

# **FULL STACK DEVELOPMENT WITH MERN**

**PROJECT TITLE: SB Foods-Food ordering**

## **Team members:**

**1.Prakash.S[TL]**

**2.Dilli Ganesh. V[FRONT END DEVOLOPER]**

**3.Vijiya.s [BACK END DEVOLPOR]**

**4.Asha.T [QUALITY ASSURANCE]**

## **1.Introduction:**

The main objective of this project is to create a smooth, user-friendly web application that will enable consumers to quickly and easily order meals from the Fresh Palette Cafe system. Customers may browse menus, pick meals, personalise their orders, and make payments online thanks to the website's user-friendly design. We intend to enhance the ordering procedure for patrons and restaurant proprietors while also enhancing the entire eating experience by offering an online platform..

The food business is one of the biggest and most varied economic sectors, encompassing anything from fast food chains to fine dining establishments. For foodies who like to eat their favourite dishes without ever leaving the comfort of their homes, we offer an online food ordering service. The website is designed to be user-friendly for a wide range of users, including busy professionals, families, and tech-savvy millennials. The need for effective and user-friendly web apps to acquire food from fresh palette café systems that are available from anywhere, at any time, has increased due to the popularity of online shopping and the accessibility of mobile devices.

To address this demand, we created a web application using the Angular development framework that enables visitors to order meals from the fresh palette café website. A strong basis for creating dynamic and responsive web apps is provided by the well-known

JavaScript framework Angular. Dependency injection, two-way data binding, and a robust template system are just a few of the many features and capabilities it provides.

These features make Angular a fantastic choice for developing complex online apps, such as a website for placing food orders. The flexible and scalable architecture of Angular also enables developers to create reusable components and modules, reducing the time and expense required for development. In-depth information on the requirements analysis, design and architecture, implementation, testing, deployment, and maintenance of our web application for ordering meals from the fresh palette cafe website will be provided in this project report. We intend to draw a large audience of customers and boost the exposure and profitability of participating businesses by offering a quick and easy substitute for purchasing food online.

We will also discuss the website's benefits and drawbacks as well as potential modifications that may be done in the future to enhance its usability. We'll talk about the skills and knowledge we gained through building a difficult web application using Angular and how we may utilise them in future work. The primary elements of the website will be covered, including order tracking, payment processing, and user interface design. We will provide code snippets and screenshots of the website as it develops in addition to test results and any defects found during testing.

We anticipate that this initiative will be a valuable tool for companies and organisations in the food sector and that it will promote more innovation and development in this quickly growing sector. Overall, this project is an outstanding example of how Angular can be used to create cutting-edge online applications for purchasing meals from Fresh Palette Cafe's website.

## **Problem Statement**

Due to COVID-19 outbreak a reliable and efficient system for placing food orders is needed, one that enables customers to do so online and make secure payments. Certainly. The problem statement outlines the issue or challenge that the initiative aims to resolve. The problem in this situation is the typical way of ordering meals in a restaurant, which

may be time-consuming and unproductive, especially during busy hours. Additionally, the COVID-19 pandemic has compelled restaurants to adopt contactless ordering and payment procedures in order to safeguard the safety of customers and staff.

Going in person, perusing the menu, and putting your order with the server is the traditional way to order at a restaurant. However, this process could be time-consuming and unsuccessful, especially if restaurants are busy at the time.

Due to the pandemic, more people are deciding to use online ordering and delivery services, which has fundamentally altered the food industry. If they want to remain competitive, restaurants must provide customers a quick and simple way to order food online.

However, developing a web application to purchase meals from the fresh palette cafe website may be a challenging and complex process since it requires a range of technical skills and knowledge. Thus, the issue description underlines the need for a stable and efficient web application that enables customers to place orders online and make safe payments for meals from Fresh Palette Cafe.

## Objectives

- 1. Using Angular, create a web application that allows users to order meals from the fresh palette café website:** The main objective of this project is to design and build a meal ordering website using Angular. The creation of complex and scalable web programmes is possible thanks to the front-end programming framework Angular, which is well-liked and dependable. Angular may be used to create a responsive and userfriendly website that gives users a simple method to place food orders.
- 2. Make a web application with Angular that enables customers to place orders for food from the fresh palette café web application:** The main objective of this project is to design and build a meal ordering web application using Angular. The creation of complex and scalable web programmes is

possible thanks to the front-end programming framework Angular, which is well-liked and dependable. Angular may be used to create a responsive and user-friendly web application that gives users a simple method to place food orders.

- 3. Use user feedback and usability testing throughout the development process:** To guarantee a user-friendly and effective web application, take into consideration user feedback and usability testing at every level of the development process. Throughout the development process, user feedback must be gathered in order to adjust the web application towards this goal. Usability testing includes evaluating the usability of the web application and determining its weak points.
- 4. Optimize the website for performance and scalability:** For the website to handle large visitor levels during peak hours, performance and scalability optimisations must be made. Utilising recommended practises for web development, including image optimisation, page load time optimisation, and cache implementation, is necessary to achieve this goal.

The project's overall objectives are to utilise Angular to create a robust, scalable, and userfriendly online application for food ordering from the fresh palette cafe website that enables simple and efficient ordering for customers and streamlines the ordering process for restaurant owners.

## Scope

- 1. Technologies:** The following technologies will be used in the development of the web application for ordering meals from the fresh palette café website:  
MongoDB for database administration.  
Node.js and Express for server-side development;  
Angular for front-end development; Stripe for payment processing.

