ONLINE FOOD DELIVERY WEB APPLICATION (MERN STACK)

MINI PROJECT REPORT

Submitted by

Rathinavel pandian C (23205041)

Sangezhil M (23205043)

Udhaya kumar O (23205055)

Yuvaraj K (23205060)

in partial fulfilment for the award of the degree

of

Postgraduate

in

Master of Computer Applications
Rathinam Technical Campus

Eachanari – 641021

An Autonomous Institution

Affiliated to Anna University, Chennai – 600 025

DEC – 2024

ANNA UNIVERSITY, CHENNAI

BONAFIDE CERTIFICATE

Certified that this Report titled **ONLINE FOOD DELIVERY WEB APPLICATION** is the bonafide work of **Rathinavel pandian.C (23205041)**, **Sangezhil.M (23205043), Udhaya kumar.O (23205055)**, **Yuvaraj K (23205060)** who carried out the work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other phase 1 report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr.A.B.Arockia Christopher

Professor and Head and Supervisor

Master of Computer Applications

Rathinam Technical Campus

Eachanari,Coimbatore

Anna university, Chennai.

Submitted for the Mini-project report Viva – Voce examination held on ……………

Internal Examiner **External Examiner**

ABSTRACT

Online Food Delivery Web Application project

The Online Food Delivery Web Application project aims to create a comprehensive platform for the food service industry, utilizing the MERN stack (MongoDB, Express.js, React.js, Node.js) to enhance both customer and administrative functionalities. This application features a customer site that includes user login, an "Add to Cart" option, and a payment page, allowing users to easily browse menus and place orders.Customers can register for an account, log in securely, and select multiple items to add to their cart. They have the option to choose between online payment and cash on delivery. The application provides a real-time experience, enabling customers to view their order history, track the status of their current orders, and receive timely confirmations.

On the administrative side, the application includes a dedicated admin interface for managing the food menu and processing orders. Admins can add new food items, update existing listings, and oversee order tracking to ensure timely fulfillment. This functionality enhances operational efficiency and improves customer satisfaction.This project showcases the transformative potential of web applications in food service delivery, illustrating how the MERN stack can facilitate seamless interactions for both customers and restaurant administrators in a dynamic business environment.

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of this project depends largely on the encouragement and guidelines of many others. We take this opportunity to praise the almighty and express our gratitude to the people who have been instrumental in the successful completion of our project.

We wish to acknowledge with thanks for the excellent encouragement given by the management of our college and we thank **Dr. B.Nagaraj, M.E., Ph.D., PDF (Italy) , CBO** for providing us with a plethora of facilities in the campus to complete our project successfully.

We wish to express our hearty thanks to **Dr. K.Geetha, M.E., Ph.D., Principal of our college**, for her constant motivation regarding our internship towards project.

Our heartfelt thanks go to **Dr. A.B. Arockia Christopher, M.E., Ph.D., Professor & Head of Department - MCA**, who not only served as our **project guide** but also provided tremendous support and valuable insights throughout the completion of our project. His guidance, patience, and encouragement have been vital in ensuring the success of this endeavor.

Finally, we would like to extend our deepest gratitude to our parents, friends, and faculty members for their constant support, encouragement, and belief in us throughout the course of this project. Their valuable guidance and assistance have been vital in the achievement of our goals.

We sincerely appreciate the help and contributions of everyone who made this project a success.

Rathinavel pandian C

Sangezhil M

Udhaya kumar O

Yuvaraj K

CONTENTS

LIST OF FIGURES

CHAPTER-1   INTRODUCTION

1.1 Introduction

Building modern and dynamic online apps now necessitates a strong grasp of web development. The MERN stack is a well-known and powerful mix of technologies that enables programmers to create extremely scalable and successful online applications. In this introduction, the components of the MERN stack will be covered, as well as their significance in web development.

The MERN stack is comprised of MongoDB, Express, React, and Node.js. Each component is critical to the development process and aids in the overall operation and execution of the web application.

1. **MongoDB:** MongoDB is a NoSQL database that employs an adaptive document-based data model. It saves data as JSON-like documents, making it simple to integrate and alter data within the application. Because of its scalability and ability to handle massive amounts of data, MongoDB is the best solution for web applications that require dynamic data storage.

2. Express.js: Express.js is a simple and versatile Node.js web application framework. It provides a comprehensive set of API and web app development tools and information.Express.js makes it simpler to manage routes, process requests and responses, and build middleware.It is extremely adaptable and effective for developing web apps due to its lightweight and modular architecture.

3.React.js: React.js is a JavaScript library for developing user interfaces. It allows for the design of reusable UI components as well as their effective updating and rendering in response to data changes. React.js' component-based architecture promotes reuse, modularity, and maintainability. It also has a virtual DOM (Document Object Model) for increased performance and rendering.

4. Node.js: Node.js is a JavaScript server-side runtime environment that allows programmers to run JavaScript code outside of a web browser. Its event-driven, non-blocking design makes it exceptionally scalable and excellent at managing several requests at once. Because it provides server-side scripting, file system operations, and network connectivity, Node.js is an excellent choice for building the backend of websites.

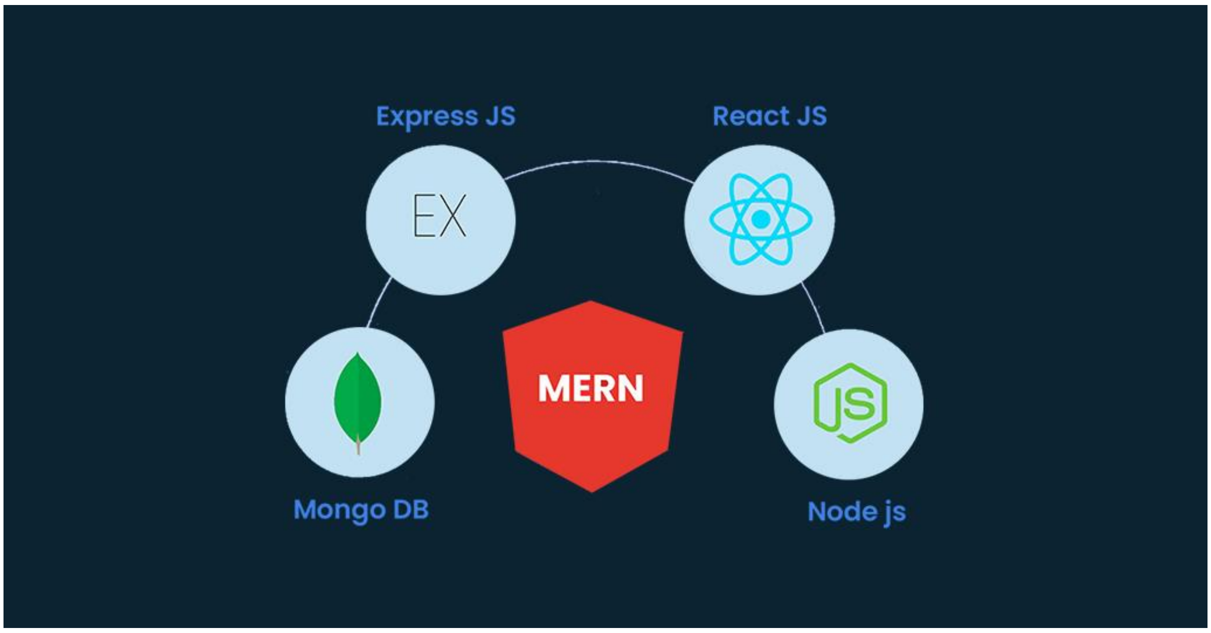


Image 1: MERN stack

These technologies work together to create a comprehensive web development solution.MongoDB handles the database layer, Express.js handles the backend and routing, React.js

handles the frontend user interface, and Node.js handles the server-side runtime environment.The MERN stack provides various advantages to web developers. It provides a single

JavaScript-based development environment that allows frontend and backend components to connect seamlessly.

Finally, the MERN stack offers a stable and fast platform for web development. It integrates MongoDB, Express.js, React.js, and Node.js to develop full-stack, scalable online apps.

1.2 SCOPE OF THE PROJECT

The Online Food Delivery Web Application project aims to create a user-friendly platform for both customers and restaurant administrators, utilizing the MERN stack (MongoDB, Express.js, React.js, Node.js). Customers will be able to easily browse menus, add items to their cart, and securely complete payments via online options or cash on delivery.

They can also track the real-time status of their orders and view their order history. On the admin side, the application will allow restaurant staff to manage the food menu, process customer orders, and track performance through a dedicated dashboard.

The system will ensure smooth interactions through features like secure user authentication, order management, and payment processing. With a focus on ease of use, security, and operational efficiency, this project aims to enhance the food delivery experience for both customers and restaurant administrators.

1.3 OBJECTIVE OF THE PROJECT

The main goal of this project is to create an easy-to-use system for managing food delivery orders. The system will help administrators efficiently handle key tasks like managing food categories, food items, customer orders, delivery addresses, and shopping carts. The system is designed to be accessed only by **authorized administrators**, ensuring secure control over all operations.

Key Objectives:

1. **Item Category Management**:
    - Administrators can organize food items into categories (e.g., appetizers, main courses, desserts). They can also add, update, or remove categories as needed.
2. **Food Menu Management**:
    - Administrators can add, edit, or delete food items on the menu. Each item can include details like name, description, price, and availability.

3. **Customer Order Management**:
    - The system tracks customer orders, including food items, quantities, total cost, and payment status. Administrators can update order statuses, such as "Preparing" or "Delivered."

4. **Delivery Address Management**:
    - Administrators can view and manage customer delivery addresses to ensure orders are sent to the correct location. They can update addresses if needed.

5. **Shopping Cart Management**:
    - Administrators can oversee customer shopping carts, check what items are being added, and ensure everything is ready for checkout.

6. **Reporting**:
    - The system generates reports on orders, popular items, and revenue, helping admins make data-driven decisions.

7. **Security and Access Control**:
    - Only authorized users (administrators) can access sensitive information like customer orders, payments, and delivery details.

