Summer Internship Project Report

Dhirubhai Ambani Institute of Information and Communication Technology.

M.Sc. (Information Technology)
2nd Semester

Title - Online Food Ordering System

Mayank Tailor(202312052)

Guided By
Dr. Shruti Bhilare

Table of content

1. Introduction

"Tomato" is a comprehensive food ordering platform designed for a particular restaurant. The platform offers a wide variety of dishes, which are showcased on the homepage. It features a categorized menu, allowing users to easily navigate through different types of food items. Users can add desired dishes to their cart and proceed to the payment page, where transactions are securely handled by Stripe.

The platform also includes a "My Orders" page, where users can track the status of their orders. The login and logout functionality ensures secure access to user accounts. Additionally, the admin panel provides restaurant administrators with the ability to manage food items, view and process incoming orders, and update customers on the status of their orders.

Terms	**Definition**
Online Food Display	Many types of food item pictures will be shown here for customers to make them understand what item they can choose.
Food Order‘s	They can make an order according to their needs by a few clicks.
Order Status	Customer and Admin can monitor order status like processing, delivering, or done.

1.1 Scope

The Online Food Ordering System will include two main actors: **User and Admin.** The User will be able to perform actions such as logging in, browsing food options, searching for food, adding items to cart, checking out, tracking orders, updating profile, posting reviews, and logging out. The Admin will be able to log in, view orders, accept orders, mark orders as out for delivery, manage food items, view past orders, and view earnings..

The proposed software product is the Online Food Ordering System (OFO) for the Food industry. It will be used to maintain various hotels and restaurants. Customers will use the system to give daily food orders to their needs if it is necessary for them. Customers can view the date-wise food collections or the status of the overall item of each restaurant in his/her area. The customer will be able to view their ordered items and manage them.

1.2 Definitions, Acronyms, and Abbreviations

- **User**: A person who uses the Online Food Ordering System to order food.
- **Admin**: The owner of a restaurant, cafe, or bakery who manages the Online Food Ordering System.
- **OFOS** - Online Food Ordering System
- **APIs** - Application programming interface
- **DBMS** - Database Management System
- **LAN** - Local Area Network
- **IP** - Internet Protocol

2. Overall Description

This section delineates the system's expected functionalities and limitations, outlining both supported and unsupported scenarios. It also highlights any assumptions made during the system's creation.

2.1 Product Perspective

The Online Food Ordering System will serve as a standalone system that interfaces with users through web browsers. It will also interact with a database to store user information, food items, orders, and other relevant data. The system may integrate with external payment gateways for processing payments.

The website must be available to anyone using a computer or a smart phone with internet connection.

2.2 Product Functions & Modules

The following are the major functions of the OFO software:

2.2.1 User Functions

1. **Login:** Users can log in to their accounts using their credentials.
2. **Browse Food Options:** Users can browse various food options available from different restaurants.
3. **Add to Cart:** Users can add specific items and their quantities to their cart.
4. **Checkout:** Users can proceed to the checkout page where they must enter their address and other details for delivery.
5. **Make Payment:** Users can make payment for their orders securely through the system.
6. **Track Orders:** Users can track the status of their orders, including order confirmation, preparation, and delivery.
7. **Update Profile:** Users can update their profile information, such as name, address, and contact details.
8. **Logout:** Users can log out of their accounts.

2.2.2 Admin Functions

1. **Login:** Admins can log in to their accounts using their credentials.
2. **View Orders:** Admins can view incoming orders on the homepage and manage them.
3. **Accept Orders:** Admins can accept incoming orders and start preparing them.
4. **Mark Orders as Out for Delivery:** Admins can mark orders as out for delivery once they are ready to be delivered.
5. **Manage Food Items:** Admins can manage the list of food items available for ordering, including adding, deleting, and updating items.
6. **View Past Orders:** Admins can view past orders to track order history.
7. **View Earnings:** Admins can view earnings from orders.

2.2.3 Order Processing Functions

1. **Accept Order:** Admins can accept incoming orders.
2. **Deliver Item:** Admins can mark an order as delivered.
3. **Place Order**: Users can place an order for food items.
4. **Pay Bill:** Users can pay the bill for their order.

2.3 Modules

–Admin
–User Module
–Contact Us Module
–Payment gateway

3. Specific Requirements

3.1 External Interfaces

3.1.1 User Interface

The user interface must be intuitive and user-friendly, allowing users to easily navigate and use the system.
Users should be able to access the system via web browsers and mobile applications.

3.1.2 Hardware Interfaces

The system must be compatible with standard hardware configurations for web browsing and mobile devices.

3.1.3 Software Interfaces

To the end-user there is no need for any extra software to be installed. It is to be mentioned that the user needs JavaScript enabled browsers to run the system. For an OS, it has no boundary or strict rules, it is platform independent and can run smoothly in any OS like Windows, MacOS, Linux, Android etc. However, through the channel the cryptography should be maintained through the whole system as the user can access it through the internet also from anywhere.

.

3.2 Functional Requirements

3.2.1 User Functions

- Users should be able to log in and log out of their accounts.
- Users should be able to browse food options and search for specific items.
- Users should be able to add items to their shopping cart and specify quantities.
- Users should be able to check out, enter delivery details, and place orders.
- Users should be able to track the status of their orders.
- Users should be able to update their profile information.
- Users should be able to post reviews and ratings for food items.
- Non-logged-in users should be prompted to login to access checkout.

3.2.2 Admin Functions

- Admins should be able to view incoming orders and manage them.
- Admins should be able to accept orders and mark them as out for delivery.
- Admins should be able to manage food items in the inventory, including adding, deleting, and updating items.
- Admins should be able to view past orders and earnings.
- Admins should be able to accept or reject items for inclusion in the inventory.

- Admins should be able to indicate when an item is ready for delivery.

3.3 Performance Requirements

- The system should be able to handle a large number of concurrent users.
- The system should provide fast response times for user interactions.
- The system should be available and accessible 24/7, with minimal downtime for maintenance.

3.4 Design Constraints

- The user interface should be responsive and compatible with a variety of devices and screen sizes.

3.5 Non-Functional Requirements

3.5.1 device capacity Requirements

Server software does not require any special hardware other than the minimum hardware required for running enterprise OS. Extra disk storage will be required for archives and electronic documents. Increases of memory enables efficient query processing, which is required for quick bibliographic search. Two server grade processors with clock speed core i3, 2.3 GHz, at least 4GB RAM and 1TB hard disk is recommended for the server. Client machine with recommended hardware required for desktop operating system and web browser (with open JavaScript enable).

3.5.2 Safety Requirements

As per Restaurant and Hotels work place safety rules and the Restaurant and Hotels where the server is supposed to be placed and the monitoring people.

