TITLE

**OrderOnTheGo: Your On-Demand Food Ordering Solution**

**CONTENT**

**1. Introduction**

* **Project Title**
* **Team Members**

**2. Project Overview**

* **Purpose**
* **Features**

**3. Architecture**

* **Frontend**
* **Backend**
* **Database**

**4. Setup Instructions**

* **Prerequisites**
* **Installation**

**5. Folder Structure**

* **Frontend**
* **Backend**
* **Admin**

**6. Running the Application**

**7. API Documentation**

**8. Authentication**

**9. User Interface**

**10. Testing**

**1. Introduction**

**Project Title**

**Your On-Demand Food Ordering Solution**

**Team members members and their roles:**

* Gorantla Nagaraju
* Tankasala dinesh
* Gopidesi Harikrishna
* Gnana sai manikanta

**2. Project Overview**

Purpose or objectives and goals:

The proposed system is developed to manage ordering activities in fast food restaurant. It helps

to record customer submitted orders. The system should cover the following functions in order to

support the restaurant’s business process for achieving the objectives:

1. To allow the customer to make order, view order and make changes before submitting their

order and allow them make payment through prepayment card or credit card or debit card.

2. To provide interface that allows promotion and menu.

3. To prevent interface that shows customers’ orders detail to front-end and kitchen staffs for

delivering customers’ orders

4. Tools that generate reports that can be used for decision making

5. A tool that allows the management to modify the food information such as price, add a new

menu and many others as well as tools for managing user, system menu and promotion records.

This will minimize the number of employees at the back of the counter.

The system will help to reduce labor cost involved.

The system will be less probable to make mistake, since it’s a machine.

This will avoid long queues at the counter due to the speed of execution and number of optimum

screens to accommodate the maximum throughput.

The main objective of the Online Food Ordering System is to manage the details of Item

Category,Food,Delivery Address,Order,Shopping Cart. ... The purpose of the project is to build

an application program to reduce the manual work for managing the Item Category, Food,

Customer, Delivery Address.

Features

1. Easy To Use & Low Cost-To start your Restaurant Business online is very easy. Fast

and Easy to Setup. Start Ordering in Minutes.

2. No Technical Expertise Required

3. Customer support- All customer issues, complaints, queries & technical issues are

handled by our support team.

4. Fulfillment and Delivery-food delivery is provide within time.

5. Secure Payment-payment methods are secure.

6. Analytics & Reports-Predictive Analysis Reports and Graphs for managing future sales

to increase business value

7. Increase business volume-Maximize your business potential through increased exposure

to hungry customer.

8.Comprehensive Product Catalog

9.cart

10. procdure to check out

11.Address

12.Payment

13.Myorder

14.food Tracker

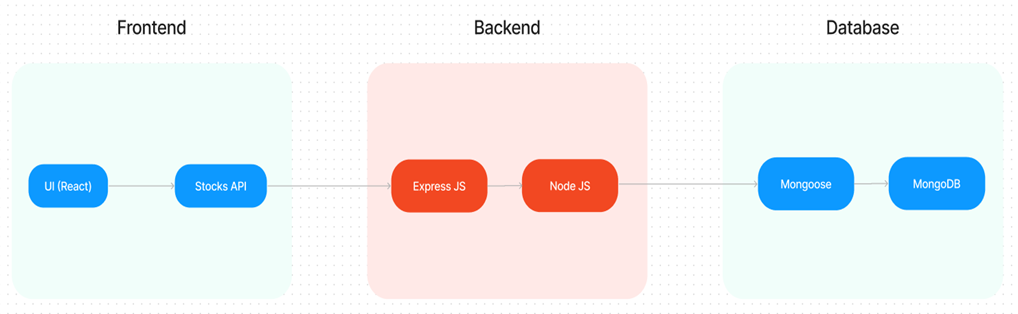
**3. Architecture**

In this architecture diagram:

• The frontend is represented by the "Frontend" section, including user interface components such as User Authentication, Cart, Products, Profile, Admin dashboard, etc.,

• The backend is represented by the "Backend" section, consisting of API endpoints for Users, Orders, Products, etc., It also includes Admin Authentication and an Admin Dashboard.

• The Database section represents the database that stores collections for Users, Admin, Cart, Orders, and products.



**4. Setup Instructions**

Pre-Requisite

To develop a full-stack food ordering app using React JS, Node.js, and MongoDB, there are  several prerequisites you should consider. Here are the key prerequisites for developing such an application:

Installation

**Node.js and npm:** Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side.

• Download: https://nodejs.org/en/download/

• Installation instructions: https://nodejs.org/en/download/package-manager/

**MongoDB:** Set up a MongoDB database to store hotel and booking information. Install MongoDB locally or use a cloud-based MongoDB service.