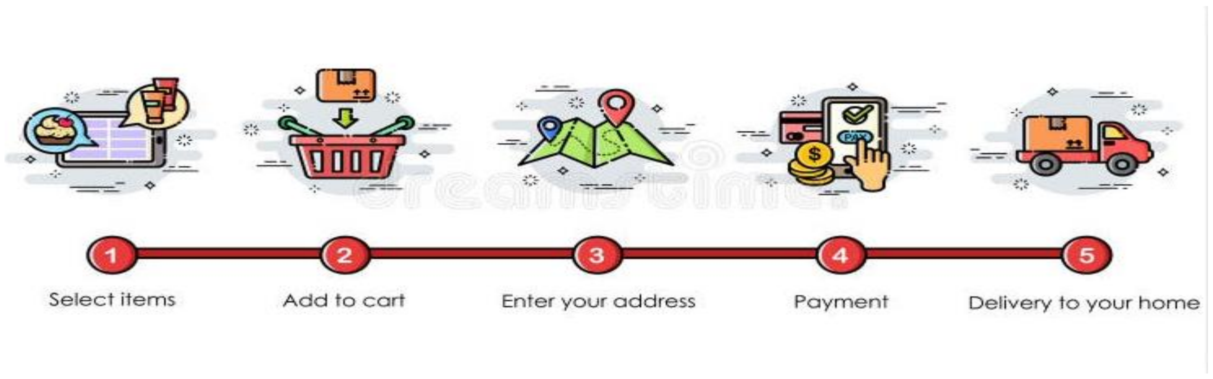


Image 2: Website process

1.4 Methodology

Methodology for Enhancing Web Development Efficiency using the MERN Stack:

● **Needs analysis:** Conduct a comprehensive requirements analysis to pinpoint the particular difficulties and pain points that developers have while using conventional web development techniques. This will assist in outlining the project's needs and goals.

● **Framework Design:** Design a thorough and well-rounded framework that makes use of the MERN stack to offer a combined frontend and backend development solution.This framework will assure excellent performance, scalability, and maintainability while addressing the highlighted pain areas.

● **Development:** Using the MERN stack, put the planned framework into use while making sure best practices and standards are followed. Create reusable React.js components to encourage the reuse of code and cut down on development time. To guarantee great performance, scalability, and reliability, test the designed solution.

● **Integration:** To ensure compatibility and lessen integration issues, integrate the generated solution with other pertinent technologies, databases, and frameworks.

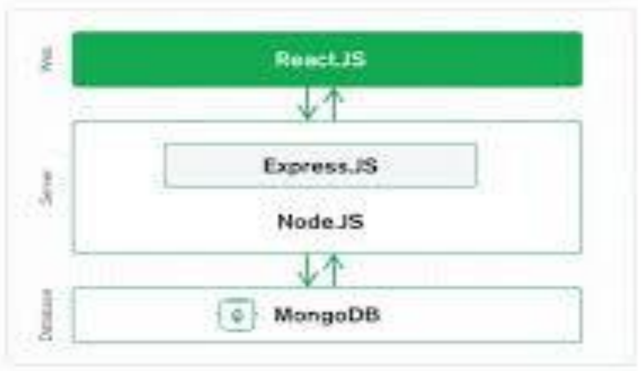


Image 3:Integration of MERN stack.

● **Deployment:** Put the designed solution into production, assuring scalability,dependability, and security. Load-test the solution to ensure that it can handle big volumes of data and concurrent requests.

● **Training:** Provide developers with training and assistance so that they can use the generated solution efficiently. This will shorten the learning curve while increasing developer productivity and cooperation.

● **Maintenance and updates:** Maintain and update the produced solution continuously to ensure that it stays current and relevant. This will give a long-term and forward-thinking approach to web development.

Overall, the goal of this technique is to provide a comprehensive and coherent solution that streamlines web development, increases efficiency, and assures excellent performance and maintainability of modern online applications utilising the MERN stack.

CHAPTER-2 LITERATURE SURVEY

2.1 LITERATURE SURVEY

The well-known MERN stack for web development consists of four technologies: MongoDB,Express.js, React.js, and Node.js. This stack is well-known for its versatility, efficacy, and usefulness. One of the primary advantages of utilising the MERN stack is that programmers may construct full-stack web apps using only JavaScript. In this review of the literature, we'll look at various facets of creating a web application with the MERN stack, such as setting up the environment for development, sending data from React to Node.js, utilising middleware like body-parser in Express.js, fetching data in React, handling post requests in Express.js, and connecting to a MongoDB database.

Developers may use a variety of online instructions to build up the development environment for a web application using the MERN stack. The procedures to link React and Node.js are described in one such tutorial, available at codedamn.com. The usage of package.json to manage dependencies and concurrently to execute the front end and back end simultaneously are highlighted in this guide.

There are numerous ways to accomplish the frequent need of sending data from React to Node.js in web development. One tutorial (tutsmake.com) explains how to use Axios to transmit data from React to a Node.js server via an HTTP POST request. It also describes

how to parse the request body using the Express.js middleware body-parser. Express.js's robust middleware capability enables developers to extend the server's capabilities. The usage of the body-parser middleware in Express.js is described in one article on Stack Overflow (stackoverflow.com). Before the handlers, this middleware is used to process incoming request bodies.

To update the front-end with fresh data, it is usual practice in web development to retrieve data from a server. One tutorial (developerway.com) describes how to utilise React's fetch API to get data from a server. Additionally, it explains how asynchronous functions are used and guarantees that the response data will be handled.

Express.js's post-request handling is a crucial component of server-side web development.How to handle post requests in Express.js and deliver a response to the client is covered in one article on Stack Overflow. This post also describes how to handle asynchronous activities by using the async/await syntax.

Using the MERN stack to construct a website requires connecting to a MongoDB database.

The procedures to set up an Express.js server with a MongoDB database are described in one tutorial at (mongodb.com). It also teaches how to build models for MongoDB collections using the Mongoose library.

There are some tips and tactics for developers in addition to the technical elements of web development utilising the MERN stack. How to duplicate a multidimensional array in JavaScript is described in one article on Stack Overflow. The usage of Axios to send HTTP requests in React is covered in another article on zetcode.com. On Telerik's website, there are lessons on how to build dynamic forms. Last but not least, there are some online resources you can utilise to learn more about React Router, the tool used to manage navigation in a React application.

CHAPTER -3 SYSTEM DEVELOPMENT

3.1 HARDWARE REQUIREMENTS

Local Development:

For building an online food delivery web application using the MERN stack (MongoDB, Express.js, React.js, and Node.js), you will need to consider both hardware and software requirements to ensure smooth performance, scalability, and security.

Processor: Modern multi-core processor (Intel i5/Ryzen 5 or above) for smooth development experience.
RAM: 8 GB of RAM (16 GB is recommended if running multiple containers or resource-heavy services).
Storage: SSD with at least 100 GB of available space for the operating system, code, and database.
Internet Connection: Stable and high-speed internet connection for fetching dependencies, API calls, and cloud deployments.

3.2 SOFTWARE REQUIREMENTS

Frontend (React.js):

- **React.js**: A JavaScript library for building user interfaces, especially single-page applications (SPA).
- **HTML5 / CSS3**: For basic web page structure and styling.
- **JavaScript (ES6+)**: Core language for frontend logic.
- **State Management**: You can use libraries like **Redux** or **Context API** to manage application state.
- **Routing**: Use **React Router** for handling routing in your application.
- **UI Libraries** (Optional): You can use libraries like **Material UI**, **Ant Design**, or **Bootstrap** for pre-built components to make development faster.
- **Authentication**: Implement JWT (JSON Web Tokens) for user authentication, ensuring secure login/logout flows.
- **API Calls**: Use **Axios** or **Fetch API** for interacting with the backend API.

Backend (Node.js + Express.js):

- **Node.js**: A JavaScript runtime for building the server-side of your application.
- **Express.js**: A lightweight framework built on top of Node.js to handle HTTP requests, middleware, and routing.
- **MongoDB**: NoSQL database for storing data like users, food items, orders, and delivery status.
- **Authentication**: Implement user authentication using **Passport.js** or **JWT** (JSON Web Tokens).

- **API Development**: RESTful APIs to interact with the frontend for CRUD operations (e.g., creating orders, updating order status, etc.).
- **Middleware**: For error handling, logging, and security (e.g., **helmet.js**, **cors**).
- **Payment Integration**: Integrate third-party services like **Stripe** or **Razorpay** for payment processing.

Development Tools:

- **Node.js**: Install the latest stable version of Node.js to run your backend server.
- **npm** or **Yarn**: Package managers for managing libraries and dependencies.
- **MongoDB**: Local development can be done with a locally installed MongoDB instance or by using a managed service like **MongoDB Atlas**.
- **Version Control**: Use **Git** for version control and **GitHub/GitLab** for repository hosting.
- **Code Editor**: Use editors like **VS Code**, **Sublime Text**, or **Atom** for writing code.

3.3 SYSTEM DESIGN

Frontend Development (React.js):

1. **UI Components**: Break down the UI into reusable components such as Header, Footer, FoodItemCard, Cart, OrderStatus, etc.
2. **State Management**: Use **React Context** or **Redux** to manage global state (e.g., cart items, user authentication).
3. **API Integration**: Use **Axios** or **Fetch API** to interact with the backend for retrieving food items, placing orders, handling payments, etc.

4. **Routing**: Implement navigation with **React Router** for pages like Home, Menu, Cart, Order Tracking, Admin Panel.
5. **Responsive Design**: Use CSS Flexbox or Grid, along with media queries to ensure the application is mobile-friendly.

Backend Development (Node.js + Express.js):

1. **API Routes**: Develop RESTful API routes for handling operations such as:
   - User sign up, login, and profile management.
   - Fetching food items, adding/removing items (for admin).
   - Placing orders, viewing order status.
   - Handling payments via Stripe API.
2. **Authentication**:
   - Use **JWT (JSON Web Tokens)** for managing authentication and protecting routes (e.g., secure order placement, admin actions).
   - Use **bcrypt** for hashing passwords and securely storing them.
3. **Order Processing**:
   - Implement order management logic (e.g., transitioning from "Pending" to "Shipped").
   - Admin route to view and manage orders.
   - Implement email notifications or SMS notifications for order status updates.