3.5.3 Security Requirements

Each time there is a security violation, the log file will be updated with the login, date, and time. Again, high level cryptography and checking should be kept to make it more secure. However, while email or request from any unwanted customer the request should drop and let that user know about the fault.

3.5.4 Performance Requirements

Two server grade processors with clock speed core i3, 2.3 GHz, at least 4GB RAM and a 500 GB hard disk is recommended for the server. Client machine with recommended hardware required for desktop operating system and web browser (with open JavaScript enabled).

4. Technologies and Frameworks

4.1 Technologies

The "Tomato" project leveraged several modern technologies and frameworks to build a robust and efficient food delivery web app:

4.1.1 Front-End

Vite React Framework: Used to create a fast and modular user interface.

JavaScript: The primary programming language for front-end development.

CSS3 and HTML5: Used for styling and structuring web pages.

Axios: A promise-based HTTP client for making API requests.

Toastify: For displaying notifications and alerts.

4.1.2 Back-End

Node.js: A JavaScript runtime for building the server-side of the application.

Express.js: A web application framework for Node.js, used for client/server connectivity and handling routing and middleware.

MongoDB Atlas: A cloud-based NoSQL database for storing application data.

Mongoose: An ODM (Object Data Modeling) library for MongoDB, used to define schemas and interact with the database.

JsonWebToken: Used to create and verify web tokens for authentication.

Bcrypt: A library for hashing passwords to ensure secure user authentication.

CORS (Cross-Origin Resource Sharing): Provides permissions for the front end to connect to the backend.

Body-Parser: Middleware for parsing incoming request bodies.

Multer: Middleware for handling image uploads and storage.

Stripe: A payment gateway for processing payments securely.

Validator: Used to validate email addresses and other input data.

Nodemon: Automatically restarts the server when changes are made to the codebase.

Envdev: For managing environment variables and configuration.

4.2 APIs

Multiple APIs were developed to handle various functionalities within the app. Some key APIs include:

4.2.1 User Authentication APIs:

- POST /api/user/login: Handles user login.
- POST /api/user/logout: Handles user logout.
- POST /api/user/register: Registers new users.

4.2.2 Food Item Management APIs:

- GET /api/food/list: Fetches all available food items.
- POST /api/food/add: Adds a new food item (admin only).
- POST /api/foods/remove: Deletes a food item (admin only).

4.2.3 Order Management APIs:

- POST /api/order/place: Place a new order.
- GET /api/orders/list: Fetches order details by order ID.
- GET /api/orders/user/verify: Verify all orders.
- POST /api/orders/:id/status: Updates the status of an order (admin only).
- POST /api/orders/:id/delete: delete an order (admin only).

4.2.4 Stripe Payment API:

- POST /api/payments: Handles payment processing via Stripe.

4.2.5 Cart API:

- GET /api/food/get: get all available cart items.
- POST /api/food/add: Adds a new food item to the cart (user only).
- POST /api/foods/remove: Deletes a food item to the cart (user only).

5. Design

5.1 Design Patterns

While specific design patterns were not explicitly identified, the project incorporated several common web development practices:

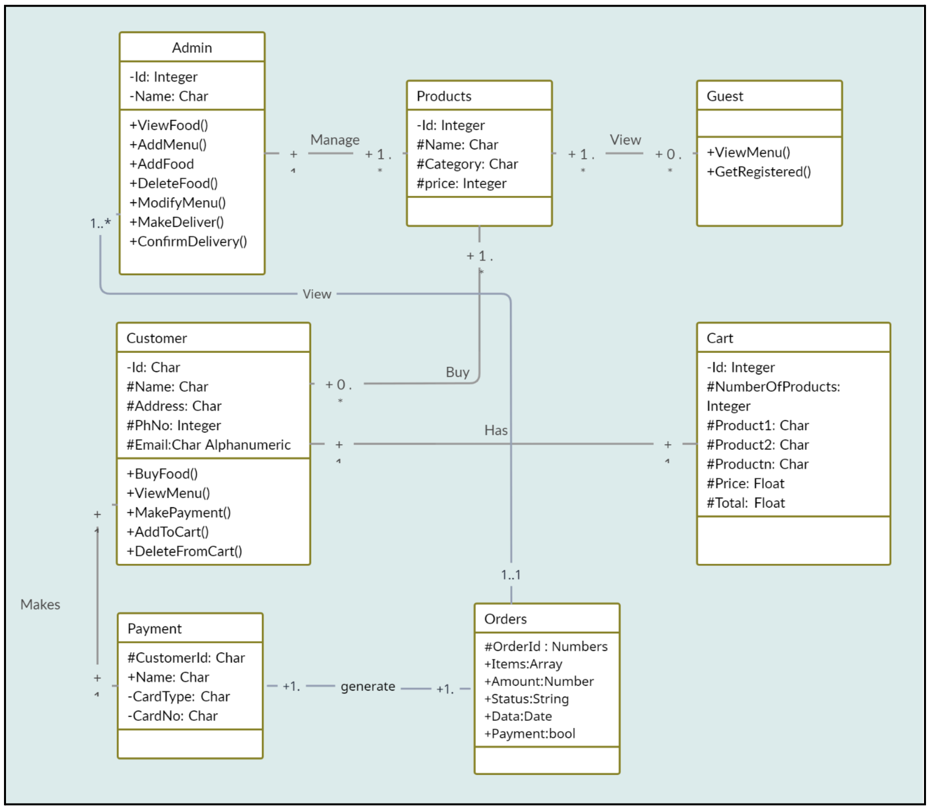
- **Model-View-Controller (MVC):** This architectural pattern was used to separate concerns and organize the codebase, making it easier to manage and maintain.
- **Responsive Design:** The app was designed to be responsive, ensuring a seamless user experience across different devices and screen sizes.
- **Reusable Components:** Reusable components were created to streamline development and ensure consistency across the app.