• Download: https://www.mongodb.com/try/download/community

• Installation instructions: https://docs.mongodb.com/manual/installation/

**Express.js:** Express.js is a web application framework for Node.js. Install Express.js to handle  server-side routing,middleware, and API development.

• Installation: Open your command prompt or terminal and run the following  command: **npm install express**

**React.js**: React.js is a popular JavaScript library for building user interfaces. It enables developers to  create interactive and reusable UI components, making it easier to build dynamic and responsive web  applications. To install React.js, a JavaScript library for building user interfaces, follow the installation  guide: https://reactjs.org/docs/create-a-new-react-app.html

**HTML, CSS, and JavaScript:** Basic knowledge of HTML for creating the structure of your app, CSS for styling,and JavaScript for client-side interactivity is essential.

**Database Connectivity:** Use a MongoDB driver or an Object-Document Mapping (ODM)  library like Mongoose to connect your Node.js server with the MongoDB database and perform  CRUD (Create, Read, Update, Delete) operations.

**Front-end Framework:** Utilize Angular to build the user-facing part of the application, including product listings, booking forms, and user interfaces for the admin dashboard.

**Version Control**: Use Git for version control, enabling collaboration and tracking  changes throughout the development process. Platforms like GitHub or Bitbucket can host  your repository.

• Git: Download and installation instructions can be found at: https://git scm.com/downloads

**Development Environment:** Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

 • Visual Studio Code: Download from https://code.visualstudio.com/download

• Sublime Text: Download from https://www.sublimetext.com/download

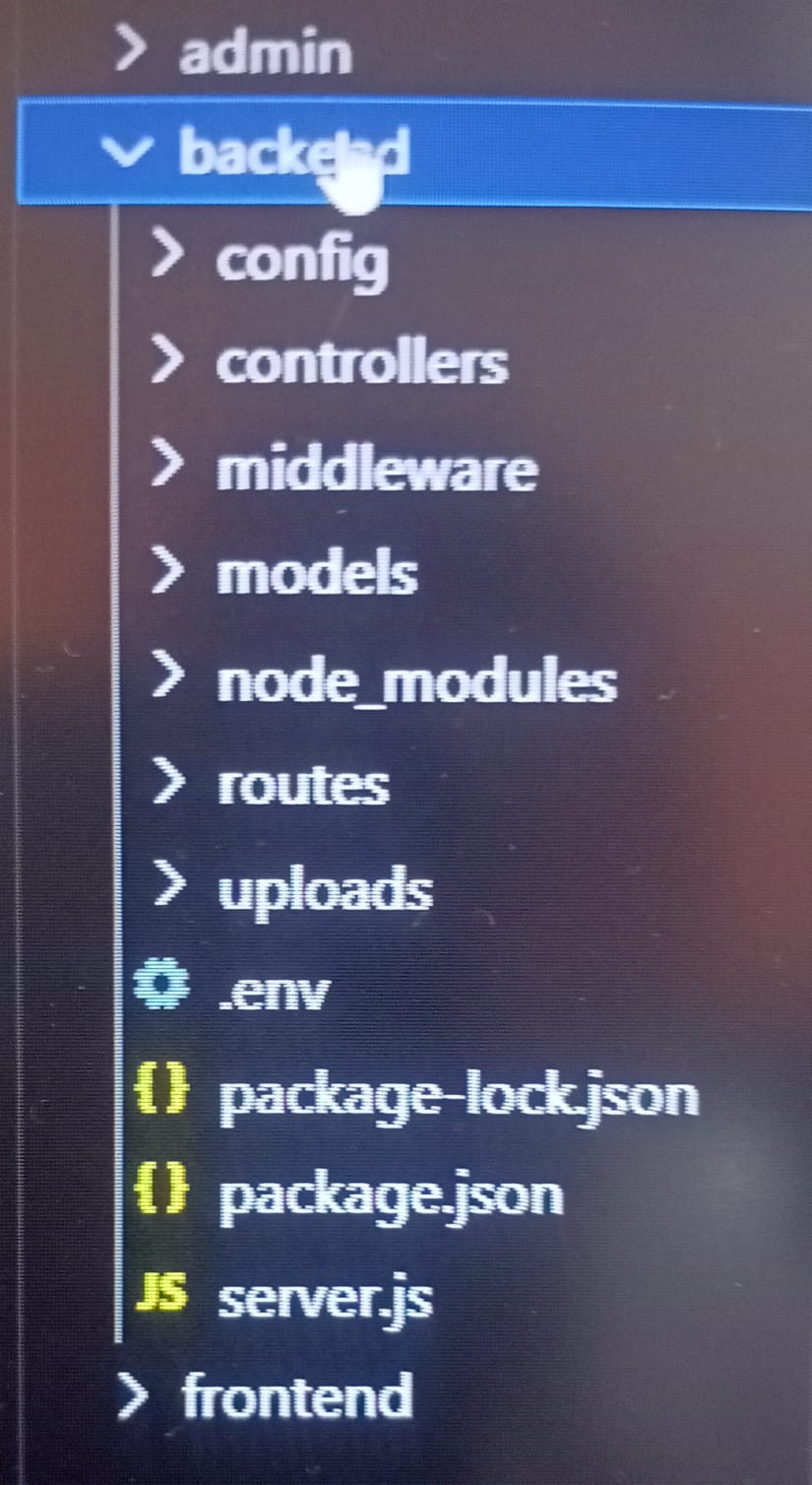
• WebStorm: Download from <https://www.jetbrains.com/webstorm/download>