Database Integration (MongoDB):

1. **MongoDB Atlas**: Use a managed MongoDB service for easy scaling and automatic backups.
2. **Mongoose ODM**: Use **Mongoose** to define schema models for food items, users, and orders.

3. **CRUD Operations**: Implement Create, Read, Update, and Delete operations for managing food items, user data, and orders.

Payment Gateway Integration (Stripe):

- **Stripe SDK**: Integrate **Stripe** for handling secure payments on the frontend and backend.
- **Stripe Webhooks**: Set up webhooks to listen for payment events (e.g., successful payment, payment failure).
- **Security**: Ensure secure handling of sensitive payment information using **Stripe Elements** or **Stripe Checkout**.

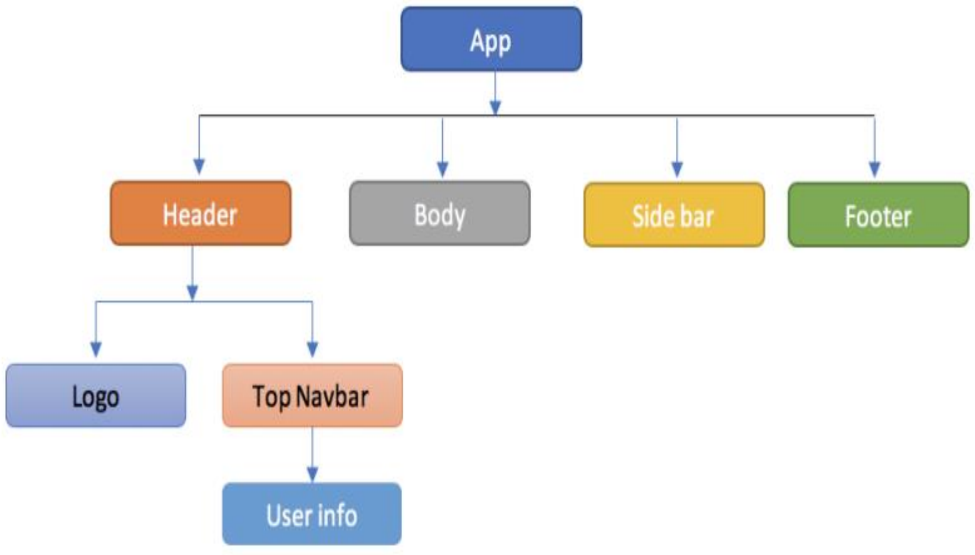


Image 4: App Outline

3.4 MODEL DEVELOPMENT

1. User Model:

```
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  address: { type: String, required: true },
  phone: { type: String },
  orderHistory: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Order' }],
  role: { type: String, enum: ['user', 'admin'], default: 'user' }
});

const User = mongoose.model('User', userSchema);
```

2. Food Item Model:

```
const foodItemSchema = new mongoose.Schema({
  name: { type: String, required: true },
  category: { type: String, required: true },
  description: { type: String },
  price: { type: Number, required: true },
  imageUrl: { type: String },
  availableStock: { type: Number, default: 0 },
});

const FoodItem = mongoose.model('FoodItem', foodItemSchema);
```

3. Order Model:

```
const orderSchema = new mongoose.Schema({
  user: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
  orderItems: [
    {
      foodItem: { type: mongoose.Schema.Types.ObjectId, ref: 'FoodItem' },
      quantity: { type: Number, required: true }
    }
  ],
  totalAmount: { type: Number, required: true },
  status: { type: String, enum: ['Pending', 'Preparing', 'Shipped', 'Delivered'],
default: 'Pending' },
  paymentStatus: { type: String, enum: ['Paid', 'Pending', 'Failed'], default:
'Pending' },
  paymentMethod: { type: String, enum: ['Stripe'], default: 'Stripe' },
  paymentDate: { type: Date },
  orderDate: { type: Date, default: Date.now }
});

const Order = mongoose.model('Order', orderSchema);
```

This gives you a solid foundation for your online food delivery web application in the MERN stack, covering analysis, design, development, algorithms, and model development. You can refine and iterate on each section as you continue to develop and scale the application.

3.5 DATA FLOW

A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel. A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later elaborated. DFDs can also be used for the visualization of data processing.

External Entity

An external entity can represent a human, system or subsystem. It is where certain data comes from or goes to. It is external to the system we study, in terms of the business process. For this reason, people used to draw external entities on the edge of a diagram.

Process

A process is a business activity or function where the manipulation and transformation of data takes place. A process can be decomposed to finer level of details, for representing how data is being processed within the process.

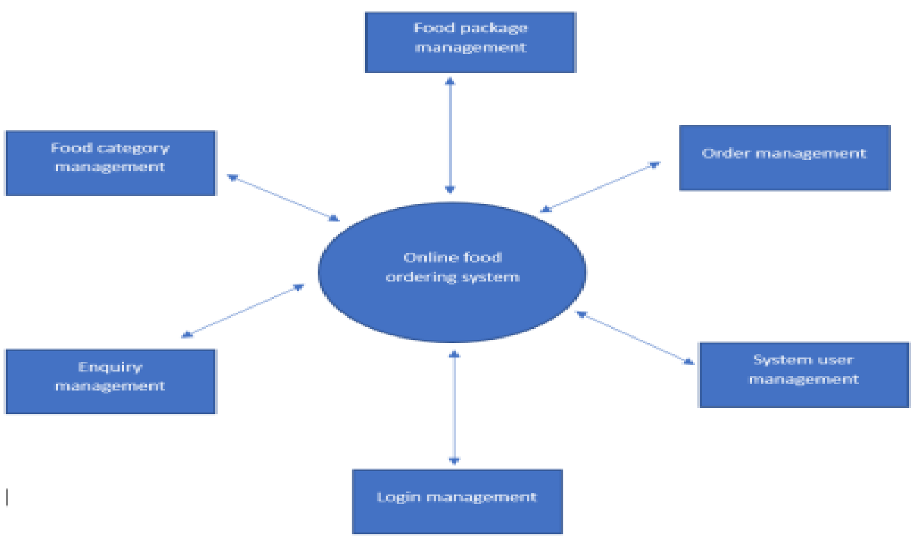
Data Store

A data store represents the storage of persistent data required and/or produced by theprocess. Here are some examples of data stores: membership forms, database table, etc.

Data Flow

A data flow represents the flow of information, with its direction represented by an arrow head that shows at the end(s) of flow connector.

DATA FLOW DIAGRAM



3.6 SOFTWARE TESTING

Testing the software is the process of validating and verifying of a softwareprogram. The errors are to be identified in order to fix those errors. Thus the main objective of software testing is to maintain and deliver a quality product to the client. Every software is expected to meet certain needs. So when software is developed it is required to check whether it fulfills those needs. Abanking software is entirely different from a software required in a shop. The needs and requirements of both those software are different. Hence it is important to check its potential. The main goal of software testing is to know the errors of the software before the user finds them. Software testing is the operating of software under controlled conditions to check whether the software works well and to rectify the errors, and also to make sure that we are delivering the correct software what the user intends.

OBJECTIVE OF TESTING:

Software testing helps to make sure that it meets all the requirement it was supposed to meet. It will bring out all the errors, if any, while using the software. Software testing helps to understand that the software that is being tested is a complete success Software testing helps to give a quality certification that the software can be used by the client immediately. It ensure quality of the product.

TYPES OF TESTING

Is the menu bar displayed in the appropriate contested some system related features included either in menus or tools? Do pull -Down menu operation and

Tool-bars work properly? Are all menu function and pull down sub function properly listed ?; Is it possible to invoke cach menu function using a logical assumptions that if all parts of the system are correct, the goal will be successfully achieved.? In adequate testing or non-testing will leads to errors that may appear few months later,

This create two problem

1. Time delay between the cause and appearance of the problem.

2. The effect of the system errors on files and records within the system The purpose of the system testing is to consider all the likely variations to which it will be suggested and push the systems to limits.

The testing process focuses on the logical intervals of the software ensuring that all statements have been tested and on functional interval is conducting tests to uncover errors and ensure that defined input will produce actual results that agree with the required results. Program level testing, modules level testing integrated and carried out. There are two major type of testing they are

1) White Box Testing.

2) Black Box Testing.

White Box Testing

White box sometimes called "Glass box testing" is a test case design uses the control structure of the procedural design to drive test case. Using white box testing methods, the following tests were made on the system

a) All independent paths within a module have been exercised once. In our system, ensuring that case was selected and executed checked all case structures. The bugs that were prevailing in some part of the code where fixed

b) All logical decisions were checked for the truth and falsity of the values.

Black Box Testing

Black box testing focuses on the functional requirements of the software. This is black box testing enables the software engineering to derive a set of input conditions that will fully exercise all functional requirements for a program. Black box testing is not an alternative to white box testing rather it is complementary approach that is likely to uncover a different class of errors that white box methods like.

1) Interface errors

2) Performance in data structure

3) Performance errors

4) Initializing and termination errors

LEVELS OF TESTING:

There are four levels of software testing:

• Unit Testing

• Integration Testing

• System Testing

• Acceptance Testing

1.Unit Testing

Unit Testing is a level of the software testing process where individual units/components of a software system are tested. The purpose is to validate that each unit of the software performs a as designed.

2. Integration Testing

Integration Testing is a level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

3. System Testing

System Testing is a level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. Acceptance Testing is a level of the software testing process where a system is tested for.

4. Acceptance Testing

Acceptability is the purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.

CHAPTER -4 PROPOSED SYSTEM

4.1 ALGORITHM

Order Placement Algorithm:

Step 1: User selects food items from the menu and adds them to the cart.

Step 2: User proceeds to checkout where the total price is calculated.