5.2 Diagrams

To further clarify the design, various diagrams were created, including:

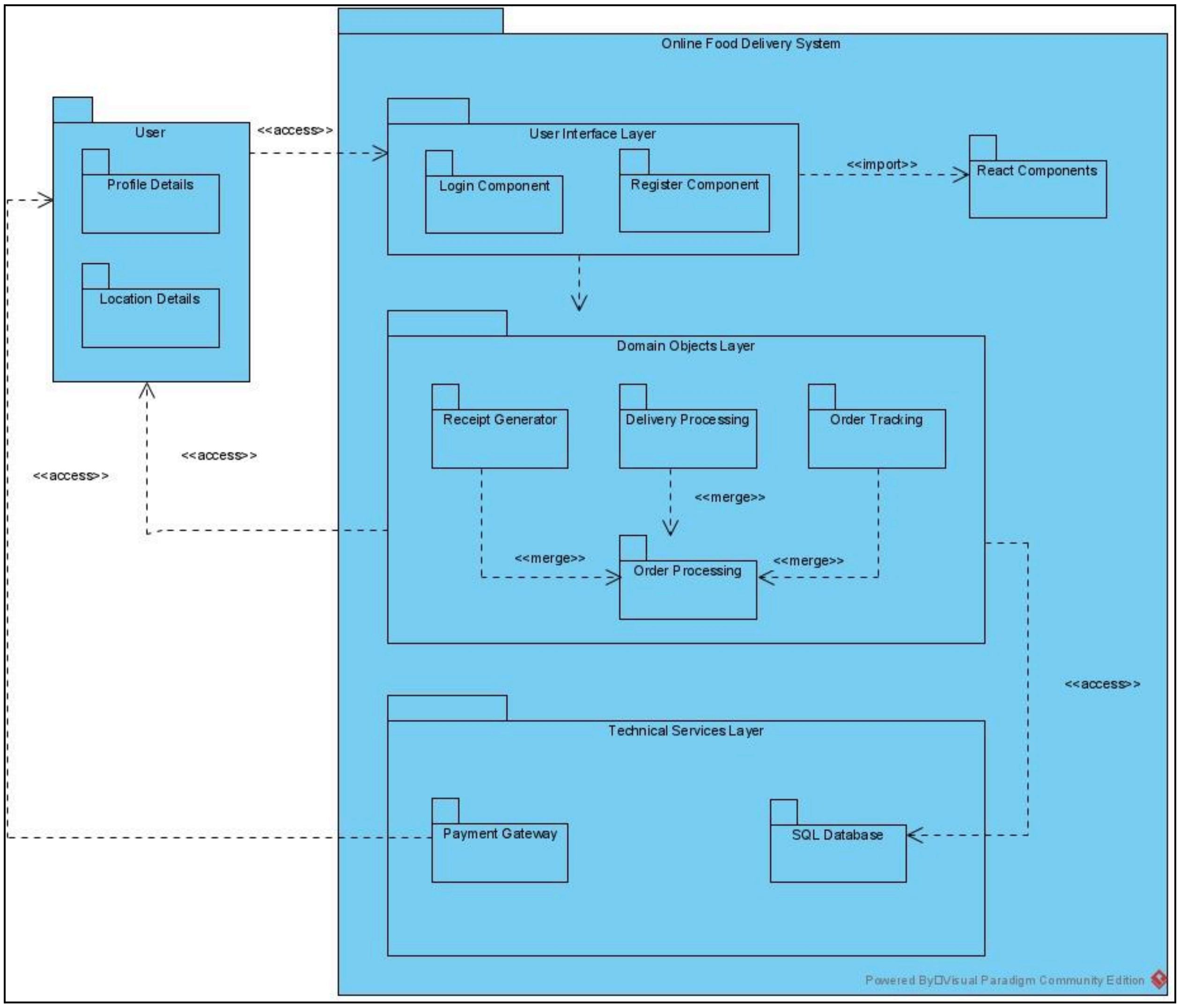
5.2.1 Class Diagrams

These diagrams illustrated the structure of the app, showing the classes and their relationships.



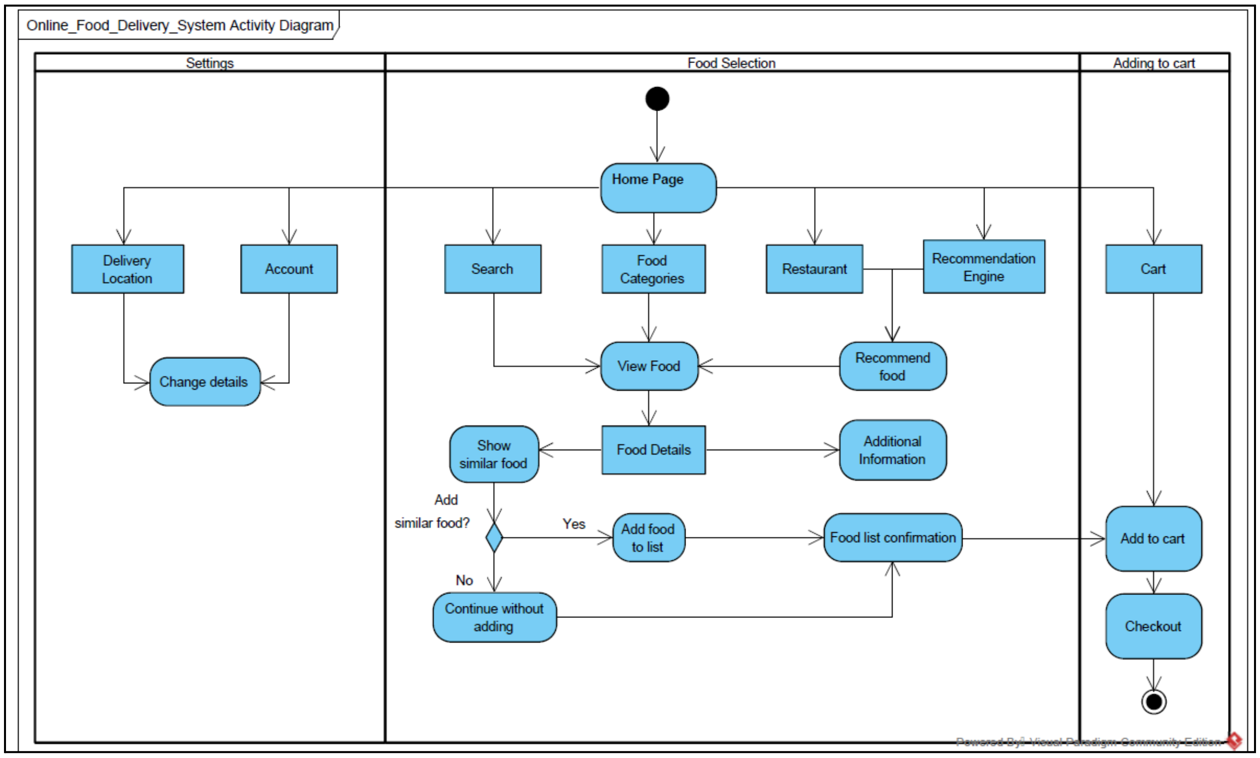
5.2.2 Package Diagrams

Package diagrams depicted the organization of the codebase into different modules and packages.