2. **Testing and Evaluation:** A mix of automated testing, human testing, and usability testing will be utilised in the testing and assessment process to ensure that the web application to purchase meals from the fresh palette cafe website is user-friendly, efficient, and secure.
3. **Timeline and Products::** The project plan will comprise the project timetable and the agreed-upon deliverables. Every stage of the development process will have specific deliverables, and a set timetable will be followed while creating the online web application to purchase food from the fresh palette cafe platform.
4. **Features and Functionality:** The website for ordering food will have the following options and features: User login and registration; menu browsing and ordering; customization options; payment processing and confirmation; order tracking and history; contact and feedback forms
5. **Development Process:** The creation of the web application to order food from fresh palette cafe website will adhere to the Agile technique, which entails incremental and iterative development, frequent testing, and ongoing stakeholder feedback.
6. **Target Audience:** The general objective of this project is to design, build, test, and evaluate a web application to purchase food from Fresh Palette's Cafe website using Angular, Node.js, and MongoDB, with a focus on user-friendliness, efficiency, and security. The project will follow an Agile methodology, have distinct deliverables, and follow a specified schedule. In the future, it could be improved depending on suggestions and demands from stakeholders.
7. **Future Enhancements:** Future updates to the fresh palette cafe website's online application that allows users to order meals may include additional capabilities and features including social media integration, reward programmes, and loyalty programmes. Stakeholders' opinions and requirements will be taken into consideration when these enhancements are implemented.

The general objective of this project is to design, build, test, and evaluate a web application to purchase food from Fresh Palette's cafe website using Angular, Node.js, and MongoDB, with a focus on user-friendliness, efficiency, and security. The project will follow an Agile methodology, have distinct deliverables, and follow a specified schedule. In the future, it could be improved depending on suggestions and demands from stakeholders.

#### **Front-end:**

- **HTML:** HTML is one of a web application's most essential building blocks. It stands for "Hypertext Markup Language". There are specific tags for each type of component, including paragraph, bold, italic, button, and form tags. The responsive front-end templates for the web application are made with Bootstrap.
- **CSS:** CSS is short for "cascading style sheet". It controls every design element, including colour, font, alignment, margin, and padding.
- You may develop interactive webpages using the server-side and client-side computer language JavaScript. The main purpose of it is to offer animations on the front end.
- **Angular:** An HTML and TypeScript-based framework and platform for building single-page client applications. Angular was developed using TypeScript.

#### **Back-end:**

1. **MongoDB:** The well-known NoSQL database MongoDB makes use of a document-oriented data architecture. It is possible to manage enormous volumes of data with outstanding speed, flexibility, and scalability. The project's MongoDB database is where the menu items, user information, and order information are all kept.
2. **Express:** A document-oriented data architecture is used by the popular NoSQL database MongoDB. Massive amounts of data may be managed with exceptional speed, adaptability, and scalability. The menu items, user data, and order data are all stored in the project's database, MongoDB.
3. **Mongoose:** Mongoose is an object data modelling (ODM) module for Node.js and MongoDB. It provides a higher level of abstraction in compared to MongoDB's native driver, making it easier to utilise MongoDB in Node.js applications. In this

project, Mongoose is used to generate the data structure, perform CRUD operations (create, read, update, delete), and validate data.

4. **JWT:** JSON online Token (JWT) is one of the often used authentication techniques for web applications. It offers a clear, secure way for parties to transfer claims. JWT is used in this project to create and validate tokens for user authentication and authorisation.
5. **Nodejs:** Back-end apps use this JavaScript framework. It is mainly utilised by servers that are event-driven and non-blocking.
  - a. A strong collection of capabilities are offered for both online and mobile apps by this Nodejs framework.
  - b. Mongo DB, a NoSQL-based database, and its library "Mongoose."
  - c. Express.js is a Node.js web application framework that is open-source and free. It is used to design and create web apps rapidly and simply.
6. **Nodemailer:** Email transmission is made possible using the Nodemailer Node.js package. In this project, order confirmation emails are sent to consumers using Nodemailer.
7. **Passport:** Passport is a well-known authentication middleware for Node.js. It provides a variety of authentication strategies for token-based, local, and social authentication, among other authentication mechanisms. The project implements local JWT authentication using Passport.

## 2. Project Overview

1. **Project Objectives:** The main goal of this project is to develop, test, and evaluate an online web application that allows users to order food from the Fresh Palette café platform using Angular, Node.js, and MongoDB. The website must be effective, secure, and easy to use.
2. **Project Background:** The Fresh Palette Cafe, which has a well-known location, seeks to increase its customer by offering online food shopping. Customers may use the website to browse the menu, customise their orders, and make payments.

- 3. Stakeholders:** Owners and managers of Fresh Palette Cafe, customers who will use the web application to order food from Fresh Palette Cafe's website, developers who will design, develop, and test the website, and testers who will assess the website's functionality and usability are the project's stakeholders.

**4. Project Deliverables:**

The project deliverables include:

- A requirements specification document
- A design specification document
- A development plans.
- A fully functional web application to order food from fresh palette cafe website.
- A user manual
- A test plan and test cases
- A project reports.

**5. Project Constraints:**

The project constraints include:

- Time: The project must be completed within a specific timeline.
- Budget: The project must be completed within a specific budget.
- Resources: The project must be completed with the available resources.
- Technology: The website must be developed using Angular, Node.js, and MongoDB.

- 6. Project Timeline:** According to an agile approach, the project timeline will include incremental and iterative development, regular testing, and continuing stakeholder feedback. The exact timeline and deliverables will be outlined in the project plan in collaboration with the stakeholders. Creating, testing, and evaluating a website for a café named Fresh Palette café while adhering to time, money, resource, and technology constraints are generally outlined in the project description for this Angular web application to order meals from the fresh palette cafe website. The project will follow an agile methodology, have clear deliverables, and involve a range of stakeholders with different roles and goals. The programme aims to meet client expectations and provide the managers and owners of Fresh Palette with a lucrative return on their investment.