Step 3: User logs in (or creates an account) and enters payment details.

Step 4: Payment is processed through Stripe API.

Step 5: If payment is successful, create a new order in the database and update the order status to Pending.

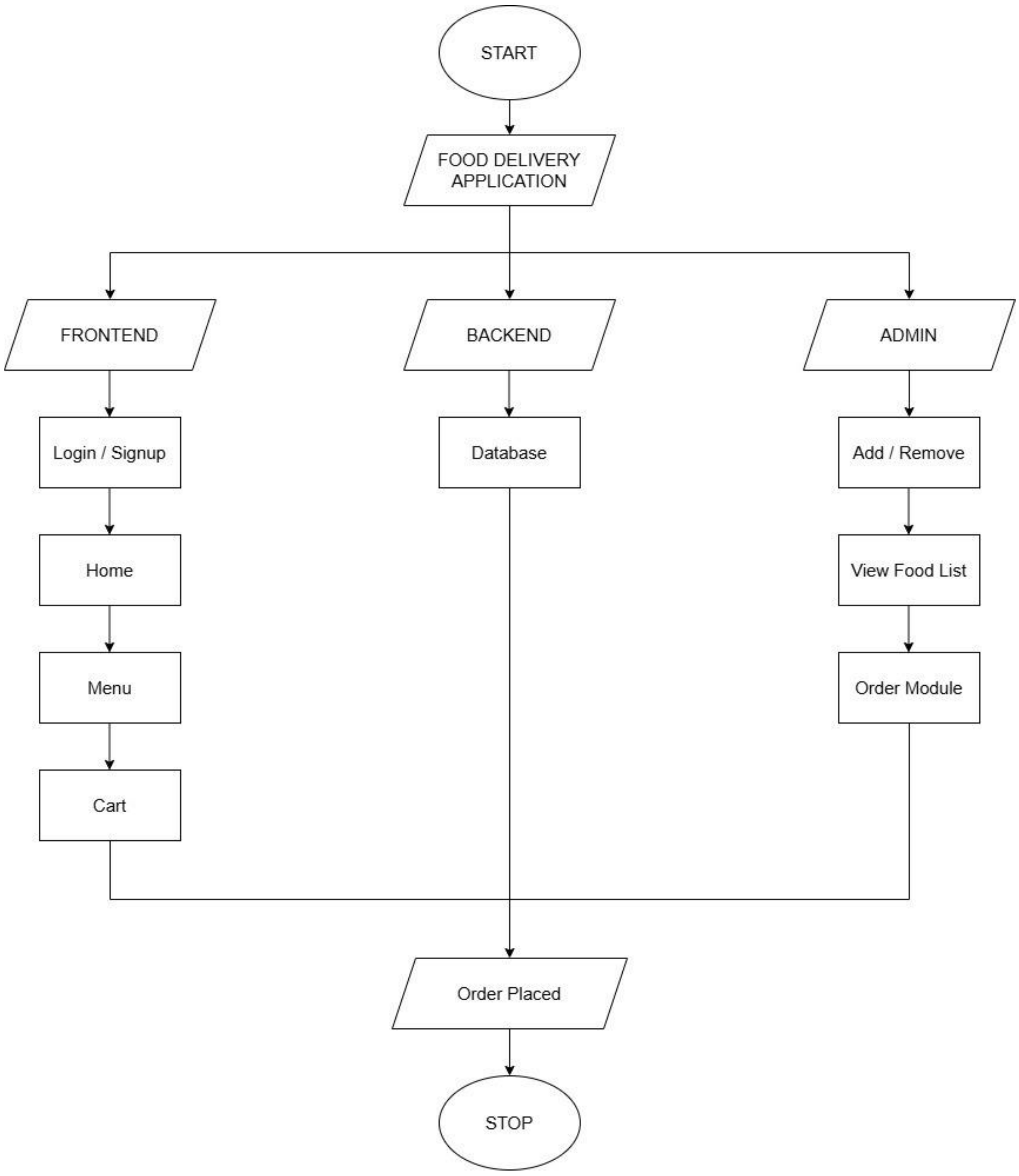
Step 6: Notify the admin that a new order has been placed.

Step7: Admin processes the order (e.g., changes status to "Preparing").

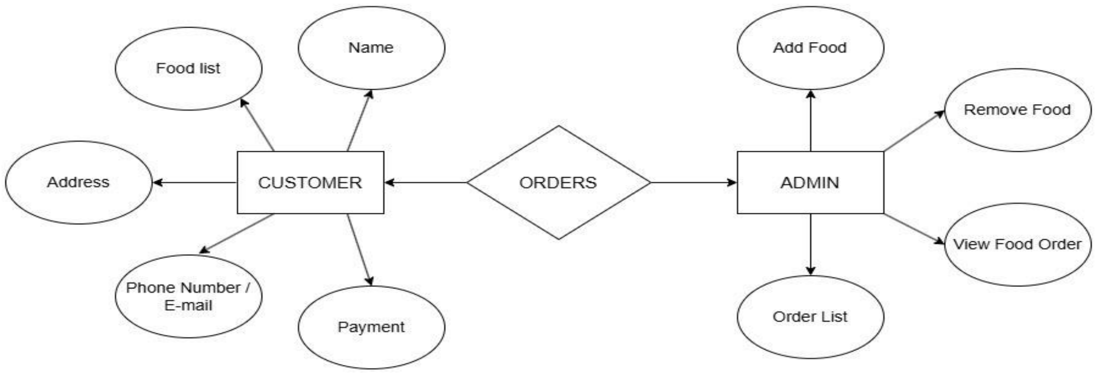
Step 8: User can track the order status.

Step 9: Admin updates the status (e.g., "Shipped" or "Delivered") and notifies the user.

4.2 FLOW OF PROJECT



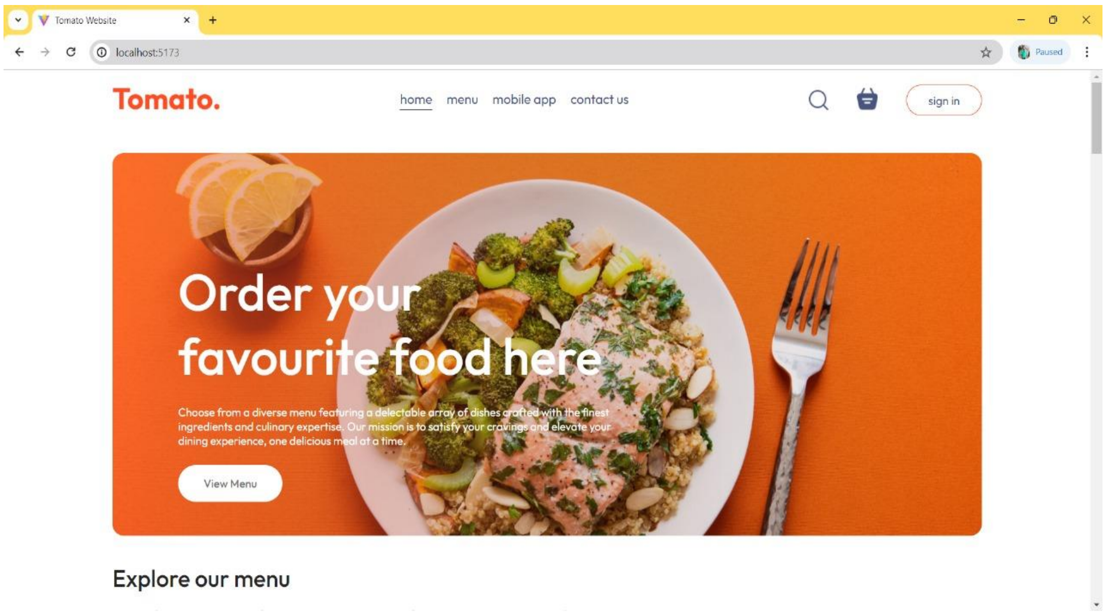
4.3 ER-Diagram



<u>CHAPTER -5 RESULT AND DISCUSSION</u>

USER INTERFACE

HOME PAGE

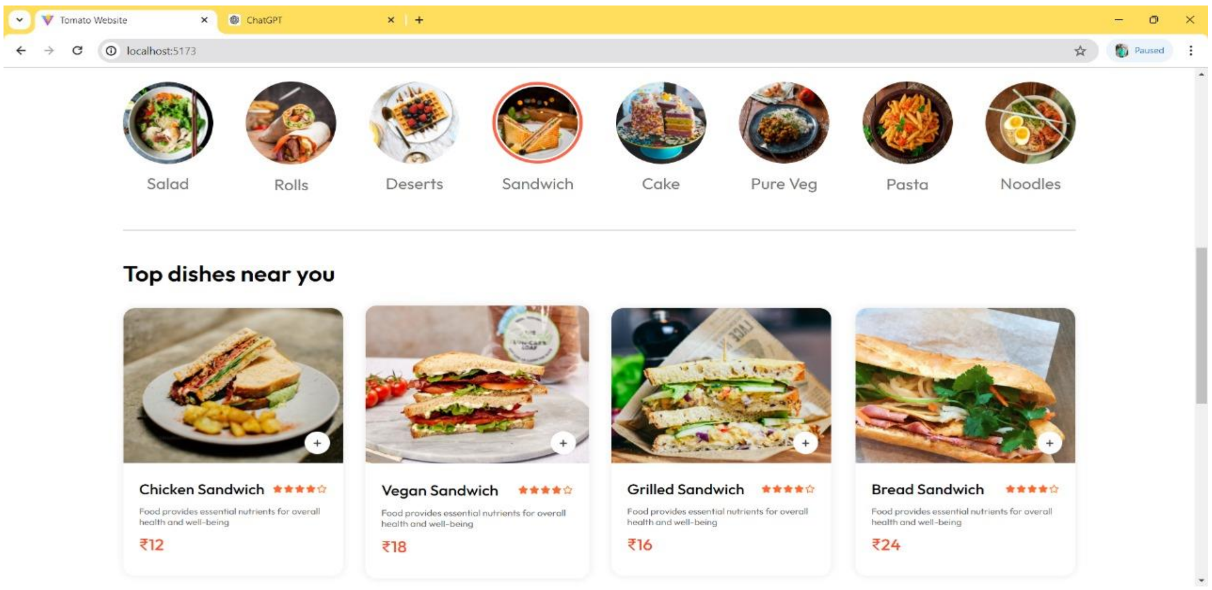


Order your favourite food here

Choose from a diverse menu featuring a delectable array of dishes crafted with the finest ingredients and culinary expertise. Our mission is to satisfy your cravings and elevate your dining experience, one delicious meal at a time.

