link" href="/inventory">Archive Inventory</a>
          </li>
          @endif
          <li class="nav-item cart">
            <a class="nav-link" href="/cart">Shopping Cart</a>
```

```
          </li>
          <li class="nav-item purchase_history">
            <a class="nav-link" href="/purchaseHistory">Purchase History</a>
          </li>
          <li class="nav-item logout">
            <a class="nav-link" href="/logout">Logout</a>
          </li>

          @endif

        </ul>
      </div>
    </div>
  </nav>
  <div class="main container-fluid">
    @!section('content')
  </div>

  <footer class="py-5 bg-dark">
    <div class="container">
      <div class="m-0 text-center text-white">Copyright &copy; OnGro 2021</div>
    </div>
  </footer>

  <script src="https://unpkg.com/popper.js@1.12.6/dist/umd/popper.js"
    integrity="sha384-
fA23ZRQ3G/J53mElWqVJEGJzU0sTs+SvzG8fXVWP+kJQ1lwFAOkcUOysnlKJC33U"
    crossorigin="anonymous"></script>
  <script src="https://unpkg.com/bootstrap-material-
design@4.1.1/dist/js/bootstrap-material-design.js"
    integrity="sha384-
CauSuKpEqAFajSpkdjv3z9t8E7RlpJ1UP0lKM/+NdtSarroVKu069AlsRPKkFBz9"
    crossorigin="anonymous"></script>
```

```
  <script>$(document).ready(function () { $('body').bootstrapMaterialDesign();
});</script>
</body>

</html>
```

CART.EDGE:

```
@layout('layouts.app')

@section('content')
<script>
  $(document).ready(function(){
      $("li").removeClass("active");
      $("li.cart").addClass("active");
  });
</script>
    @if(EmptyCart)
        <h1 class="sty" style="text-align : center;">Shopping Cart is Empty</h1>
    @else
    <h1 class="sty" style="text-align : center;">Shopping Cart</h1>
    <br>
    <div class="col-lg-12">
    <div class="row">
        @each(item in ItemsArray)
        <div class="col-lg-3 col-md-6 mb-4">
           <div class="card h-100 text-center">
              <form action="/updateCart" method="POST">
                <a href="#"><img class="card-img-top" src="{{item.imagePath}}"
alt=""></a>
                <div class="card-body">
                  <h4 class="card-title textlimit">
                    <a href="/item/{{item.itemId}}">{{item.name}}</a>
                  </h4>
                  <h5>Price: Rs.{{item.price}}</h5>
```

```
                <h5>Quantity: {{item.quantity}}</h5>
                <input type="text" style="display:none" name="itemId"
value="{{item.itemId}}">
                <input type="text" style="display:none" name="quantity"
value="{{item.quantity}}">

              </div>
              <div class="card-footer" style="text-align: center;">
                  <button class="btn addbtncolor" name="change" value="dec">-
</button>
                  <button class="btn addbtncolor" name="change"
value="inc">+</button>
            </form>
            <form action="/removeFromCart" method="POST">
             <input type="text" style="display:none" name="itemId"
value="{{item.itemId}}">
             <button class="btn btn-outline-danger">Remove</button>
           </form>
         </div>
           </div>
         </div>
         <br>
       @endeach

    </div>
    </div>
    <div style="text-align :center">
    <h3>Total Purchase: Rs.{{price}}</h3>
    <form action="/storeTransaction" method="POST">
      <input type="text" class="form form-control" name="address" required
placeholder="Delivery Address" style="max-width: 20rem;text-align : center;margin:
0 auto;">
      <br>
        <button class="btn addbtncolor">Check out & Pay</button>
```

```
    </form>
  </div>
    @endif
@endsection
```

FORMVALIDATE.JS:

```
$(document).ready(function () {
  var pattern = /^.*(?=.{8,})(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[@.$%&]).*$/;
  var noEmail = false;
  $("input").after("<span></span>");
  $("input")
    .next()
    .css("color", "red");
  $("input.email")
    .next()
    .text("Please Enter Email address");
  $("input.email")
    .next()
    .after("<p></p>");
  $("input.email")
    .next()
    .next()
    .css("color", "red");
  $("input.email")
    .next()
    .next()
    .text("Email address already exists");
  $("input[type=password]")
    .next()
    .text(
      "Password must contain at least 8 characters,1 number, 1 lowercase, 1
uppercase letter and 1 special character from @.$%&"
    );
```