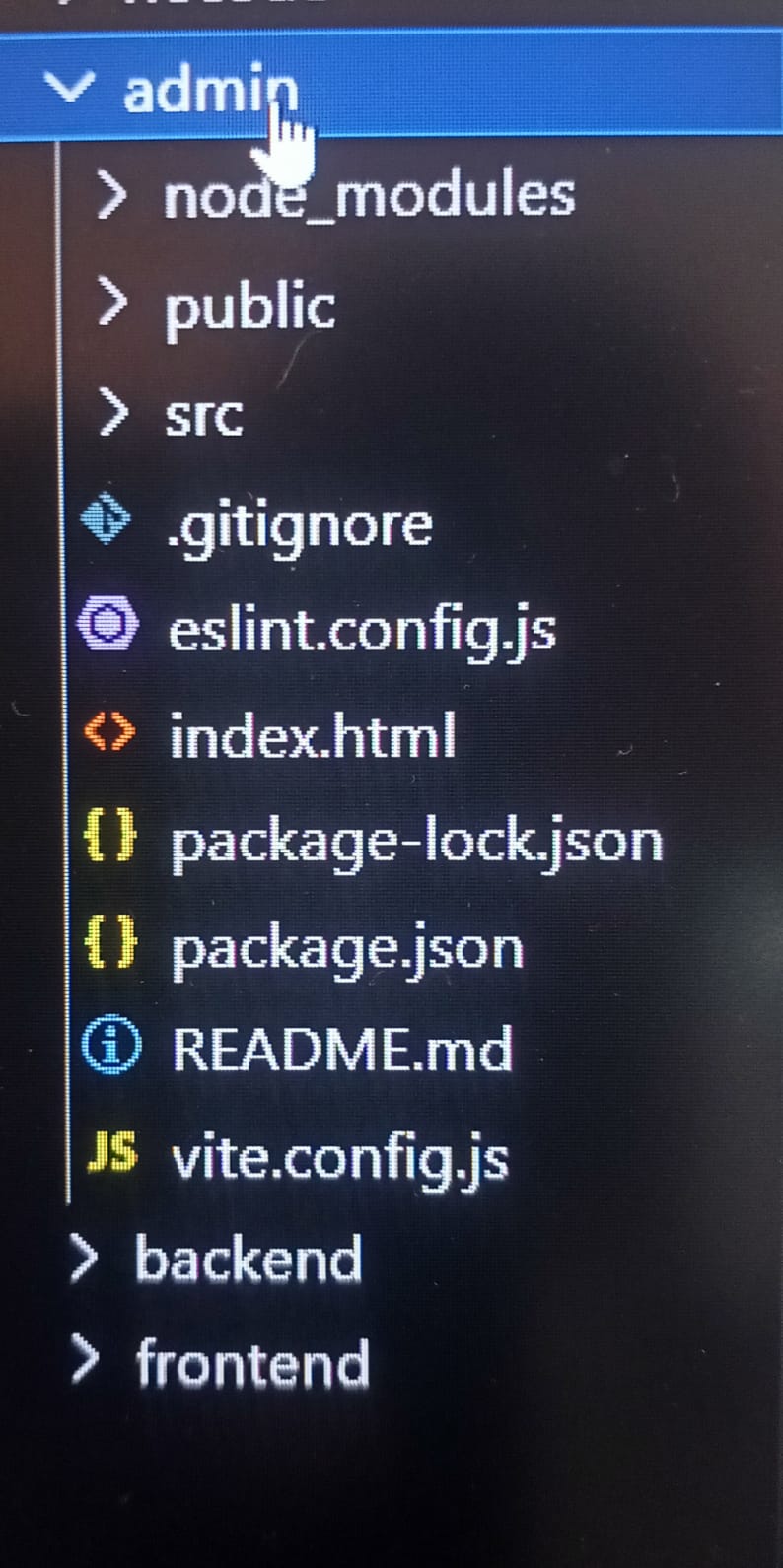
**To Connect the Database with Node JS go through the below provided link:**