Explore our menu

MENU PAGE



Top dishes near you

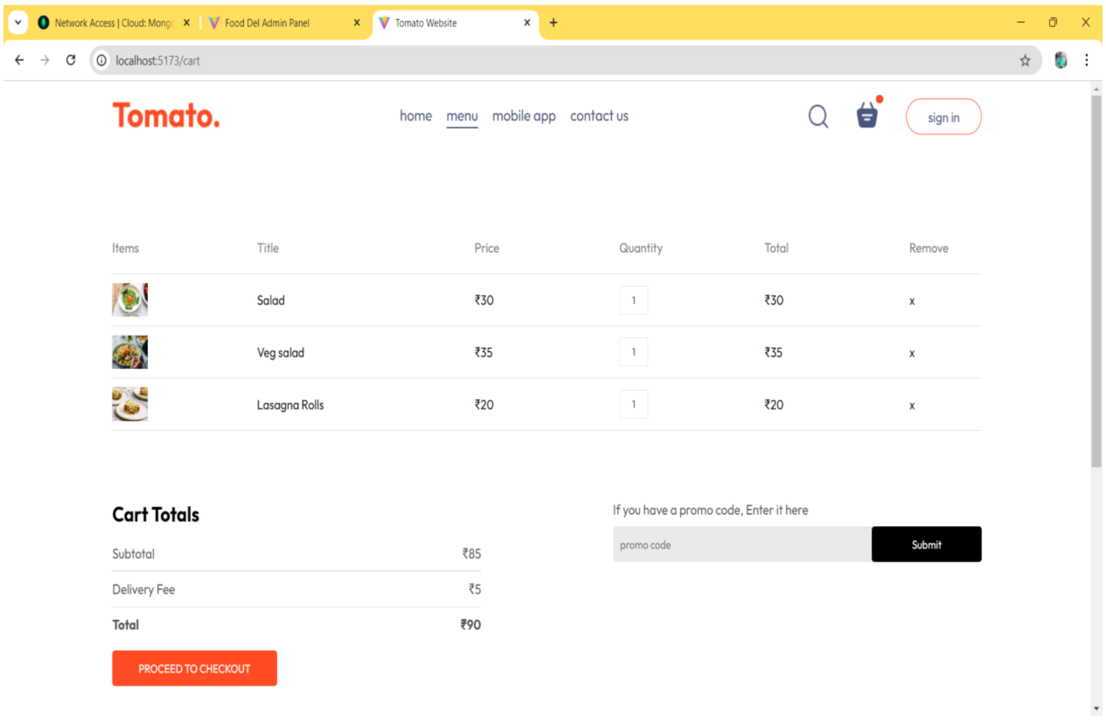
Chicken Sandwich ★★★★☆
Food provides essential nutrients for overall health and well-being
₹12

Vegan Sandwich ★★★★☆
Food provides essential nutrients for overall health and well-being
₹18

Grilled Sandwich ★★★★☆
Food provides essential nutrients for overall health and well-being
₹16

Bread Sandwich ★★★★☆
Food provides essential nutrients for overall health and well-being
₹24

CART PAGE

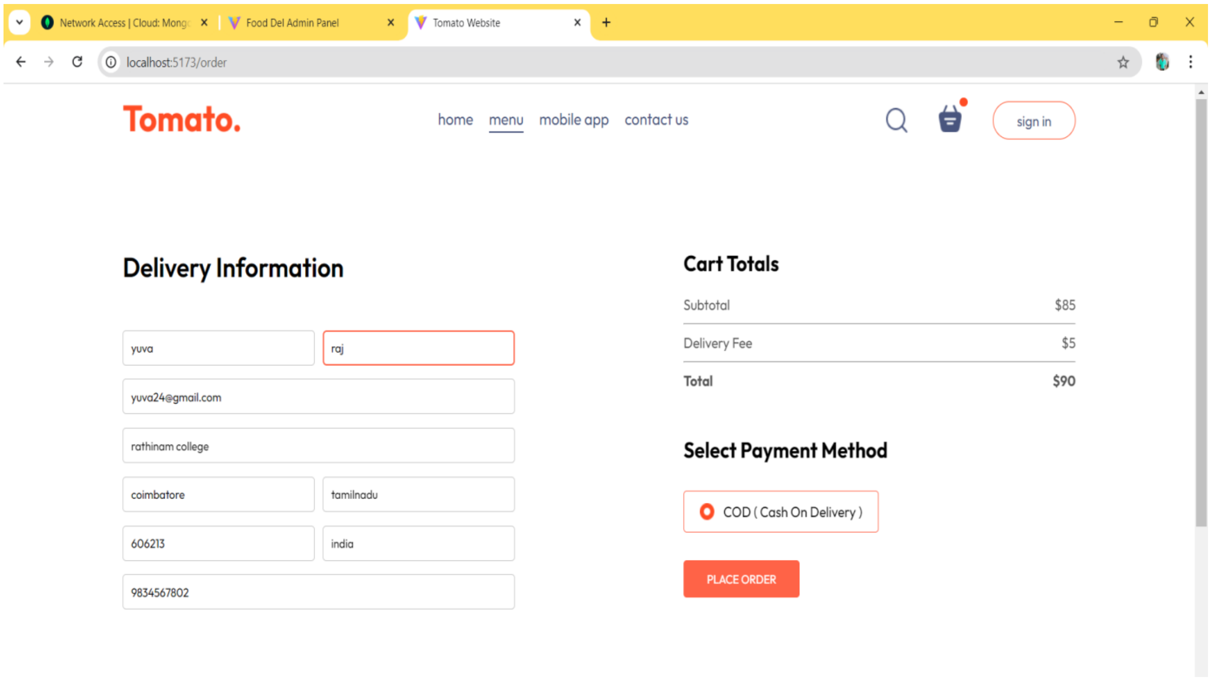


Items	Title	Price	Quantity	Total	Remove
	Salad	₹30	1	₹30	x
	Veg salad	₹35	1	₹35	x
	Lasagna Rolls	₹20	1	₹20	x

Cart Totals

Subtotal	₹85
Delivery Fee	₹5
Total	**₹90**

PLACEORDER PAGE



Delivery Information

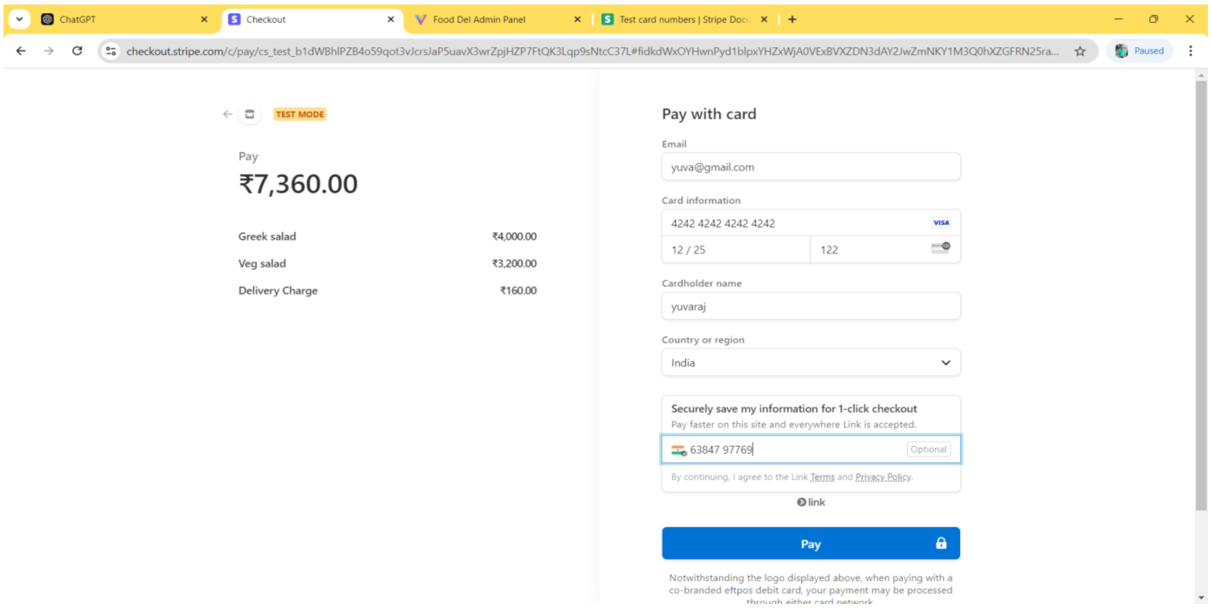
yuva
raj
yuva24@gmail.com
rathinam college
coimbatore
tamilnadu
606213
india
9834567802

Cart Totals

Subtotal	$85
Delivery Fee	$5
Total	**$90**

Select Payment Method

PAYMENT


TEST MODE

Pay

₹7,360.00

Greek salad	₹4,000.00
Veg salad	₹3,200.00
Delivery Charge	₹160.00

Pay with card

Email: yuva@gmail.com

Card information: 4242 4242 4242 4242, 12 / 25, 122

Cardholder name: yuvaraj

Country or region: India

Securely save my information for 1-click checkout

Pay faster on this site and everywhere Link is accepted.