**7. Assumptions and Risks:** Project assumptions and hazards include:

- **Assumptions:** The developers have the knowledge and expertise necessary to plan, create, and test the website. Throughout the development phase, the stakeholders will offer concise and consistent input.
- **Risks:** The project might be delayed or derailed by technical problems, such as server failures or problems with payment processing. The website could not be efficient or user-friendly, which would result in poor adoption rates.

### 3.ARCHITECTURE

Today, engaging users to websites and mobile applications depends heavily on good design. To identify the particular elements that go into successful website and mobile application design, however, not much research has been done. We aim to gather information on effective design in this literature review and give a selection of widely used research features to help designers and researchers operationalize best practises for encouraging and anticipating user engagement. The scope of this literature survey includes:

- **Frontend:** Usability as a key design element: The International Standardisation Organisation (ISO) defines usability as the effectiveness, efficiency, and satisfaction with which users may achieve their objectives. However, there isn't currently consensus on how to operationalize and assess website usability.
- **Backend:**The importance of website design: The majority of businesses and organisations now rely primarily on their websites for public communication due to the increase in internet usage over the past 10 years. While poorly designed websites have been shown to negatively effect visitor retention and purchase behaviour, well-designed websites with excellent usability have been shown to have a positive impact on both.
- Limitations and future research: Although this literature review offers a foundation for future study, more investigation and description of website design factors that affect user engagement are still required. To assist designers and researchers in creating successful website designs, clear criteria are needed.
- **Database:**The seven design elements that appear the most frequently in the literature under review are the focus of this literature assessment. Some of these elements include readability, navigation, graphical representation, arrangement, and the utility of the material. In prior studies, these elements were constructed and evaluated to see how they affected user engagement.

This review of the literature offers a thorough overview of the significance of website design, emphasises usability as a crucial component of the design, and offers a shortlist of seven design components that increase user engagement. Exploratory research can give definitions for website design aspects as well as a place to start for additional exploration.

Authors	Published Title	Technology	Description
Mohamed Sultan [1]	Angular and the Trending Frameworks of Mobile and Web-based Platform Technologies: A Comparative Analysis, FTC, 2017	Angular	The essay analyses five wellknown frameworks and compared 12 distinct frameworks for mobile and online apps. The noteworthy features and advantages of the new Angular2 framework above existing frameworks are highlighted. The finding raises the prospect of the emergence of a new generation of frameworks shortly.
Verhaeghe B, Shatnawi A et al. [2]	From GWT to Angular: An Experiment Report on Migrating a Legacy Web Application, IEEE,2021	Angular	.We designed a semiautomated migration approach that helps developers migrate applications' front ends from GWT to Angular and a tool that performs the migration.
Ziping Liu and Bidyut Gupta [3]	Study of Secured Full-Stack Web Development, CATA 2019	Full Stack Development	The article explores recommended practises for creating a safe online application utilising Angular and ASP.NET core frameworks, complete with example code, and reviews layered architecture and the MVC pattern for web development. It also addresses typical vulnerabilities in web applications.

Anirudh Bhaskar, Manjunath A.E [4]	An Interpretation and Anatomization of Angular: A Google Web Framework, IRJET,2020	Angular	In this paper, a thorough overview of the origin and history of Angular is given along with a detailed explanation of the key features which makes Angular so unique and widely used in the industry. Finally, an insight to testing is given and the paper is wound up with the advantages of Angular and the future scope of Angular.
Domokos CE, Séra B, Simon et al. [5]	<b>Netfood: A Software System for Web application to order food from fresh palette cafe and Delivery</b>  , IEEE 2018	Web Application Development	Customers may concurrently place individual or group orders from several restaurants using Netfood, an order management system focused on deliveries. A online interface allows users to place orders.
Abd Wahab MH, Kadir et al.[6]	, IEEE,2008  <b>Implementation of network-based smart order system</b>	Full Stack Development with focus on Angular	This article discusses the creation of a smart order system for restaurants with the goal of enhancing data management and communication. A hardware design phase and a system development phase make up the system development life cycle. The outcomes demonstrate that the system is efficient and that data transmission operates as anticipated.

**Table 2.1:** The Table shows the Author's name, the Proposed Approaches, the Journal it was published in, the Year of publication and their technologies.

## 4.Setup instructions:

The following services have been developed for this web application to order food from fresh palette cafe online project's system design:

- **Cart Services:** The functioning of the website's shopping cart is controlled by the cart service. Users may modify or delete things from their carts as well as examine the total cost of their carts.
- **Food Services:** This department is in charge of all activities involving food, including changing the menu to include new items and deleting older ones. The administration of food item categories such as appetisers, entrées, main meals, and desserts is also under its purview.
- **User Service:** This service is responsible for handling all user-related tasks, such as user registration, login, authentication, and authorisation. User profiles, order histories, and user reviews are also preserved.

In addition to these services, the system design includes the following:

- **Front-end:** It is a user-friendly and responsive design. There are websites for placing food orders, maintaining carts, signing in, and creating an account. The front-end and back-end services may communicate via APIs.
- **Database:** To meet the needs of the services, the database structure was created. To keep track of all the data on customers, carts, and food products, a database is developed.
- **API:** APIs were created to allow communication between all services and the front end of the website. The front-end functions that are required, such as user authentication, cart management, and a web application for ordering meals from fresh palette café, are made available through the APIs.