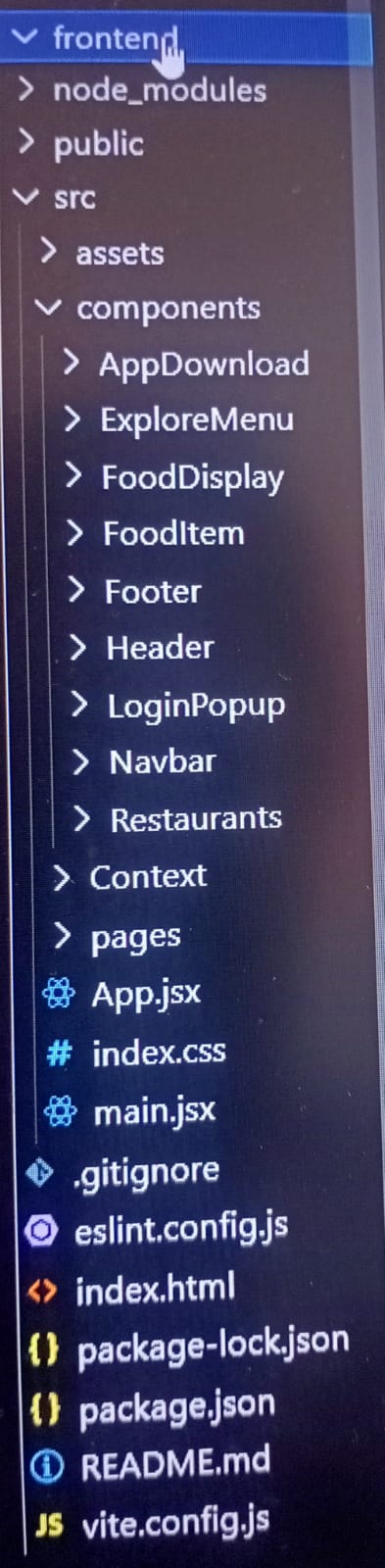
**Link:**https://www.mongodb+srv://rajugorantla859:Gorantla520@cluster0.4tf39.mongodb.net/food-den

**5. Folder Structure**

* Frontend
* Backend
* Admin







**6. Running the Application**

**1. User Flow:**

• Users start by registering for an account.

• After registration, they can log in with their credentials.

• Once logged in, they can check for the available products in the platform.

• Users can add the products they wish to their carts and order.

• They can then proceed by entering address and payment details.

• After ordering, they can check them in the profile section.

**2. Restaurant Flow:**

• Restaurants start by authenticating with their credentials.

• They need to get approval from the admin to start listing the products.

• They can add/edit the food items.

**3. Admin Flow:**

• Admins start by logging in with their credentials.

• Once logged in, they are directed to the Admin Dashboard.

• Admins can access the users list, products, orders, etc.

**Frontend:** npm start in the client directory.

npm run dev

Local: http://localhost:5173/

**Backend:** npm start in the server directory.

npm run server

Server Started on <http://localhost:4000>

**Admin:** npm start in the admin directory.

Npm run dev

Local: http://localhost:5174/

**7. API Documentation**

api endpoints

app.use("/api/food",foodRouter)

app.use("/images",express.static('uploads'))

app.use("/api/user",userRouter)

app.use("/api/cart",cartRouter)

app.use("/api/order",orderRouter)

**8. Authentication**

1. **User Credentials**: Users provide credentials, typically a username and password.

2. **Secure Storage**: Passwords are securely stored using hashing algorithms (like bcrypt or Argon2) to prevent exposure of sensitive information.