63847 97769 Optional

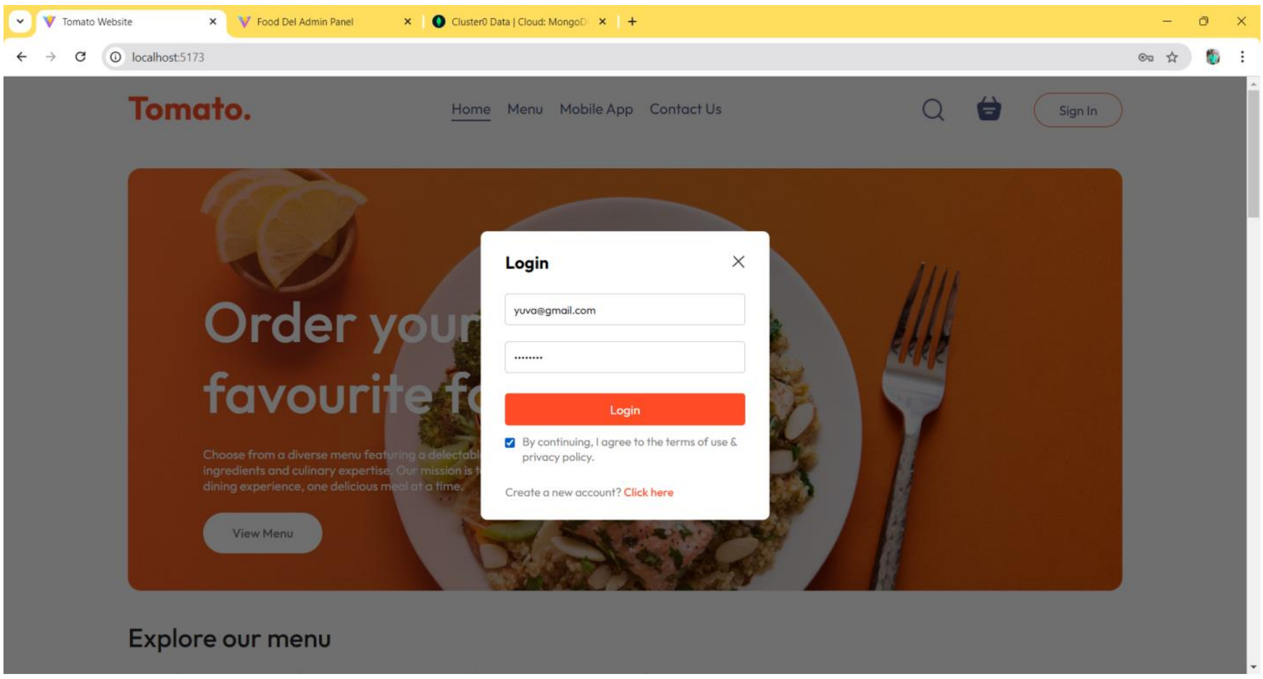
By continuing, I agree to the Link Terms and Privacy Policy.

link

Pay

Notwithstanding the logo displayed above, when paying with a co-branded eftpos debit card, your payment may be processed through either card network.

LOGIN PAGE



Login

yuva@gmail.com

••••••••

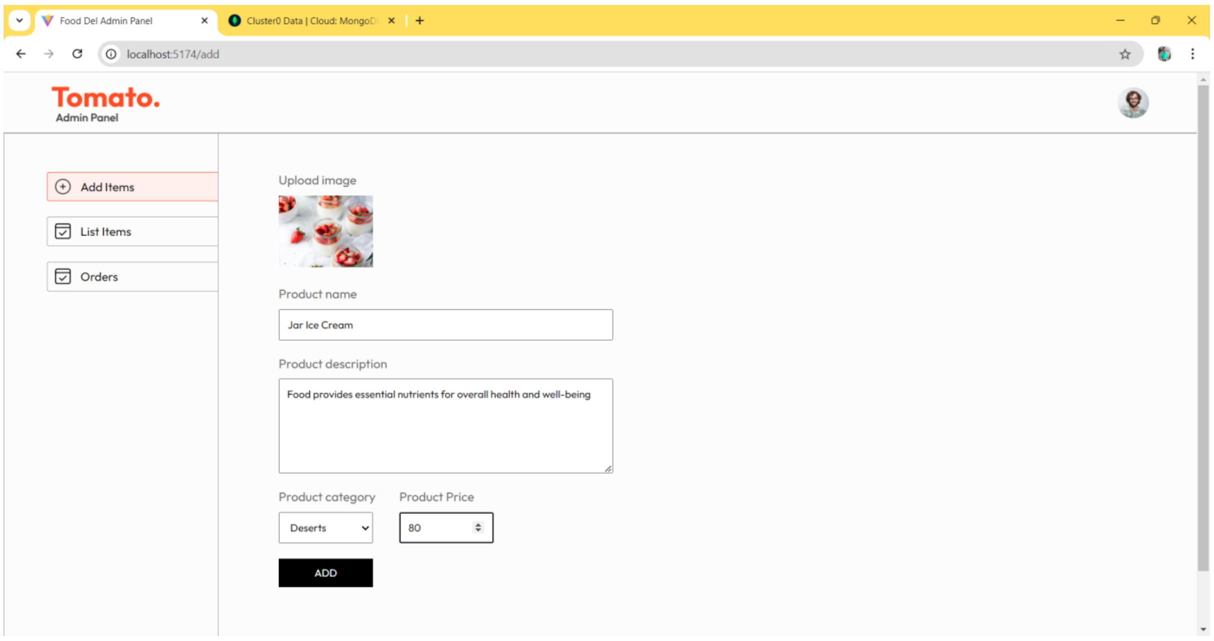
Login

By continuing, I agree to the terms of use & privacy policy.

Create a new account? Click here

ADMIN PAGE

ADD ITEMS



Upload image

Product name

Jar Ice Cream

Product description

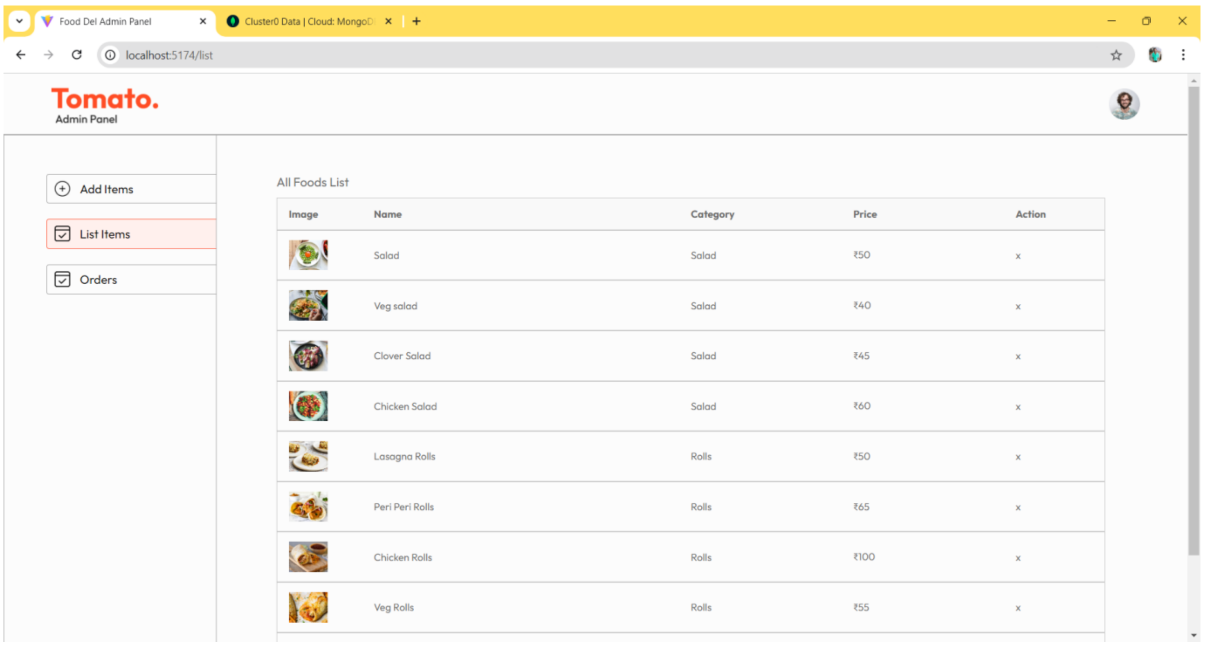
Food provides essential nutrients for overall health and well-being

Product category: Deserts

Product Price: 80

ADD

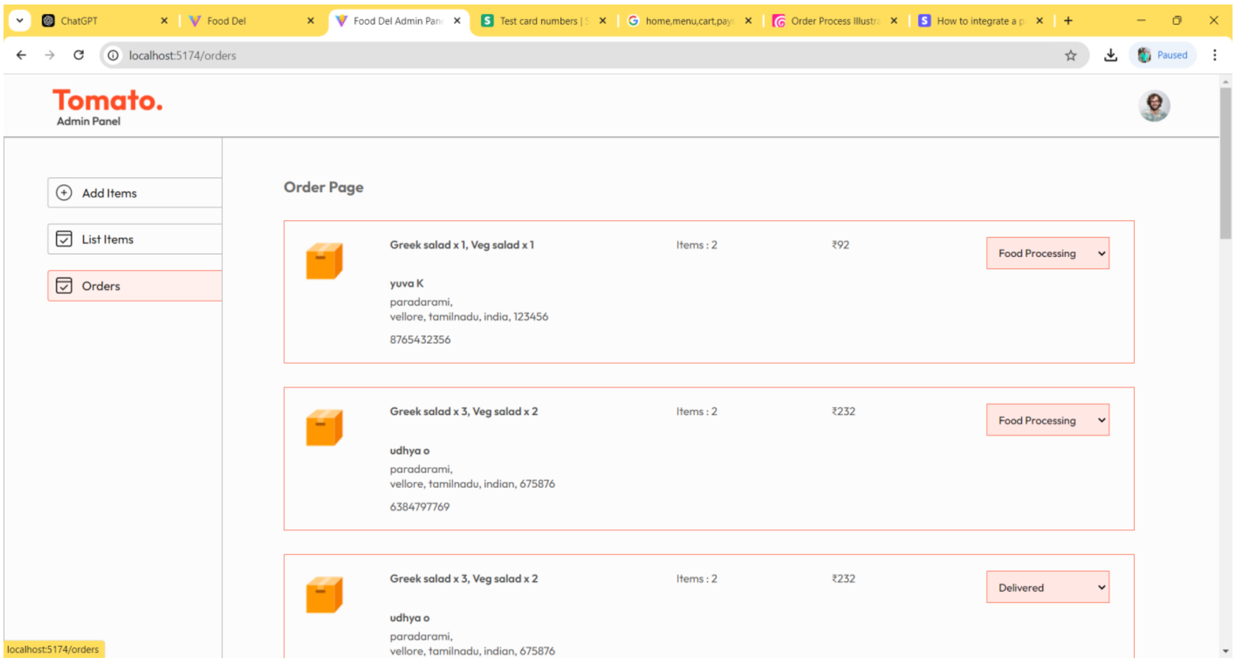
LIST ITEMS



All Foods List

Image	Name	Category	Price	Action
	Salad	Salad	₹50	x
	Veg salad	Salad	₹40	x
	Clover Salad	Salad	₹45	x
	Chicken Salad	Salad	₹60	x
	Lasagna Rolls	Rolls	₹50	x
	Peri Peri Rolls	Rolls	₹65	x
	Chicken Rolls	Rolls	₹100	x
	Veg Rolls	Rolls	₹55	x