```
$("input.confpass")
  .next()
  .after("<p></p>");
$("input.confpass")
  .next()
  .next()
  .css("color", "red");
$("input.confpass")
  .next()
  .next()
  .text("Passwords doesn't match,Please check");
$("input.confpass")
  .next()
  .next()
  .hide();
$("span").hide();
$("p").hide();
$("input").focusin(function () {
  $(this)
    .next()
    .hide();
  if (
    $(this)
      .next()
      .next()
      .show()
  ) {
    $(this)
      .next()
      .next()
      .hide();
  }
});
$("input").focusout(function () {
```

```
    if ($(this).val() == "") {
      $(this)
        .next()
        .show();
    }
    if (!$(this).hasClass("email")) {
      if (!pattern.test($(this).val())) {
        $(this)
          .next()
          .show();
      } else {
        $(this)
          .next()
          .hide();
      }
      if ($("input.pass").val() == $("input.confpass").val()) {
        $("input.confpass")
          .next()
          .next()
          .hide();
      } else {
        $("input.confpass")
          .next()
          .next()
          .show();
      }
    } else {
      var boolmail = "/isRegistered/" + $(this).val();
      $.getJSON(boolmail, function (result) {
        if (result.alreadyRegistered == true) {
          $("input.email")
            .next()
            .next()
            .show();
```

```
      } else {
        $("input.email")
          .next()
          .next()
          .hide();
      }
    });
  }
});
$("button.regcheck").click(function (event) {
  if ($("input.email").val() == "") {
    $("input.email")
      .next()
      .show();
    event.preventDefault();
  } else {
    var boolemail = "/isRegistered/" + $("input.email").val();
    $.getJSON(boolemail, function (result) {
      if (result.alreadyRegistered == true) {
        $("input.email")
          .next()
          .next()
          .show();
        event.preventDefault();
      } else {
        noEmail = true;
      }
    });
    if ($("input[type=password]").val() == "") {
      $("input[type=password]")
        .next()
        .show();
      event.preventDefault();
    }
```

```
      if (!pattern.test($("input[type=password]").val())) {
        $("input[type=password]")
          .next()
          .show();
        event.preventDefault();
      }
      if ($("input.pass").val() != $("input.confpass").val()) {
        $("input.confpass")
          .next()
          .next()
          .show();
        event.preventDefault();
      }
      if (
        $("input.pass").val() == $("input.confpass").val() &&
        $("input.email").val() != "" &&
        $("input[type=password]").val() != "" &&
        pattern.test($("input[type=password]").val()) &&
        noEmail == true
      ) {
        $(this).trigger("click");
      }
    }
  });
});
```

ADDTOCART.JS:

```
const transaction = require("../database/models/Transaction");
const Item = require("../database/models/Item");

module.exports = async (req, res) => {
  itemId = req.body._id;
```

```
item_price = req.body.price;
item_name = req.body.name;
item_imgPath = req.body.imagePath;
console.log(item_imgPath);
if (!req.session.userId) {
  return res.redirect("/login");
}
userId = req.session.userId;
console.log(new Date());

if (req.session.cart) {
  console.log("ALready Have a cart");
  Cart = req.session.cart;
  alreadyPresent = false;

  for (var i in Cart.items) {
    total = 0;
    const item = await Item.findById(Cart.items[i].itemId);
    quantityAvailable = item.totalQuantity;
    console.log(quantityAvailable);
    console.log(Cart.items[i].itemId);
    if (Cart.items[i].itemId == itemId) {
      if (Cart.items[i].quantity < quantityAvailable) {
        price = parseFloat(Cart.items[i].price, 10);
        Cart.items[i].price =
          (price + parseFloat(item_price, 10)).toFixed(2) + "";
        Cart.price =
          (parseFloat(Cart.price, 10) + parseFloat(item_price, 10)).toFixed(
            2
          ) + "";
        Cart.items[i].quantity = Cart.items[i].quantity + 1;
      }
      alreadyPresent = true;
    }
```