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
styles.css, styles.js | styles | 219.05 kB |
main.js | main | 106.52 kB |
runtime.js | runtime | 6.51 kB |
| Initial Total | 3.35 MB

Build at: 2023-05-03T19:08:41.072Z - Hash: d72d351209b35ebf - Time: 3867ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.

```

Frontend Runs on localhost:4200

```
PS C:\Users\Atishya Jain\Desktop\FP-cafe\Backend> npm start

> backend@1.0.0 start
> cd src && nodemon server.ts

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: ts,json
[nodemon] starting `ts-node server.ts`
Website is Running on http://localhost:5000
```

Backend Runs on localhost:5000

## Prerequisites:

A payment gateway is integrated with the website allow users to make payments online securely.

The following services can be used for the development of the web application to order food from fresh palette cafe website:

1. **Nodejs:** Node.js is the first: Using Node.js, a server-side JavaScript runtime, the back-end of the web application to order meals from the fresh palette café website may be developed. It offers a non-blocking I/O paradigm to manage large amounts of data.
2. **Angular Framework:** Using the well-known front-end development framework Angular, the user interface of the website for the online web application to purchase meals from Fresh Palette Cafe may be constructed. It includes features that speed up and optimise development, like dependency injection, data binding, and component-based design.
3. **Express.js:** Using the popular Node.js framework Express.js, the server-side application for the web application to order meals from the fresh palette café website may be constructed. Its middleware, routing, and other features expedite and boost the efficiency of development. It's an open-source programme created especially for web applications and APIs. It is a quick and easy framework that

enables the creation of web apps quickly, easily, and with the least amount of code.

Benefits of it include:

- Has get and post request middleware components.
- It is simple to combine with a variety of template engines, including Jade, Vash, and EJS.
- It is possible to connect to databases like MongoDB, MySQL, and Redis with ease.
- It also offers middleware for managing errors.
- Enables us to define routes using URLs and HTTP methods.

4. **MongoDB:** A NoSQL database called MongoDB may be utilised to store the information for the food delivery website. Scalability, versatility, and high performance are qualities that it offers, making it the best choice for processing massive volumes of data.
5. **Firebase:** The backend of the online web application for ordering meals from the fresh palette cafe system might be built using the popular backend-as-a-service (BaaS) technology called Firebase. It provides features that speed up and optimise development, such as real-time databases, cloud storage, and authentication.
6. **Body-Parser:** A middleware for Node.js parsing, to be precise. It is in responsible of parsing the incoming request body before we handle a request. In order to move data from the back end to the front end for presentation, it is widely used in our project to extract the names of various components, particularly forms.