3. **Login Process**:

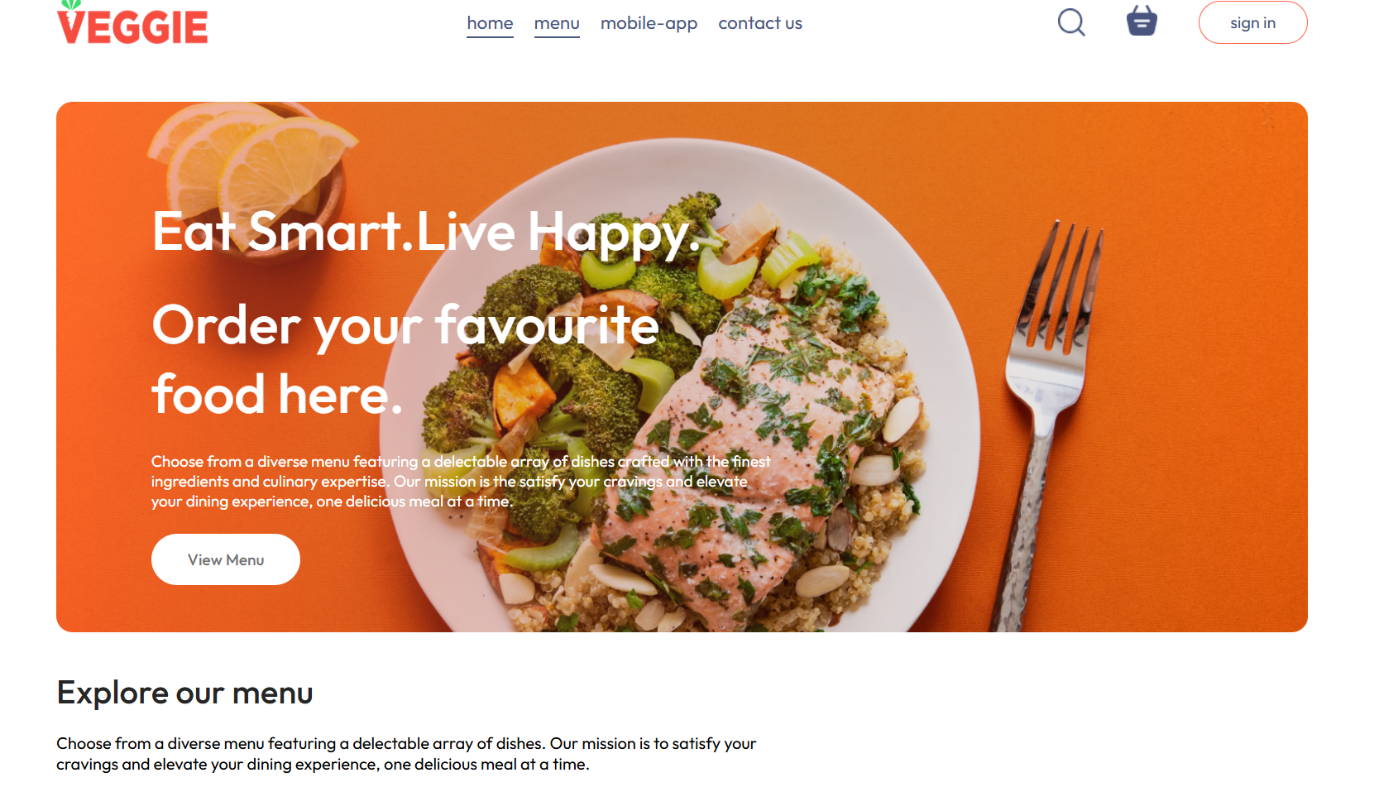
* When a user attempts to log in, the provided credentials are sent to the server.
* The server hashes the provided password and compares it to the stored hash.
* If they match, the user is authenticated.

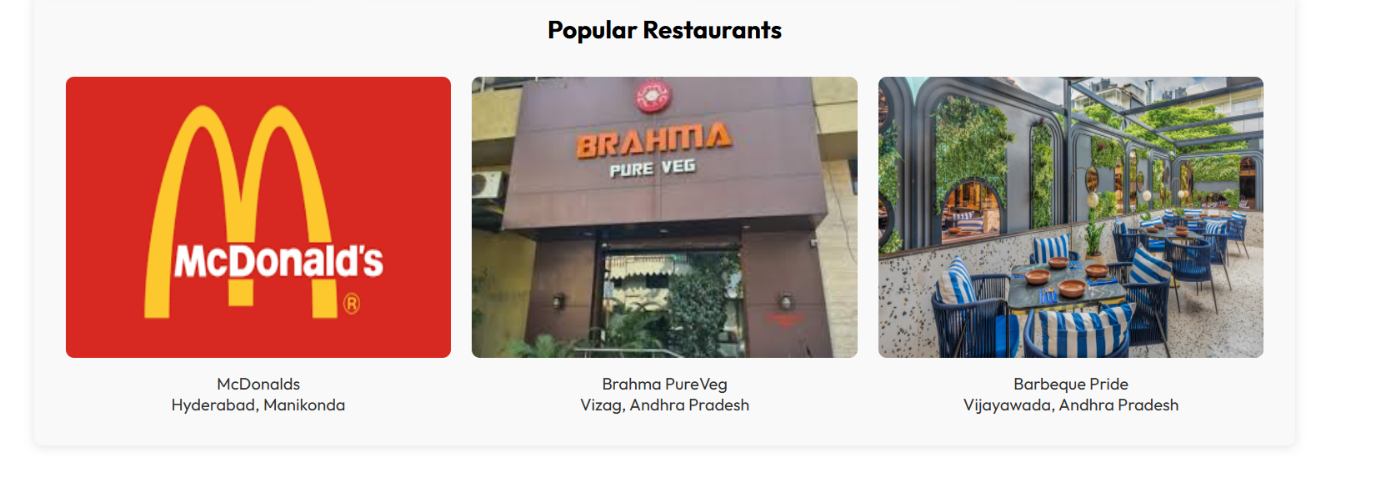
4. **Authentication Flow**:

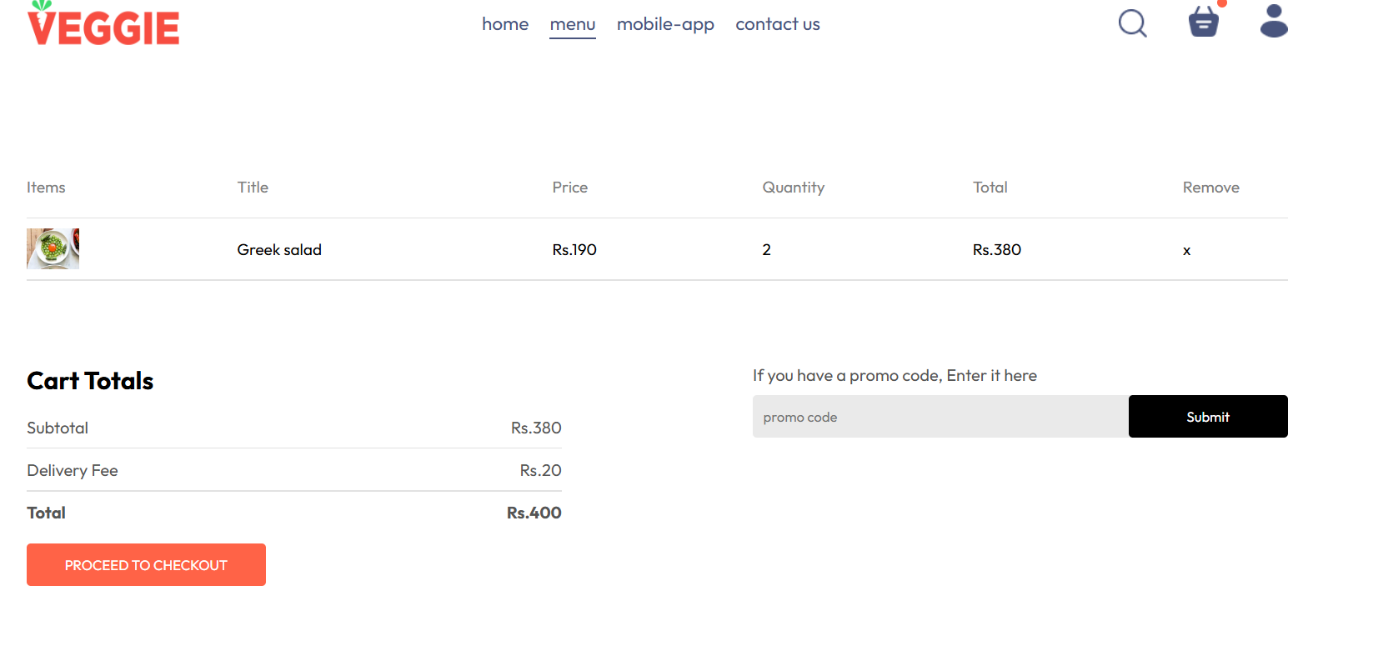
1. User submits login form.
2. Server checks credentials.
3. On success, server generates a JWT (JSON Web Token) and sends it back to the client.