ORDERS PAGE



Order Page

Greek salad x 1, Veg salad x 1 Items : 2 ₹92 Food Processing

yuva K
paradarami,
vellore, tamilnadu, india, 123456
8765432356

Greek salad x 3, Veg salad x 2 Items : 2 ₹232 Food Processing

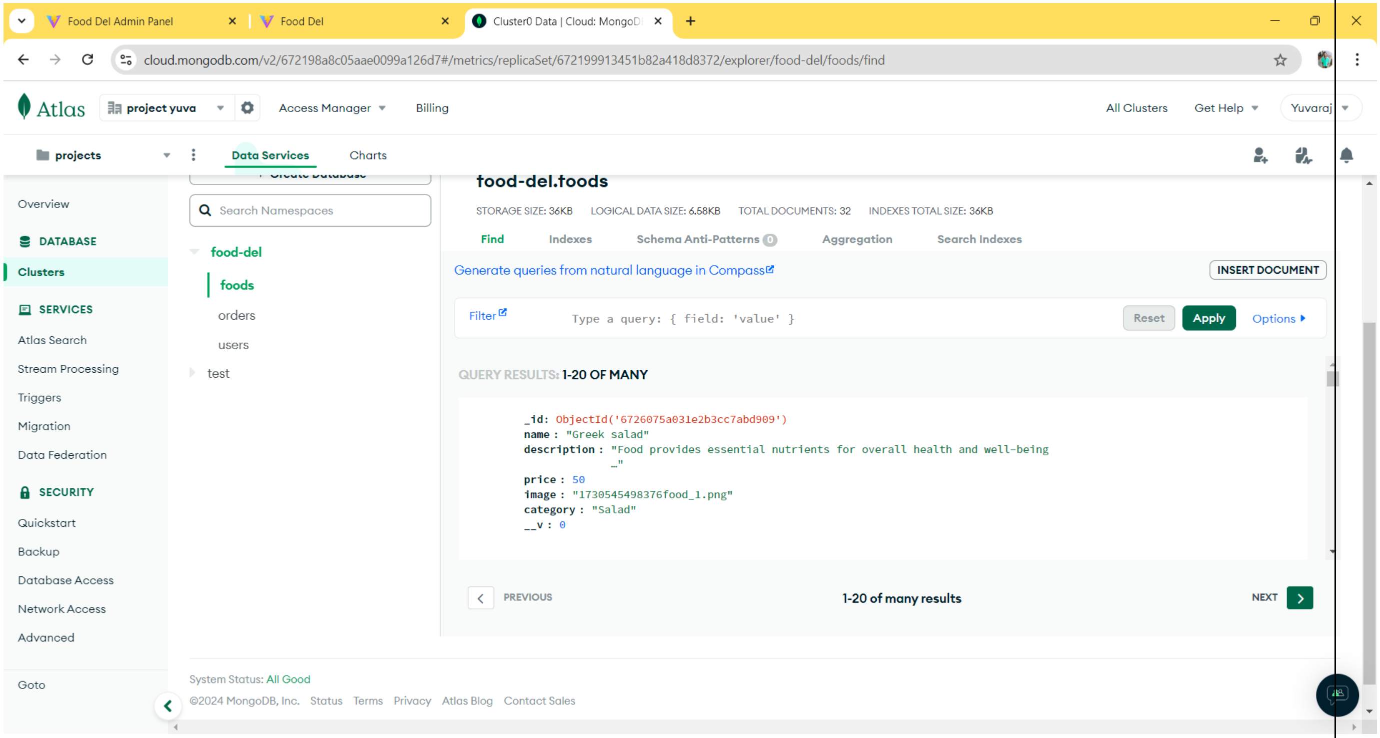
udhya o
paradarami,
vellore, tamilnadu, indian, 675876
6384797769

Greek salad x 3, Veg salad x 2 Items : 2 ₹232 Delivered

udhya o
paradarami,
vellore, tamilnadu, indian, 675876

DATABASE

FOOD ITEMS



food-del.foods

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 6.58KB TOTAL DOCUMENTS: 32 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

Generate queries from natural language in Compass

INSERT DOCUMENT

Filter Type a query: { field: 'value' } Reset Apply Options

QUERY RESULTS: 1-20 OF MANY

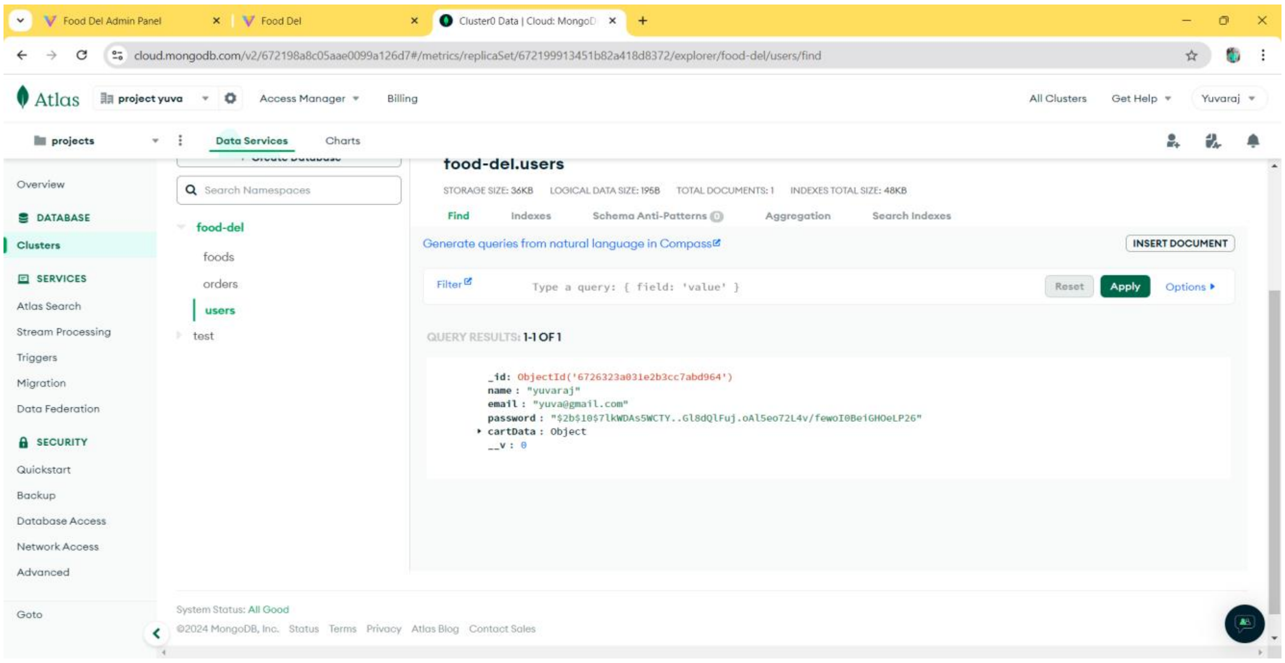
```
_id: ObjectId('6726075a031e2b3cc7abd909')
name : "Greek salad"
description : "Food provides essential nutrients for overall health and well-being
  ."
price : 50
image : "1730545498376food_1.png"
category : "Salad"
__v : 0
```