7. **Strip:** Strip, a well-known payment processor, is used to handle secure online payments on the web application for ordering meals from the fresh palette café website. It is a well-liked substitute for online businesses since it provides benefits including easy integration, a variety of payment options, and fraud prevention.

## Installation:

The system architecture design of the web application to order meals from Fresh Palette Cafe online project is built on the Model-View-Controller (MVC) architectural pattern.

MVC is a well-liked design pattern for making web applications. The three interrelated components of the application are the Model, the View, and the Controller.

RESTful APIs are also used in the project to create a web application for ordering meals from the fresh palette café website. The APIs provide consistent communication between the client and the server. A set of RESTful APIs that the server exposes may be accessed by the client via HTTP requests. The APIs are developed using Express.js, a well-liked Node.js framework for building web applications.

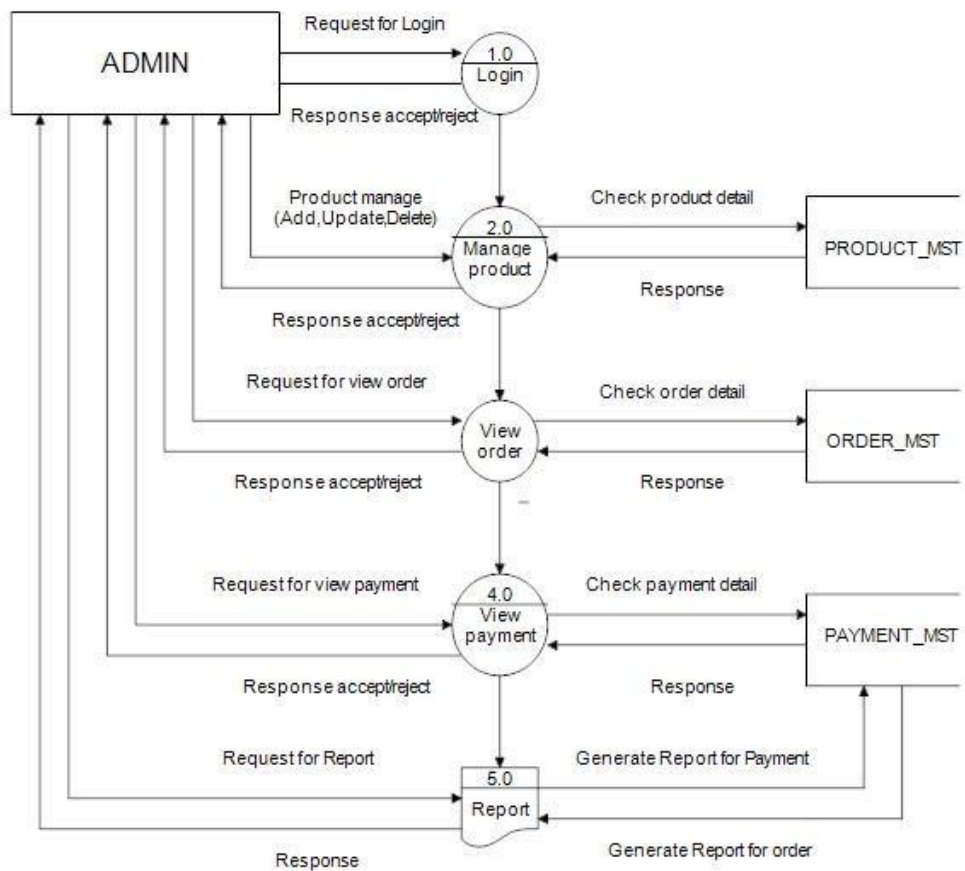
The system architecture of the website project for the online application to order meals from Fresh Palette Cafe is based on a client-server architecture, in which a web browser acts as the client and a web server hosts the website application. The client-server architectural paradigm in web application is frequently utilised. The HTTP protocol is used in this setup for all online interactions among the client and server.

The popular JavaScript server-side runtime framework Node.js is used to create the Controller component. After taking requests from the View and getting data from the Model, it answers to the View's inquiries. The relationship between each of the View and the Model components is controlled by the Controller component of the website project for the Fresh Palette Cafe online application.

The Controller element of the web application to order meals from fresh palette cafe website project manages the link between the View and the Model components. It responds to the View's queries after receiving information from the Model and accepting requests from the View. The Controller component is developed using Node.js, a wellliked JavaScript server-side runtime environment

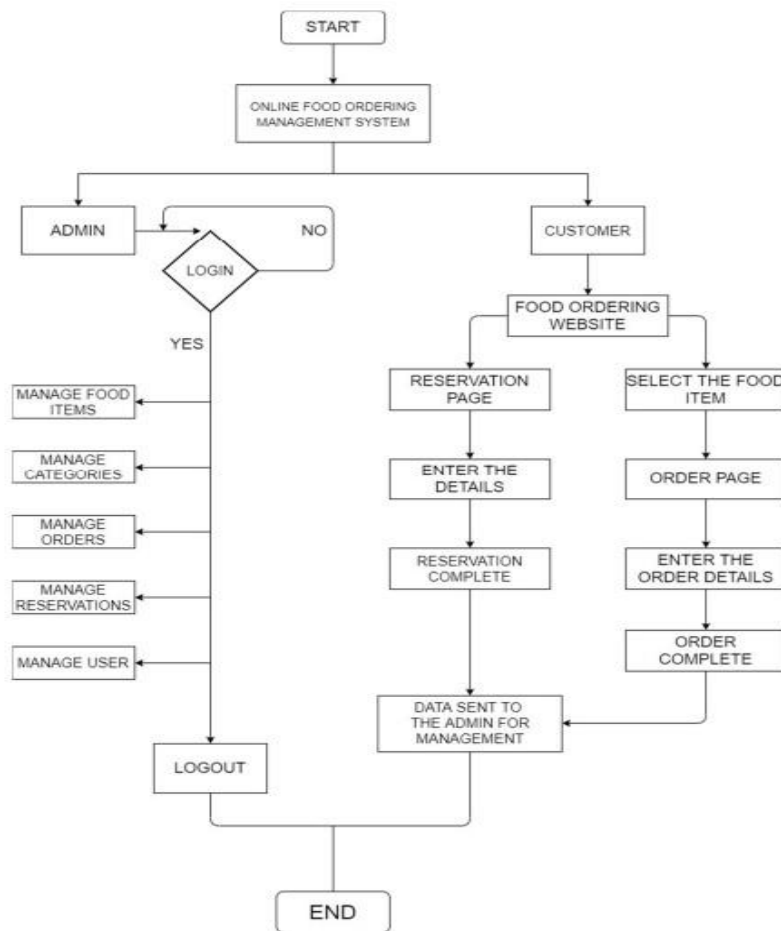


## 5.Folder structure

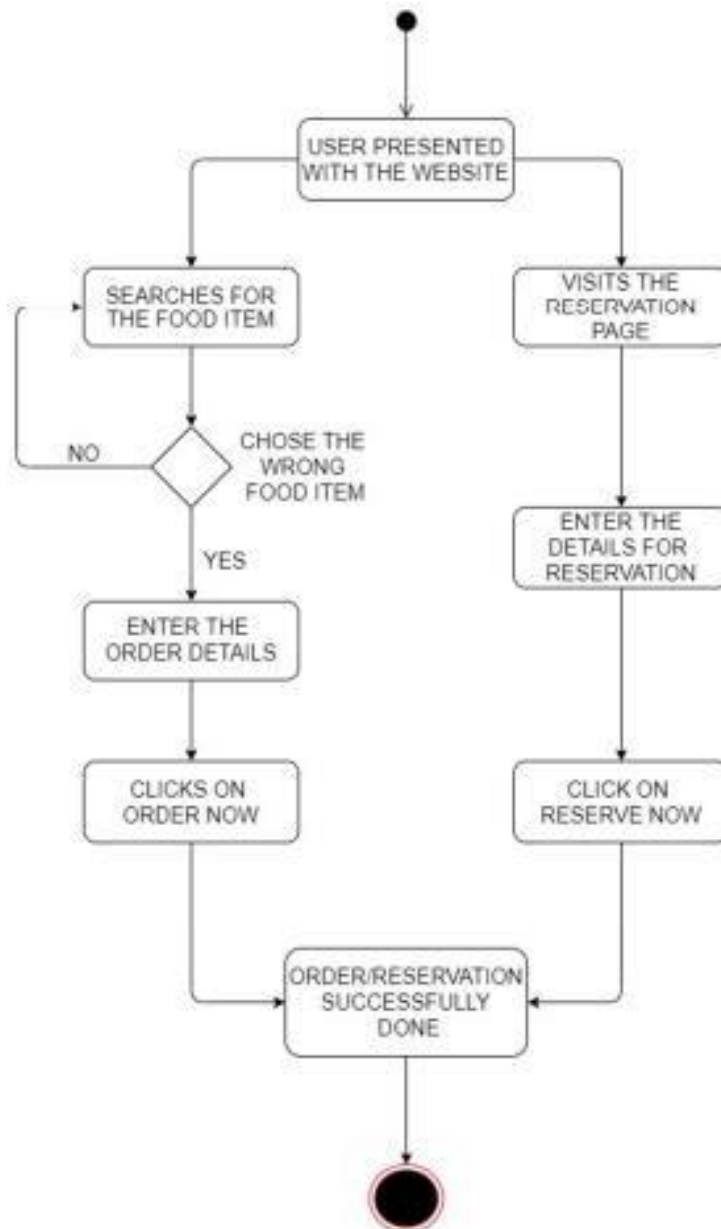


Schematic diagram

## Data Flow Diagram

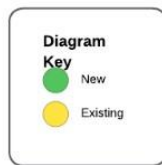


Flow char



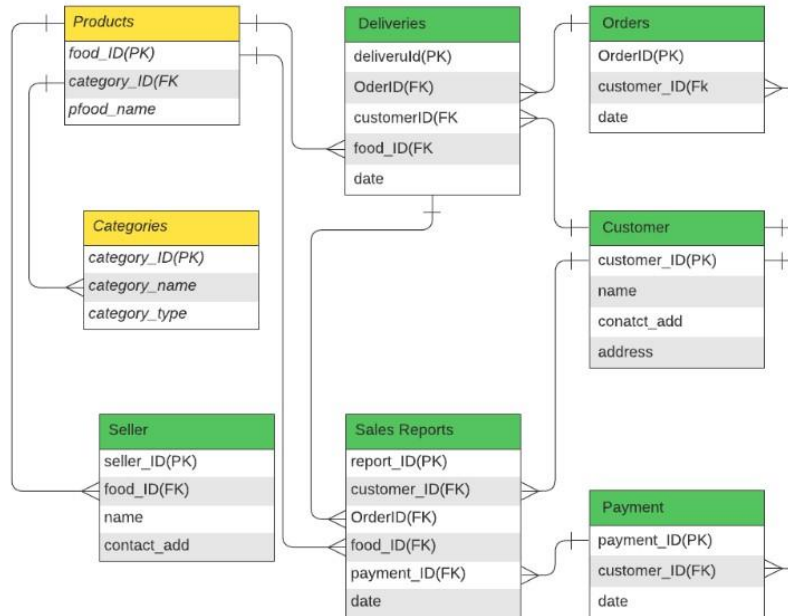
Customer Flow chart

## Database Design



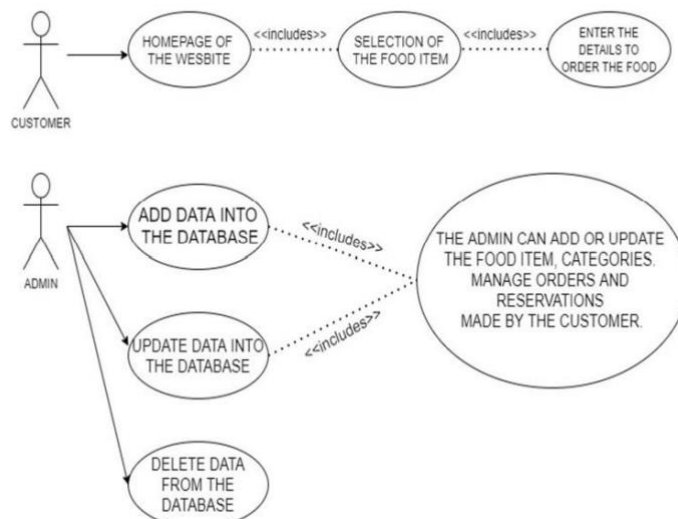
### ERD with colored entities (UML notation)

Atishya Jain | May 4, 2023



## Database Design for web application to order food from fresh palette cafe

### Used Case Diagram



Used case diagram

## 6. Running the Application

## Frontend:

1. **Dependency Injection:** A design strategy known as dependency injection manages object dependencies in software systems. It makes the code more modular and maintainable, provides dependencies on other parts, services, and areas of the application in Angular.
2. **Reactive programming:** A design strategy known as reactive programming manages object dependencies in software systems. It makes the code more modular and maintainable, provides dependencies on other parts, services, and areas of the application in Angular.

### Dependencies:

The different types of Dependency Injection used in a web application to order food from fresh palette cafe website built with Angular:

**NgModule Injection:** To summarise, NgModule is a crucial part of Angular's dependency injection design. It provides a technique for configuring the DI system and specifying the compilation context in which the system operates. NgModule's several dependence injection types—providers, declarations, imports, exports, and bootstrap—allow programmers to construct complex applications by reducing large apps into smaller, more manageable modules.

**DevDependencies:** All devdependencies that are used for development purpose:

- **@angular-devkit/build-angular:** This package offers the settings and tools required to create and run Angular apps. It is a component of the toolkit for Angular CLI.
- **@angular/cli:** The Angular CLI is a command-line interface tool that gives developers the ability to build brand-new Angular projects, produce components, services, and modules, and carry out a number of other development activities.
- **@angular/compiler-cli:** This package contains a command-line tool for converting TypeScript and Angular templates into JavaScript code.
- **@types/jasmine:** TypeScript type definitions for the Jasmine testing framework are contained in this package.

typescript: A superset of JavaScript, typescript enhances the language with optional static typing and other capabilities. It serves as the writing language for Angular apps. Although these dependencies are not necessary for the application's production version, they are crucial for development

## Backend:

Through the use of property injection, dependencies may also be injected into a component or service. In property injection, the dependencies are specified as public properties of the component or service. When the class is built, Angular immediately injects the dependency because it is defined in the class as a property. The Angular framework generates another instance of the component or service and modifies the values of the attributes to indicate the necessary dependencies. Contrary to constructor injection, this kind of injection is less frequent. The dependency injection strategy, also known as property injection in Angular, involves injecting a dependency into a class using a public property. The following are the various types of property injections used in the given project:

- The Angular package Ngx-Toastr provides a toast notification component. It is injected into the component via a property injection.
  - The Angular library Ng-starrating provides a star rating component. It is injected into the component via a property injection.
- JavaScript library for reactive programming called Rxjs. It is injected into the component via a property injection.
- Node.js: This module makes it possible to use Angular applications inside of zones. It is injected into the component via a property injection.

Property injection is a useful technique to add dependencies to a class because there is no need to write the constructor for the class..A brief explanation of each of these dependencies is also feasible. It is also feasible to discuss the advantages and disadvantages of property injection. Because the dependencies are not expressed explicitly in the constructor, property injection may make it more difficult to study the code. The component utilises properties injection to inject the required parts for ng-starrating, ngxtoastr, rxjs, and zone.js, as can be seen in the project report.