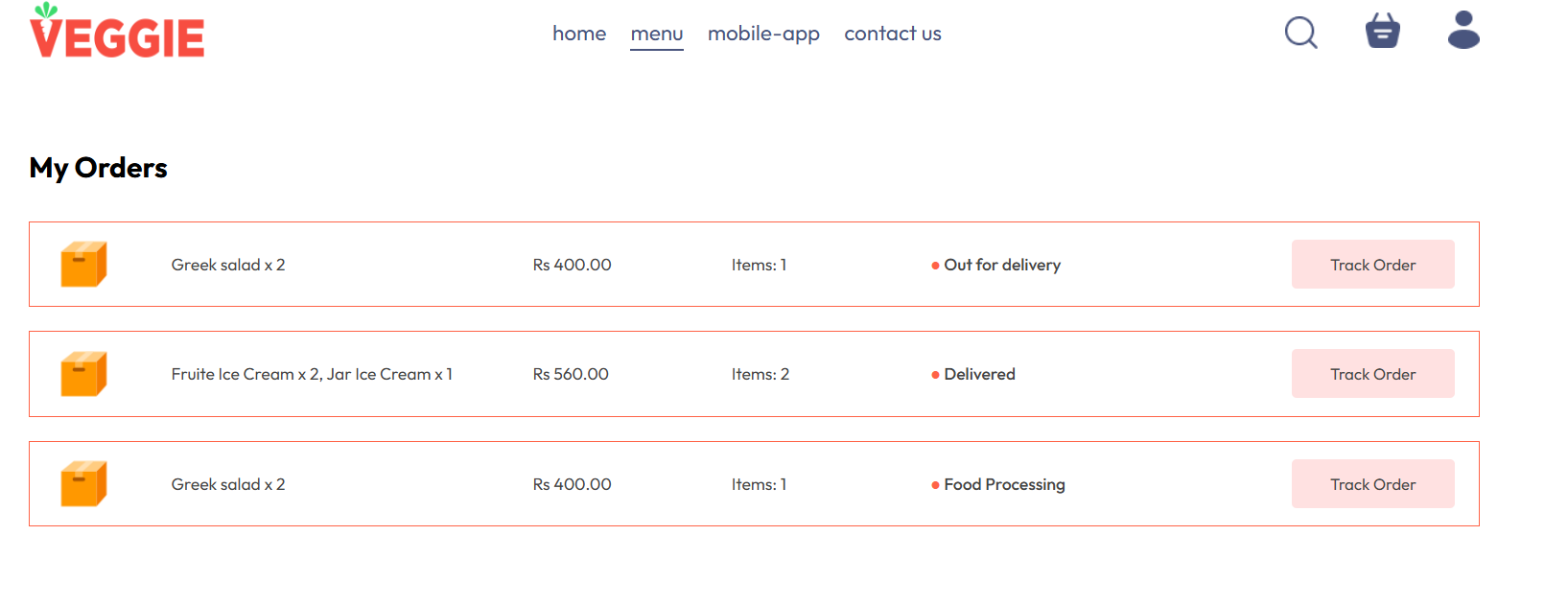
**9. User Interface**

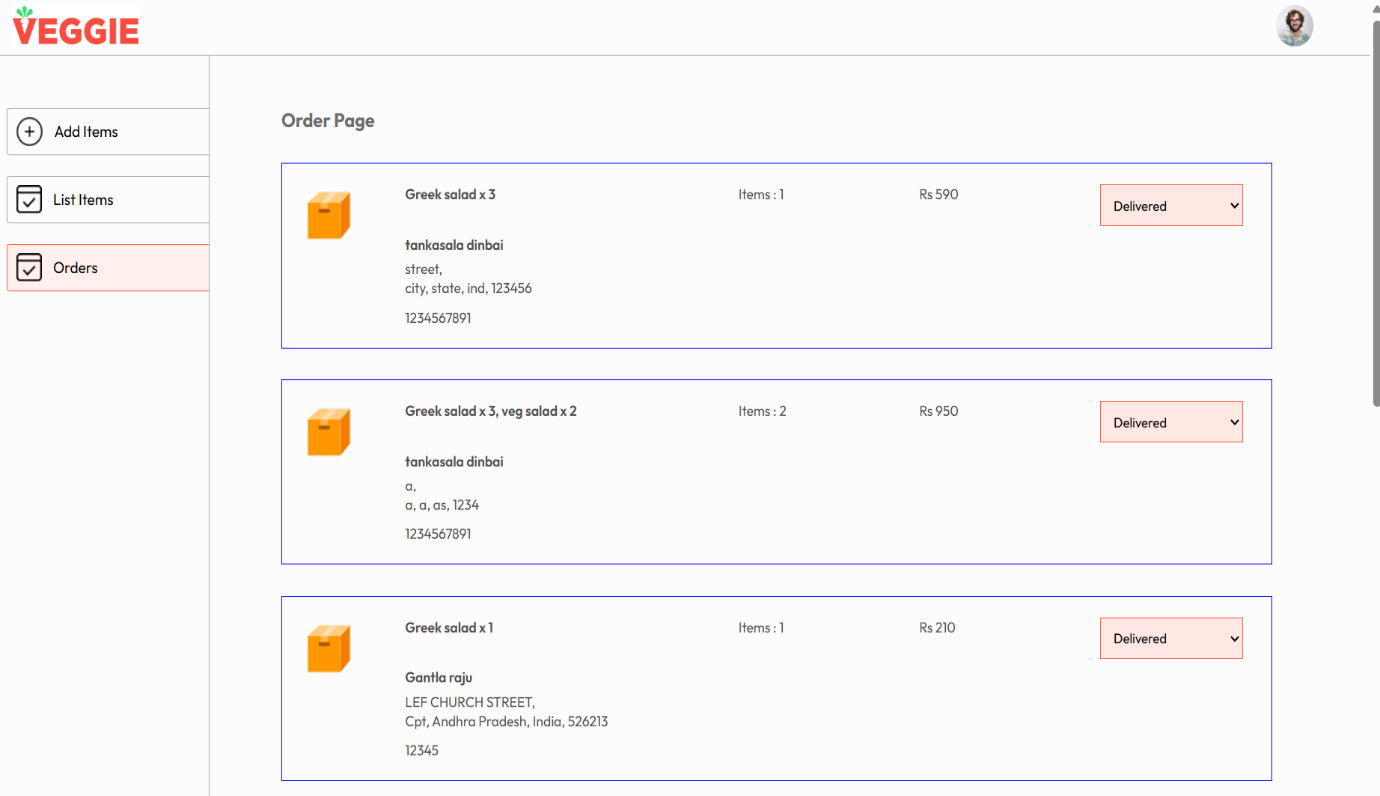
Screenshots or Demo











**10. Testing**

Outputs and Reports Testing

Features to be tested are: 1. Proper redirected login as per the user. 2. Validation of login and register

module. 3. View Shopping Cart and the Catalogue. 4. View information of various fields. 5. Validation of

booking form.

1. Black Box Testing Black box testing treats the software as a black box without the knowledge of

internal behavior. It aims to test the functionality according to the requirements. Thus, the

tester only inputs data and sees the output from the test object. This kind of testing requires

through test cases to be provided to the tester who then can simply verify that for a given input,

the output value is the same as the expected value specified in the test cases. 2. White Box

Testing White box testing is however, is when the tester has access to the internal data

structures, code and the algorithms. These methods include creating tests to satisfy some code

coverage criteria. For example, the test designer can create test to cause all statements in the

program to be executed at least once. Other examples of white box testing are mutation testing

and fault injection method.

2. Black box testing or functional validation test cases and results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test  code | Test case | Test Step | Expected result | Actual  result | ststus  (Pass  or  Fail) |
| Test1 | Check administrator  login | Go to Login  Page.Enter  valid username  and password.  Then Click on  “login” | Successful  Login | Login  successful | Pass |
| Test2 | Place order | Go to order  placing Page.  check the menu  details and  select menu  from it.  Then Click on  “place order” | Placed order  successfully | Placed  order  successfully | pass |