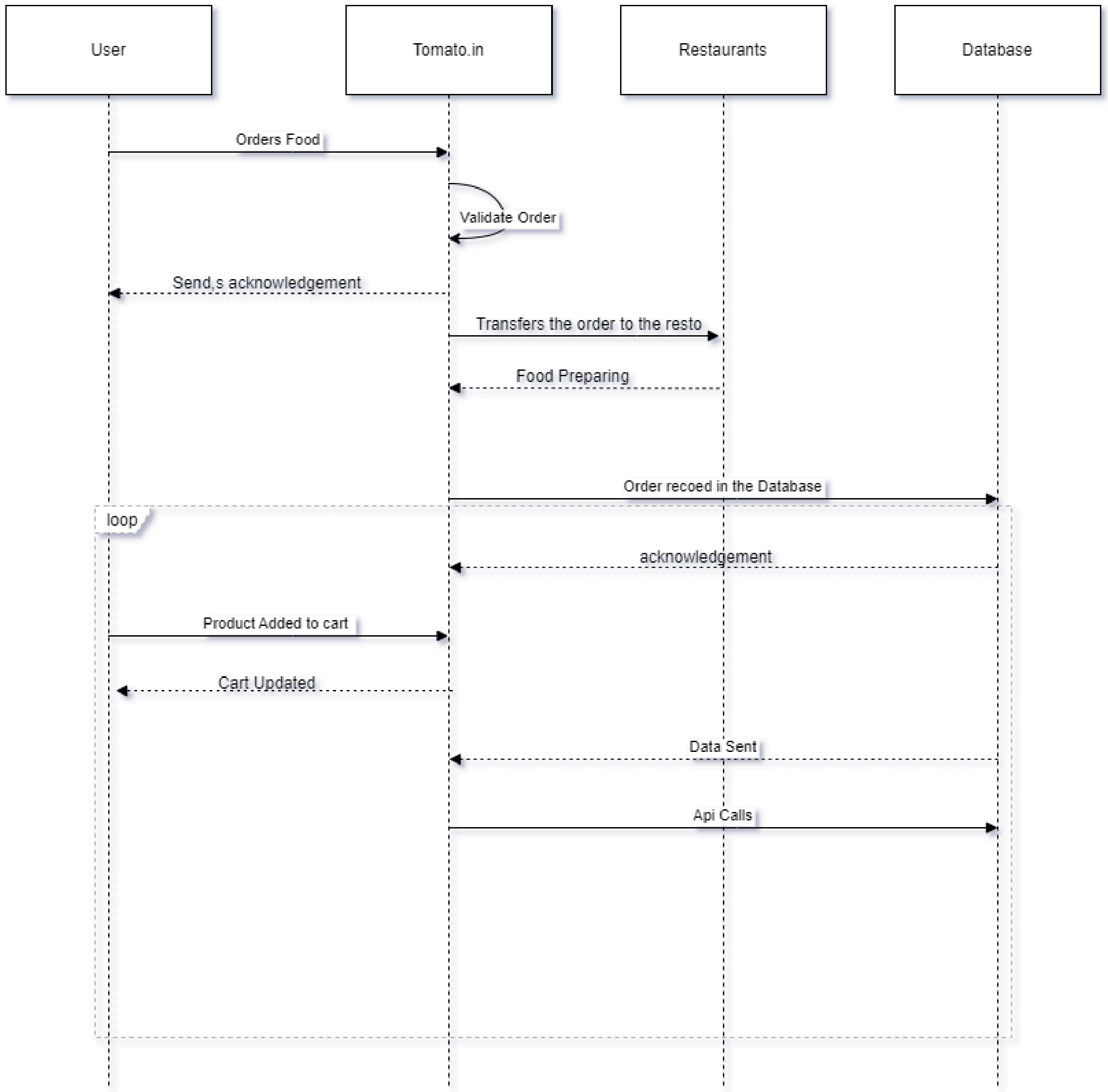
5.2.3 Activity Diagrams

As mentioned earlier, these diagrams detailed the workflows of key processes



5.2.4 Sequence Diagram

As mentioned earlier, these diagram detailed the workflows of key processes



6. Testing

6.1 Testing Methods

To ensure the reliability and performance of the "Tomato" web application, various testing methods were employed:

1. **Unit Testing:**
   - Individual components and functions were tested to ensure they work as expected.
2. **Integration Testing:**
   - Tested the interaction between different modules and components of the application.
   - Ensured that API endpoints correctly interacted with the database and other services.
   - Used tools like Thunder Client for testing API endpoints and verifying responses.
3. **End-to-End (E2E) Testing:**
   - Simulated real user scenarios to test the entire application flow from start to finish.
   - Used testing frameworks like Cypress to automate E2E tests and ensure the application behaves correctly in a production-like environment.
4. **Performance Testing:**
   - Assessed the application's performance under various load conditions to ensure it can handle a large number of concurrent users.

6.2 Tools

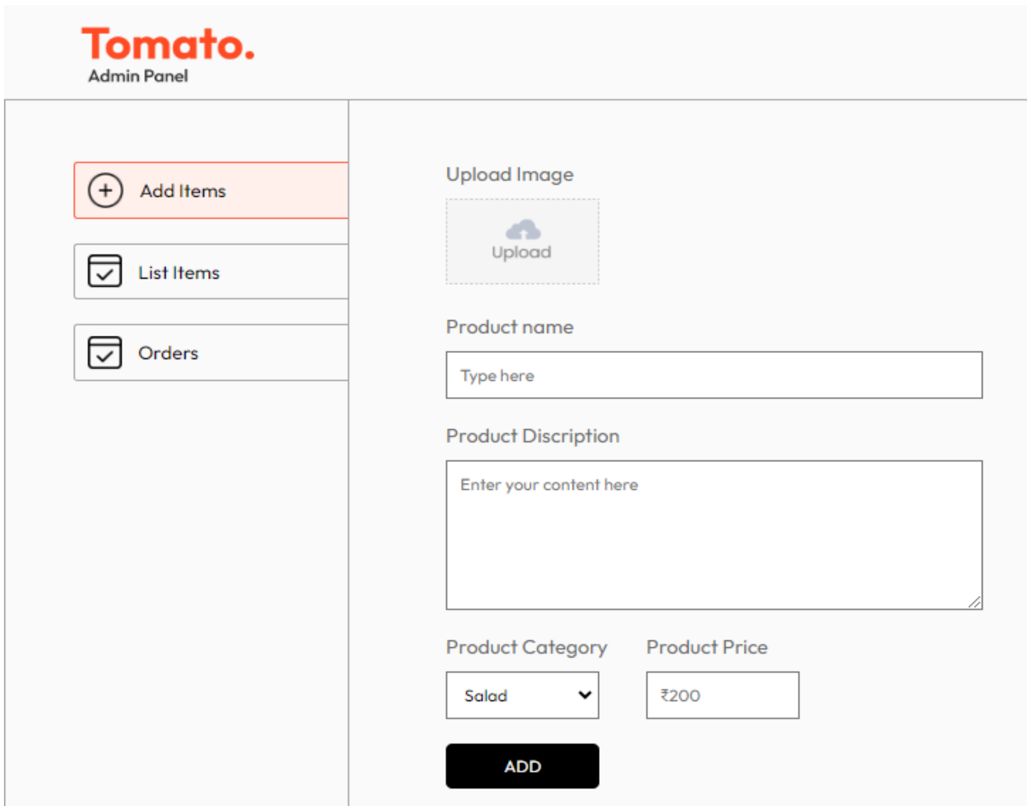
Several tools were utilized throughout the testing process to streamline and automate testing activities:

- **Jest:** A JavaScript testing framework for running unit tests.
- **Thunder Client:** A tool for testing APIs by making HTTP requests and verifying responses.
- **Cypress:** An end-to-end testing framework for web applications.

7. Snapshot

7.1 Admin Panel

Add Items:



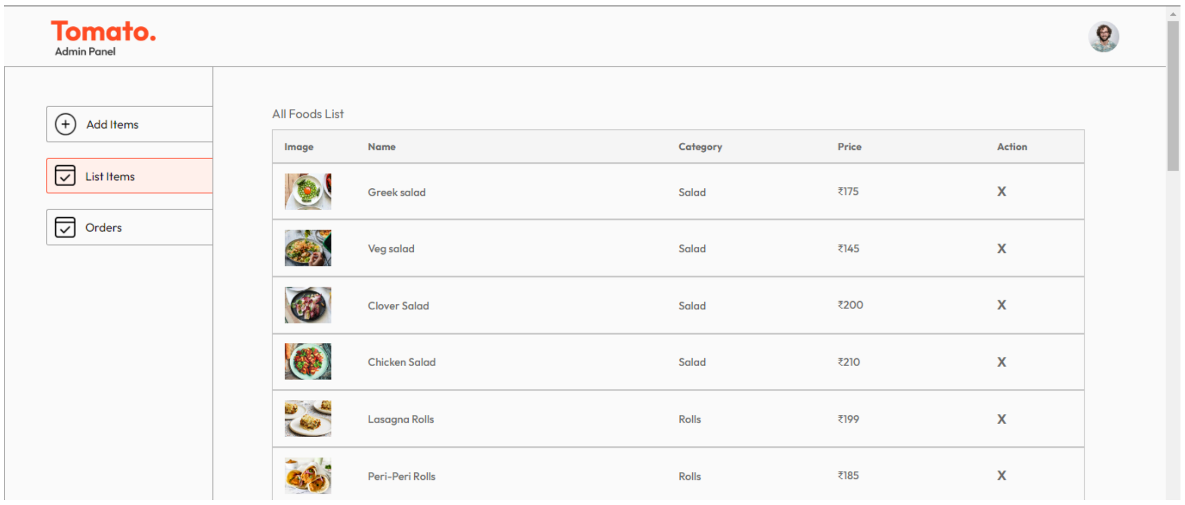
Type here

Enter your content here

Salad

₹200

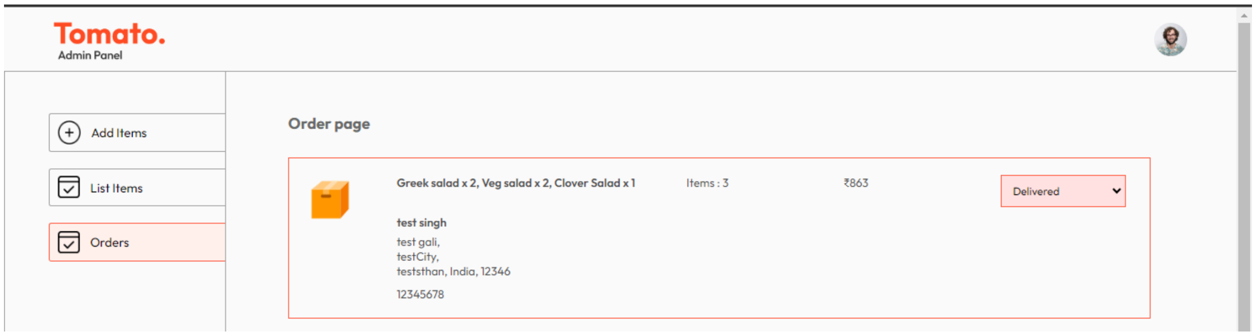
List Items :



All Foods List

Image	Name	Category	Price	Action
	Greek salad	Salad	₹175	X
	Veg salad	Salad	₹145	X
	Clover Salad	Salad	₹200	X
	Chicken Salad	Salad	₹210	X
	Lasagna Rolls	Rolls	₹199	X
	Peri-Peri Rolls	Rolls	₹185	X

Check Orders:

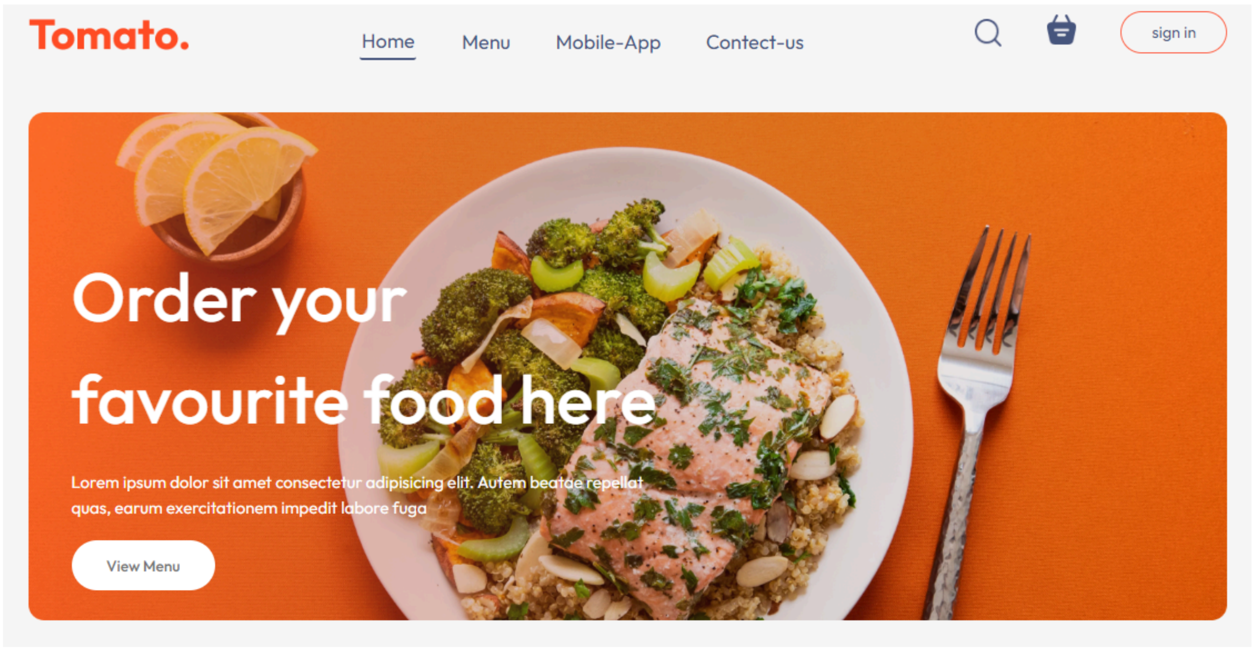


Greek salad x 2, Veg salad x 2, Clover Salad x 1 Items : 3 ₹863 Delivered

test singh
test gali,
testCity,
teststhan, India, 12346
12345678

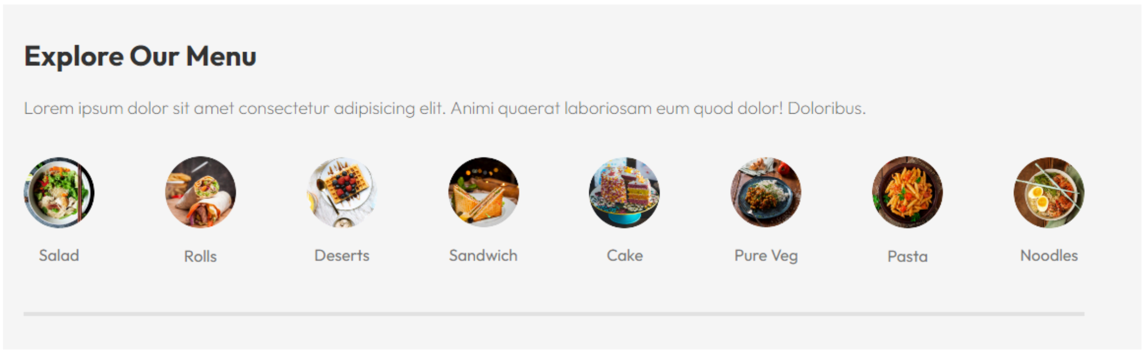
7.2 Tomato.in

Home Page:



Lorem ipsum dolor sit amet consectetur adipisicing elit. Autem beatae repellat quas, earum exercitationem impedit labore fuga

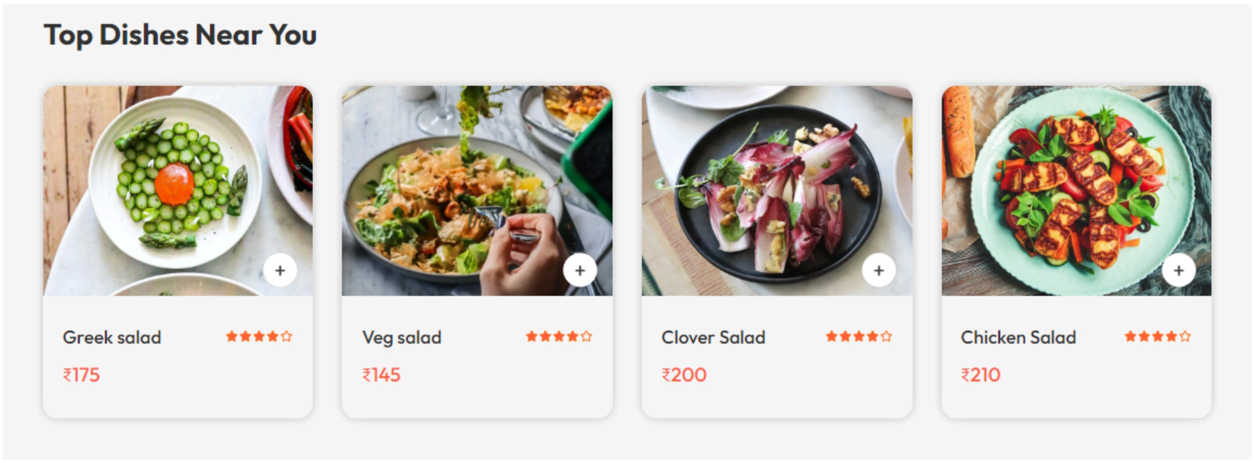
Menu:



Explore Our Menu

Lorem ipsum dolor sit amet consectetur adipisicing elit. Animi quaerat laboriosam eum quod dolor! Doloribus.

Food Items:


Top Dishes Near You

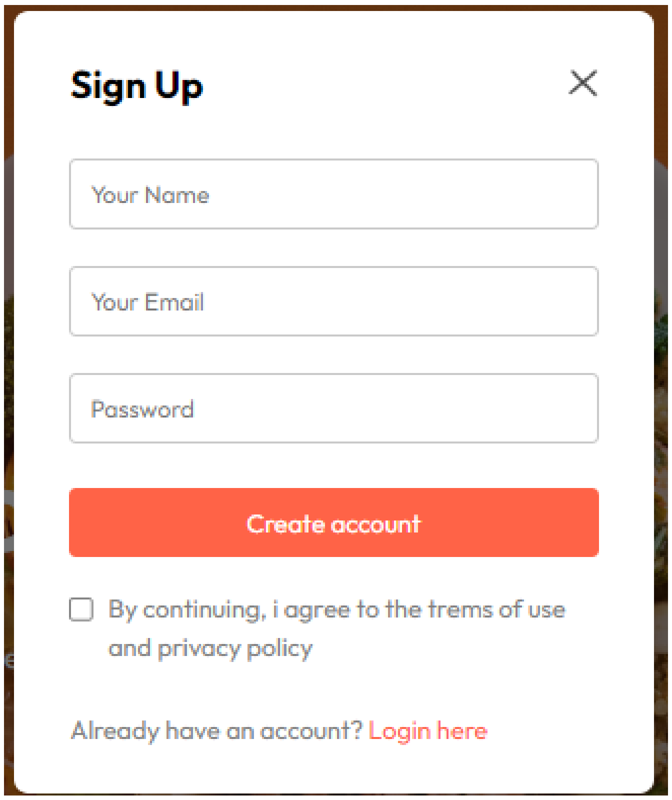
Greek salad ₹175

Veg salad ₹145

Clover Salad ₹200

Chicken Salad ₹210

Signup page:


Sign Up

Your Name

Your Email

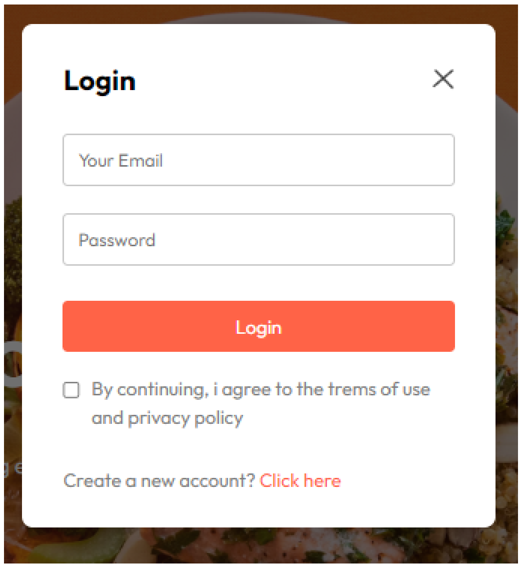
Password

Create account

By continuing, i agree to the trems of use and privacy policy

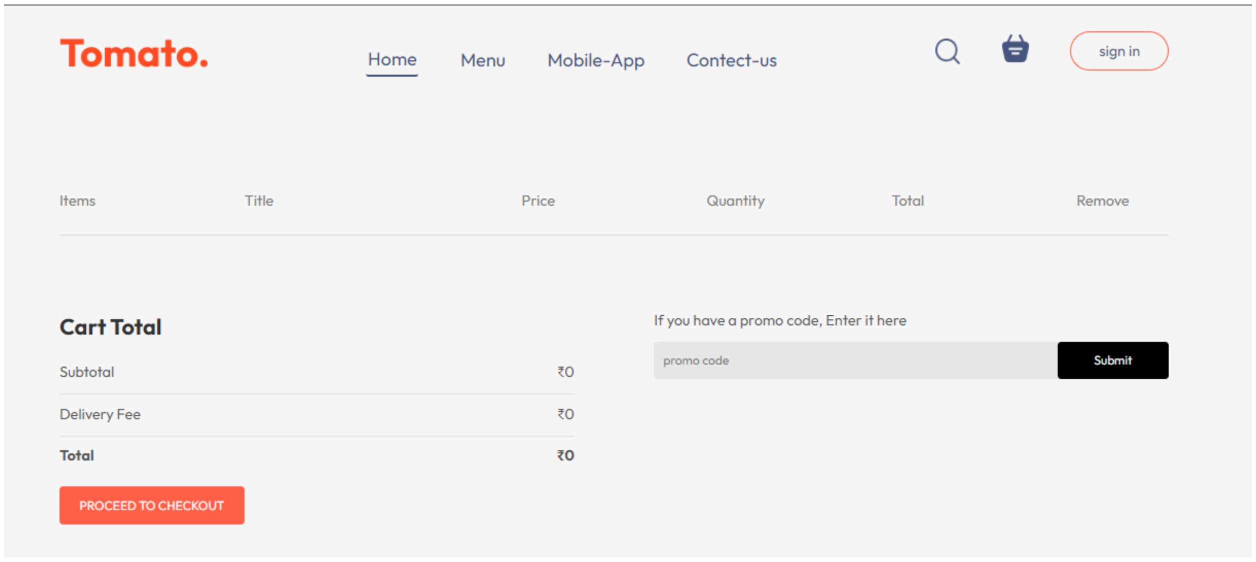
Already have an account? Login here

Login - page:



Login
Your Email
Password
Login
By continuing, i agree to the trems of use and privacy policy
Create a new account? Click here

Cart:

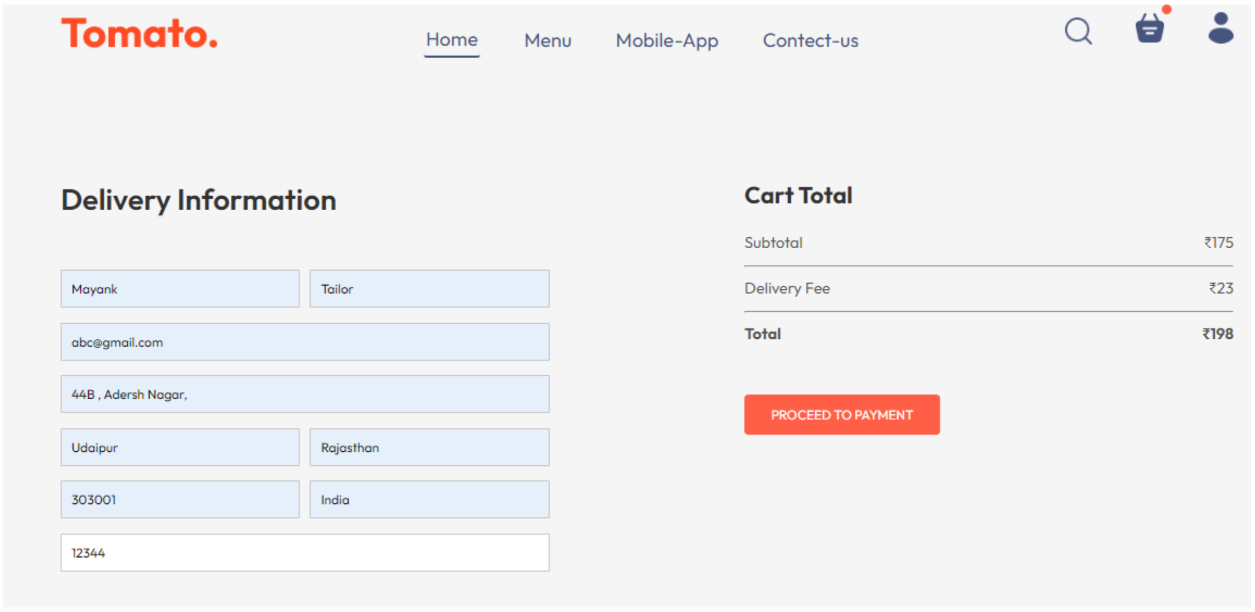


Items	Title	Price	Quantity	Total	Remove

Cart Total

Subtotal	₹0
Delivery Fee	₹0
Total	**₹0**

Order Page:



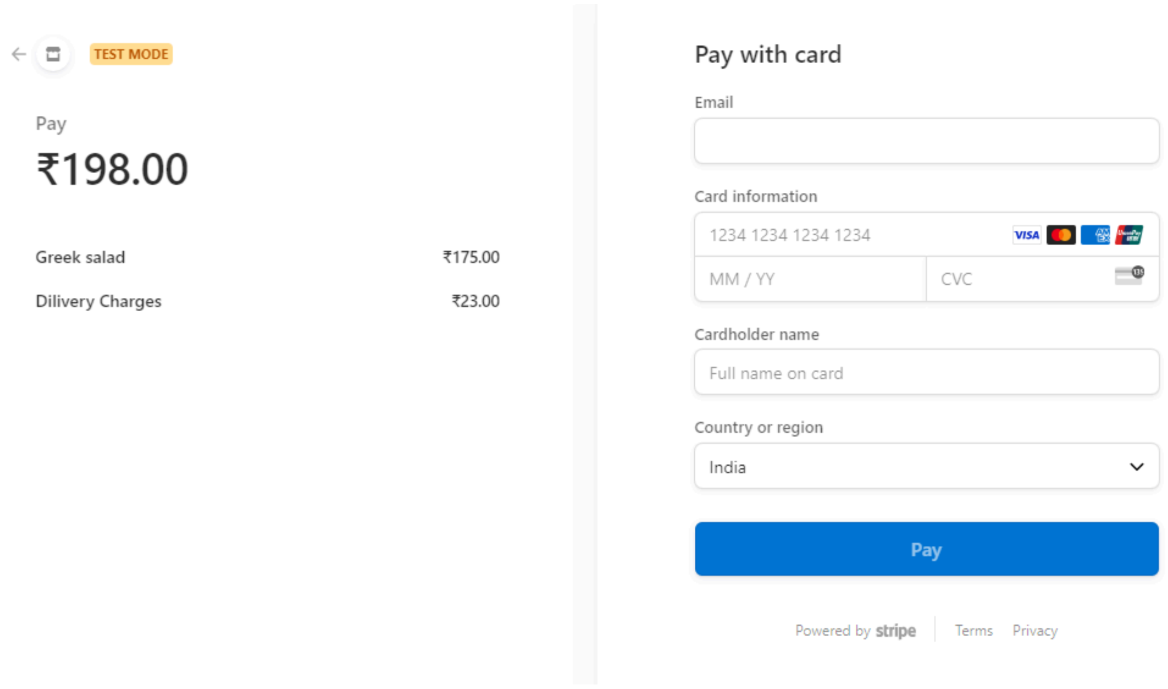
Delivery Information

Mayank	Tailor
abc@gmail.com	
44B , Adersh Nagar,	
Udaipur	Rajasthan
303001	India
12344	

Cart Total

Subtotal	₹175
Delivery Fee	₹23
Total	₹198

Payment Page:



Pay
₹198.00

Greek salad	₹175.00
Dilivery Charges	₹23.00

Pay with card

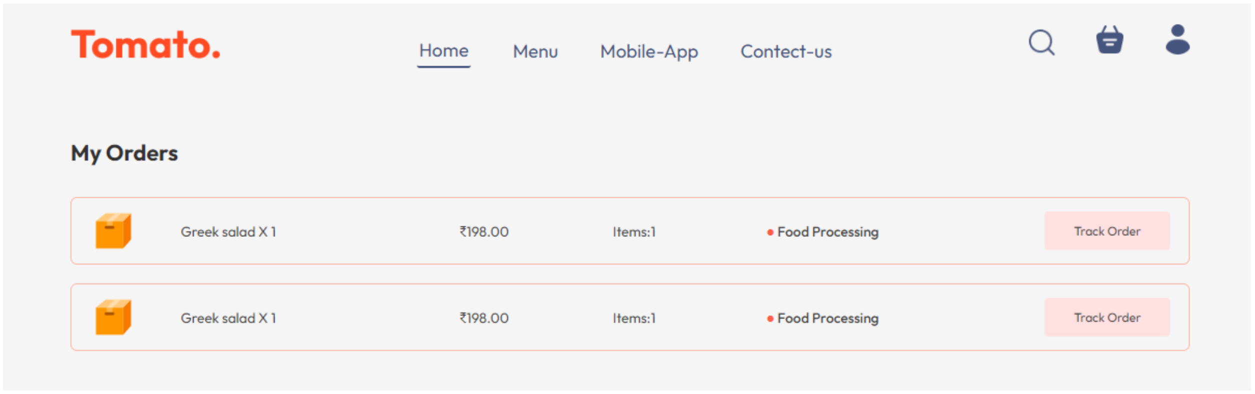
Email:

Card information: 1234 1234 1234 1234 | MM / YY | CVC

Cardholder name: Full name on card

Country or region: India

My orders Page:



My Orders

Greek salad X 1	₹198.00	Items:1	● Food Processing	Track Order
Greek salad X 1	₹198.00	Items:1	● Food Processing	Track Order

7.3 Github project Link

https://github.com/tailormayank/Tomato

7.4 Live Demo

7.4.1 Tomato.in

https://tomato-frontend-r1pc.onrender.com/

7.4.2 Admin Panel

https://tomato-admin-uelq.onrender.com

8. Summery

The "Tomato" food delivery web application is designed to provide a seamless and efficient platform for users to order food online and for admins to manage their restaurant operations. With a user-friendly interface and robust functionality, "Tomato" aims to enhance the overall experience of ordering food online.

By incorporating features such as user login, browsing food options, searching for specific items, adding items to the cart, checking out, tracking orders, updating profiles, and posting reviews, "Tomato" caters to the needs of both users and admins. Admins are empowered to manage orders, accept payments, manage inventory, and track earnings, ensuring smooth operations and customer satisfaction.

With a focus on performance, scalability, and maintainability, "Tomato" is poised to meet the evolving needs of its users and provide a reliable and efficient platform for online food ordering. The integration of modern technologies and frameworks ensures that the application is robust, secure, and capable of handling a large number of concurrent users. "Tomato" is set to redefine the online food ordering experience, making it easier and more enjoyable for customers to get their favorite meals delivered to their doorsteps.

\

9. Lessons Learned

Effective Use of Modern Technologies:

- Leveraging modern technologies such as the Vite React framework, MongoDB Atlas, and Node.js facilitated the development of a robust and scalable application. These technologies provided the necessary tools to build a fast, responsive, and maintainable web app.

User-Centric Design:

- Focusing on user experience from the beginning ensured that the application was intuitive and easy to navigate. User feedback was instrumental in refining the interface and features, leading to a more satisfying user experience.

Continuous Learning and Adaptation:

- The project provided numerous opportunities for learning and skill development. Staying updated with the latest developments in web technologies and being open to new ideas and approaches was essential for overcoming challenges and improving the application.