```

"ng-starrating": "^1.0.20",
"ngx-toastr": "^16.1.1",
"rxjs": "~7.8.0",
"tslib": "^2.3.0",
"zone.js": "~0.12.0"

```

Property Versions

```

"devDependencies": {
  "@angular-devkit/build-angular": "^15.2.4",
  "@angular/cli": "~15.2.4",
  "@angular/compiler-cli": "^15.2.0",
  "@types/jasmine": "~4.3.0",
  "jasmine-core": "~4.5.0",
  "karma": "~6.4.0",
  "karma-chrome-launcher": "~3.1.0",
  "karma-coverage": "~2.2.0",
  "karma-jasmine": "~5.1.0",
  "karma-jasmine-html-reporter": "~2.0.0",
  "typescript": "~4.9.4"
}

```

Dev dependencies included in project

## 7.API Documentation

Building Angular apps requires the Angular framework and. The following is a quick description of each packag

There are four fundamental files within the world of browser-based angular development:

@angular/platform-browser,

@angular/platform-browser-dynamic,

@angular/router,and

@anuglar/animations:-

Opting out of these particular packages would impede your ability to produce efficient and effective applications using this framework.The various functionalities which each file

provides include but not limited to obtaining necessary angular code required by browsers such as with PlatformBrowserDynamic function , bootstrap capability for dynamic processes within browsers with PlatformBrowserDynamic package use , implementation of robust routing mechanisms through router package and ability introduce animation fortifications via animations library.

- `@angular/platform-browser`: This file provides the Angular code required for a browser to run Angular applications, such as the `platformBrowserDynamic` function.
- `@angular/platform-browser-dynamic`: Provides a set of tools for bootstrapping dynamic Angular applications in the context of a browser.
- `@angular/router`: Provides a robust routing mechanism for Angular projects.
- `@angular/animations`: This package gives Angular apps the ability to use animation.
- `@angular/common`: Comprises the pipes, directives, and structural directives `ngIf` and `ngFor` that are common to all Angular applications.
- `@angular/compiler`: The Angular compiler turns your Angular templates and components into JavaScript code that can be executed.
- `@angular/core`: The primary Angular library contains tools for constructing directives, pipelines, services, components, and other things.
- `@angular/forms`: Helps Angular app developers create forms.

These packages are often loaded as dependencies when you start a new Angular project using the Angular CLI, and they are instantly imported in your application's modules when you add new components, services, and other Angular artefacts.



```

"dependencies": {
  "@angular/animations": "^15.2.8",
  "@angular/common": "^15.2.0",
  "@angular/compiler": "^15.2.0",
  "@angular/core": "^15.2.0",
  "@angular/forms": "^15.2.0",
  "@angular/platform-browser": "^15.2.0",
  "@angular/platform-browser-dynamic": "^15.2.0",
  "@angular/router": "^15.2.0",
  "ng-starrating": "^1.0.20",
  "ngx-toastr": "^16.1.1",
  "rxjs": "~7.8.0",
  "tslib": "^2.3.0",
  "zone.js": "~0.12.0"
},

```

Version of Angular Framework us

A collection of instructions that may be used to create, test, and launch the application are contained in the "scripts" section of the package.json file of a standard Angular project. These commands are essentially shortened versions of the foundational resources that Angular offers.

- **start:** Execute this script to start the development server. When you run this command, a development server and a browser window referencing the app are launched.
- **Ng:** The Angular framework may be accessed with this command-line interface tool.
- **test:** This script is used to run the application's tests. When you use this command, it runs each of the application's tests and generates a report.
- **note:** This script rebuilds the application whenever the code is modified. When you give this command, the programme is automatically rebuilt while monitoring for changes to the code.
- **build:** Use this script to construct the production version of the application. This command creates a software that is ready for production under the dist/ folder.

The creation and deployment of Angular apps can benefit greatly from these scripts. They offer a straightforward and user-friendly method of interacting with the framework and automating typical development operations.

## TypeScript Configuration