| Test3 | Check and verify  customer order. | Go to order  Page. Check the  order details  and verify it.  Then Click on  “verify order” | Display  message to  customer  that “your  order is  submitted” | Order is  submitted | pass |
| Test4 | Customer Payment | Go to payment  page. Select  payment option  then click on  pay option. | Display  message to  customer  that “your  order is  processed” | order is  processed | pass |

3.White box testing or functional validation test cases and results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test | Test case | Test Step | Expected  result | Actual  result | status  (Pass  or  Fail) |
| Test1 | Verify response  when a valid  username and  password is used | Check the valid  username and  password that  written in  respective fields. | Successful  Login | Login  Successful | pass |
| Test2 | Verify all decision  making statements | Check and verify  all decision  making  statements used  in coding. | Verify  successfully | Verify  successfully | Pass |
| Test3 | Verify all looping  statements | Check and verify  all looping  statements used  in coding. | Verify  successfully | Verify  successfully | pass |
| Test4 | Code optimization | Verify to remove  repeated and  extra code | Verify code  optimization | Verify code  optimization | Pass |

11.Conclusion

The main objective of the application is to help Computer Science students

understands the basics of Java, JavaScript and HTML. The following results have been achieved

A complete food ordering website / app using React JS, MongoDB, Express, Node JS and Stripe payment gateway. In this Full Stack Food delivery app project we will create the Frontend website, Admin Panel and Backend server. We will create the user authentication system so that anyone can create an account and login this food order website. We will create the shopping cart functionality so that user can add food items in their cart and order food from this app. We will also integrate the Stripe payment gateway to place the order and with online payment. Then we will create the order status update features also.