```
    }
    if (!alreadyPresent) {
      Cart.items.push({
        itemId: itemId,
        quantity: 1,
        price: item_price,
        name: item_name,
        imagePath: item_imgPath
      });
      Cart.price =
        (parseFloat(Cart.price, 10) + parseFloat(item_price, 10)).toFixed(2) +
        "";
    }
    req.session.cart = Cart;
    console.log("Cart is here :", Cart);
  } else {
    console.log("Creating a cart");
    req.session.cart = {
      items: Array({
        itemId: itemId,
        quantity: 1,
        price: item_price,
        name: item_name,
        imagePath: item_imgPath
      }),
      price: item_price
    };
  }
  res.redirect("/cart");
};
```

STORETRANSACTION.JS:

```
const transaction = require("../database/models/Transaction");
const Item = require("../database/models/Item");
module.exports = (req, res) => {
  itemId = req.session.cart;
  address = req.body.address;
  console.log(address);
  userId = req.session.userId;
  CartItems = req.session.cart.items;
  CartPrice = req.session.cart.price;
  currentDate = new Date();
  req.session.cart = "";
  if (CartPrice > 0) {
    transaction.create(
      {
        userId,
        items: CartItems,
        timestamp: currentDate,
        totalPrice: CartPrice,
        deliveryAddress: address
      },
      (error, transaction) => {
        if (error) {
          console.log(error);
        } else {
          for (var i in CartItems) {
            Item.update(
              { _id: CartItems[i].itemId },
              {
                $inc: { totalQuantity: -CartItems[i].quantity }
              },
              (error, item) => {
                console.log(error);
              }
            );
```

```
          }
        }
        res.render("transactionCompleted", {
          userId,
          CartItems,
          currentDate,
          CartPrice,
          address
        });
      }
    );
  } else {
    var alreadyPurchased = 1;
    res.render("transactionCompleted", { alreadyPurchased });
  }
};
```

UPDATECART.JS

```
const Item = require("../database/models/Item");
module.exports = async (req, res) => {
  itemId = req.body.itemId;
  quantity = req.body.quantity;
  change = req.body.change;
  console.log(req.body);
  if (quantity > 0) {
    if (!req.session.userId) {
      return res.redirect("/login");
    }
    userId = req.session.userId;
    if (req.session.cart) {
      console.log("ALready Have a cart");
      Cart = req.session.cart;
```

```
    console.log("hi");
    for (var i in Cart.items) {
      total = 0;
      console.log(Cart.items[i].itemId);
      const item = await Item.findById(Cart.items[i].itemId);
      quantityAvailable = item.totalQuantity;
      console.log(quantityAvailable);
      if (Cart.items[i].itemId == itemId) {
        price = parseFloat(Cart.items[i].price, 10);
        if (change == "dec" && quantity > 1) {
          Cart.items[i].price =
            (price - price / Cart.items[i].quantity).toFixed(2) + "";
          Cart.price =
            (
              parseFloat(Cart.price, 10) -
              price / Cart.items[i].quantity
            ).toFixed(2) + "";
          Cart.items[i].quantity = Cart.items[i].quantity - 1;
        } else if (change == "inc" && quantity < quantityAvailable) {
          Cart.items[i].price =
            (price + price / Cart.items[i].quantity).toFixed(2) + "";
          Cart.price =
            (
              parseFloat(Cart.price, 10) +
              price / Cart.items[i].quantity
            ).toFixed(2) + "";
          Cart.items[i].quantity = Cart.items[i].quantity + 1;
        }
      }
    }
    req.session.cart = Cart;
    console.log("Cart is here :", Cart);
  }
}
```

```
  res.redirect("/cart");
};
```

CHAPTER 4

CONCLUSION

The project entitled **Online shopping system** was completed successfully.

The system has been developed with much care and free of errors and at the same time it is efficient and less time consuming. The purpose of this project was to develop a web application and an android application for purchasing items from a shop.

This project helped us in gaining valuable information and practical knowledge on several topics like designing web pages using html & css, usage of responsive templates, designing of android applications, and management of database using mysql . The entire system is secured. Also the project helped us understanding about the development phases of a project and software development life cycle. We learned how to test different features of a project.

This project has given us great satisfaction in having designed an application which can be implemented to any nearby shops or branded shops selling various kinds of products by simple modifications.

There is a scope for further development in our project to a great extend. A number of features can be added to this system in future like providing

moderator more control over products so that each moderator can maintain their own products. Another feature we wished to implement was providing classes for customers so that different offers can be given to each class. System may keep track of history of purchases of each customer and provide suggestions based on their history. These features could have implemented unless the time did not limite