PREVIOUS 1-20 of many results NEXT

System Status: All Good

©2024 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

USERS



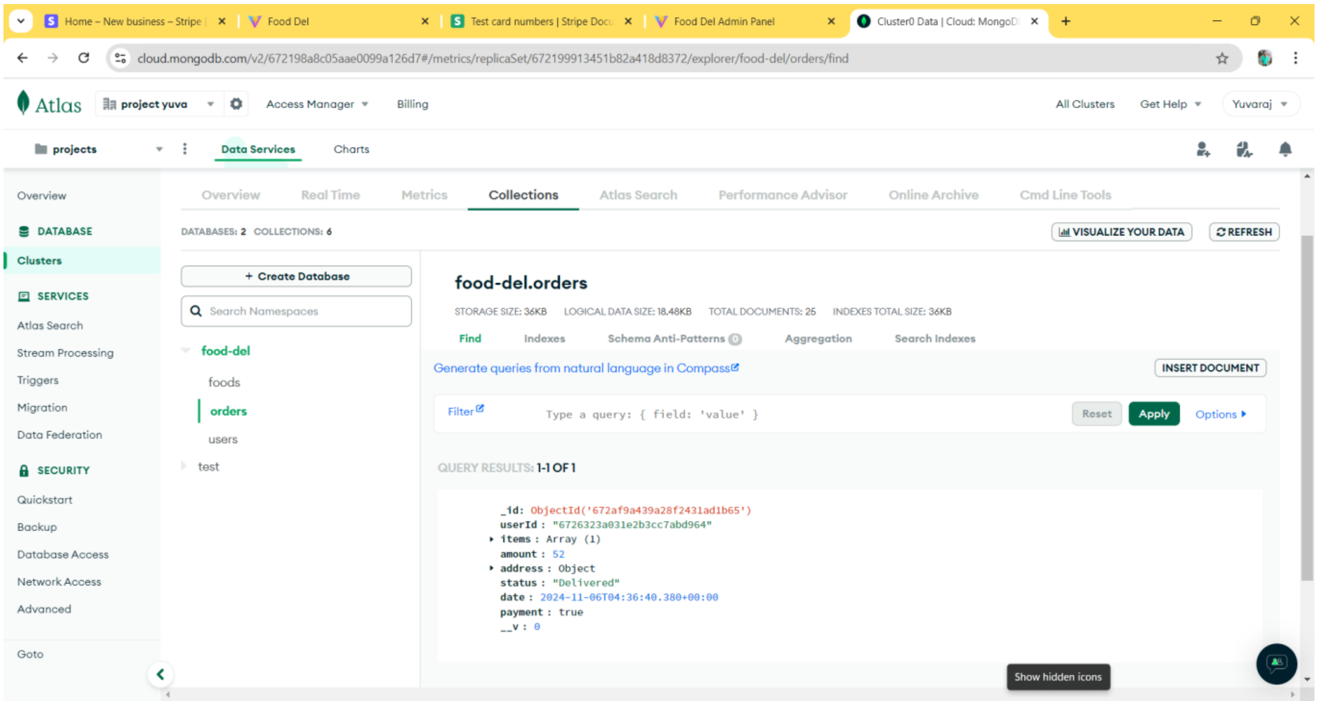
food-del.users

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 195B TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 48KB

QUERY RESULTS: 1-1 OF 1

```
_id: ObjectId('6726323a031e2b3cc7abd964')
name : "yuvaraj"
email : "yuva@gmail.com"
password : "$2b$10$7lkWDAs5WCTY..Gl8dQlFuj.oAl5eo72L4v/fewoI0BeiGHOeLP26"
cartData : Object
__v : 0
```

ORDERS



food-del.orders

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 18.48KB TOTAL DOCUMENTS: 25 INDEXES TOTAL SIZE: 36KB

QUERY RESULTS: 1-1 OF 1

```
_id: ObjectId('672af9a439a28f2431ad1b65')
userId : "6726323a031e2b3cc7abd964"
items : Array (1)
amount : 52
address : Object
status : "Delivered"
date : 2024-11-06T04:36:40.380+00:00
payment : true
__v : 0
```

DISCUSSION :-

UI/UX Design:

The design of your web application should focus on simplicity, ease of navigation, and responsiveness. The key pages to be designed include:

1. **Sign Up / Login Pages**:
    - User-friendly forms with email and password fields.
    - Input validation to prevent incorrect inputs.
    - Option for users to reset passwords.
    - Include "Remember me" option and social login integrations (e.g., Google or Facebook).
2. **Home Page**:
    - A hero section with a catchy banner or promotional offers.
    - Quick links to the menu, cart, and order tracking.
    - Search bar to search for specific food items.
3. **Menu Page**:
    - Display food items with images, descriptions, prices, and ratings.
    - Option to filter by categories or cuisines.
    - "Add to Cart" button for each food item.
4. **Cart Page**:
    - Display added items with the ability to update quantity or remove items.
    - Show total price and estimated delivery cost.
    - Button to proceed to checkout and make payment.

5. **Order Tracking Page**:

    - Display order details (order ID, food items, total cost).
    - Real-time order status (e.g., "Preparing," "Out for Delivery," etc.).
    - Option to cancel or modify orders if the status allows.

6. **Admin Panel**:

    - Dashboard displaying key metrics (e.g., number of orders, active users).
    - A section to add, edit, or remove food items from the menu.
    - An order management screen to process orders (e.g., Mark as Delivered, Prepare, etc.).
    - Ability to view past orders and track their status.

Database Schema Design (MongoDB):

The following collections could be included in your MongoDB database:

1. **Users Collection**:

    ➢ Fields: user_id, name, email, password_hash, address, phone, order_history[], role (user/admin).

2. **Food Items Collection**:

    ➢ Fields: food_id, name, category, description, price, image_url, available_stock.

3. **Orders Collection**:

- Fields: order_id, user_id, order_date, total_amount, status (Pending, Preparing, Shipped, Delivered), order_items[], payment_status.

4. **Payments Collection**:

   - Fields: payment_id, order_id, user_id, payment_method (Stripe), payment_status, payment_date.

CHAPTER -6 CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION

The online food delivery web application provides a seamless platform for users to browse menus, place orders, and make payments securely. Using the MERN stack (MongoDB, Express.js, React.js, and Node.js), the application offers a strong backend for managing user data, food items, and orders, while the frontend ensures a smooth and user-friendly experience.

With features like user registration, order tracking, and an admin panel for managing the menu and orders, the application serves as a complete solution for online food ordering and delivery, ensuring convenience and efficiency for users and administrators.

FUTURE ENHANCEMENT

Real-Time Order Tracking:

The system allows customers to track the status of their orders in real time. From order confirmation to food preparation and delivery, users can monitor the entire process, ensuring transparency and reducing anxiety about delivery times.

More Payment Options:

The platform supports multiple payment methods, including credit/debit cards, digital wallets (like PayPal, Google Pay, or Apple Pay), and cash on delivery. This flexibility provides users with a variety of payment choices, making the ordering process more convenient and accessible.

Ratings and Reviews:

Customers can rate dishes and leave reviews based on their dining experience. This feature helps other users make informed decisions and provides valuable feedback to restaurants, allowing them to improve their menu offerings and service quality.

Admin Analytics:

The **admin dashboard** provides detailed analytics and reports, such as sales trends, popular dishes, customer behavior, and order volume. This data helps restaurant owners make informed decisions, manage inventory more efficiently, and create targeted promotions.

Voice Search:

With the integration of **voice search**, customers can place orders hands-free by simply speaking into their devices. This feature makes the ordering process even more convenient, especially for users on the go or with accessibility needs.

Customer Support:

An integrated **customer support** feature, such as live chat or a support hotline, ensures that customers can get help quickly if they have issues with their orders, payments, or delivery. This boosts customer satisfaction and builds trust in the platform.

BIBLIOGRAPHY

- https://stackoverflow.com/questions/33092682/online-users-on-asp-net-application
- https://www.w3schools.com/html/html5_webstorage.asp
- https://www.jqueryscript.net/demo/noty-Cool-Jquery-Notification-Plugin/demo/
- https://www.itsolutionstuff.com/post/jquery-notification-popup-box-example-using- toastr-jsplugin-with-demoexample.html
- https://developer.squareup.com/docs/orders-api/how-it-works#orders-objects-and- datatypes
- https://www.jqueryscript.net/tags.php?/Notification/
- http://bootstrap-notify.remabledesigns.com/
- https://www.codeproject.com/Questions/665016/how-to-view-online-users-in-asp-net
- https://api.jquery.com/jquery.ajax/

APPENDIX

FRONTEND CODE

APP.jsx

```
import React, { useState } from 'react';

import Navbar from './components/Navbar/Navbar';

import { Route, Routes } from 'react-router-dom';

import Home from './pages/Home/Home';

import Cart from './pages/Cart/Cart';

import PlaceOrder from './pages/PlaceOrder/PlaceOrder';

import Footer from './components/Footer/Footer';

import LoginPopup from './components/LoginPopup/LoginPopup';

import Verify from './pages/Verify/Verify';

import MyOrder from './pages/MyOrder/MyOrder';

const App = () => {

  const [showLogin, setShowLogin] = useState(false);

return (

    <>
```

```
      {showLogin ? <LoginPopup setShowLogin={setShowLogin} /> : null}
      <div className='app'>
        <Navbar setShowLogin={setShowLogin} />
        <Routes>
          <Route path='/' element={<Home />} />
          <Route path='/cart' element={<Cart />} />
          <Route path='/order' element={<PlaceOrder />} />
          <Route path='/verify' element={<Verify />} /> {/* Corrected this line */}
          <Route path='/myorders' element={<MyOrder />} />
        </Routes>
      </div>
      <Footer />
    </>
  );
};
export default App;
```

ADMIN PANEL:-

App.jsx

```
import React from 'react'

import Navbar from './components/Navbar/Navbar'

import Sidebar from './components/Sidebar/Sidebar'

import { Route, Routes } from 'react-router-dom'

import Add from './pages/Add/Add'

import List from './pages/List/List'

import Orders from './pages/Orders/Orders'

import { ToastContainer } from 'react-toastify';

import 'react-toastify/dist/ReactToastify.css';

const App = () => {

  const url="http://localhost:4000";

  return (

    <div className='app'>

      <ToastContainer />

      <Navbar />
      <hr />

      <div className="app-content">

        <Sidebar />

        <Routes>

          <Route path="/add" element={<Add url={url} />} />

          <Route path="/list" element={<List url={url} />} />

          <Route path="/orders" element={<Orders url={url} />} />

        </Routes>

      </div></div>

  )}export default App
```

BACKEND CODE

Server.js

```
import express  from "express"

import cors from 'cors'

import { connectDB } from "./config/db.js"

import userRouter from "./routes/userRoute.js"

import foodRouter from "./routes/foodRoute.js"
import 'dotenv/config'

import cartRouter from "./routes/cartRoute.js"

import orderRouter from "./routes/orderRoute.js"



// app config

const app = express()

const port = process.env.PORT || 4000;

// middlewares

app.use(express.json())

app.use(cors())

// db connection

connectDB()

// api endpoints

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

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

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

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

app.use("/api/order",orderRouter)
app.get("/", (req, res) => {

    res.send("API Working")

  });

app.listen(port, () =>{

  console.log(`Server started on http://localhost:${port}`)

})
```