```
tsconfig.json > {} compilerOptions > [ ] lib > 1
1  /* To learn more about this file see: https://angular.io/config/tsconfig. */
2  {
3    "compileOnSave": false,
4    "compilerOptions": {
5      "baseUrl": "./",
6      "outDir": "./dist/out-tsc",
7      "forceConsistentCasingInFileNames": true,
8      "strict": true,
9      "noImplicitOverride": true,
10     "noPropertyAccessFromIndexSignature": false,
11     "noImplicitReturns": true,
12     "noFallthroughCasesInSwitch": true,
13     "sourceMap": true,
14     "declaration": false,
15     "downlevelIteration": true,
16     "experimentalDecorators": true,
17     "moduleResolution": "node",
18     "importHelpers": true,
19     "target": "ES2022",
20     "module": "ES2022",
21     "useDefineForClassFields": false,
22     "lib": [
23       "ES2022",
24       "dom"
25     ],
26   },
27   "angularCompilerOptions": {
28     "enableI18nLegacyMessageIdFormat": false,
29     "strictInjectionParameters": true,
30     "strictInputAccessModifiers": true,
31     "strictTemplates": true
32   }
33 }
34
```

Shows all the TypeScript configuration

# 8.Authentication

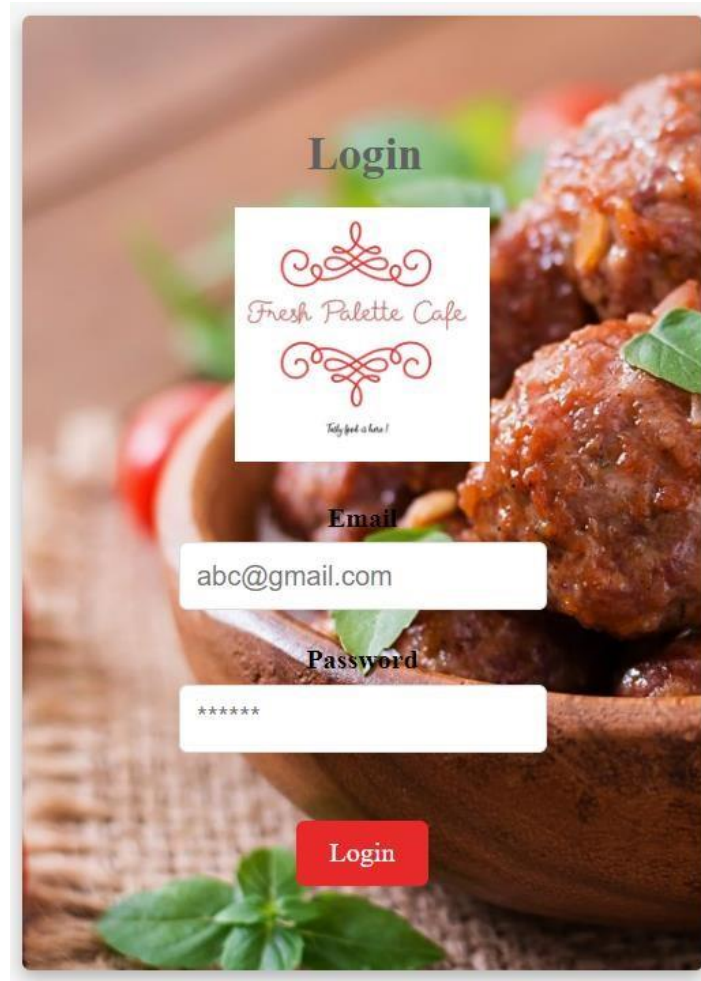
## Functional Requirements

### User Login:

- Registered users should be able to log in to the website using their email address and password.

### User Registration:

- The registration form should include adequate validation checks to ensure that the user submits legitimate information.



**Figure : Login Layout**

**Forgot Password:** If a user loses their password, they should have the opportunity to reset it. To provide users a link to reset their passwords, the "Forgot Password" option on the login page needs to ask for their email address.

**Email Verification:**

Users should get an email with a verification link after registering on the website; they must click the link to verify their email address before they can access the website.

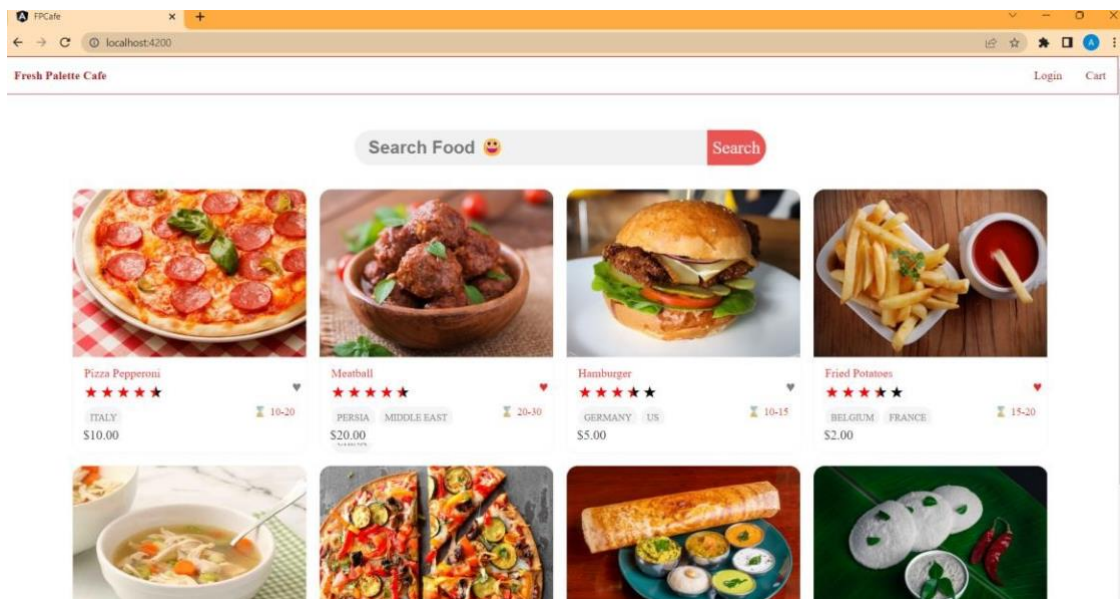
**Social Media Login:**

Users should have the option of logging in with their Google, Facebook, and other social media accounts, but suitable authentication methods should be implemented to ensure that the user's social network account is valid.

## User Profile Management:

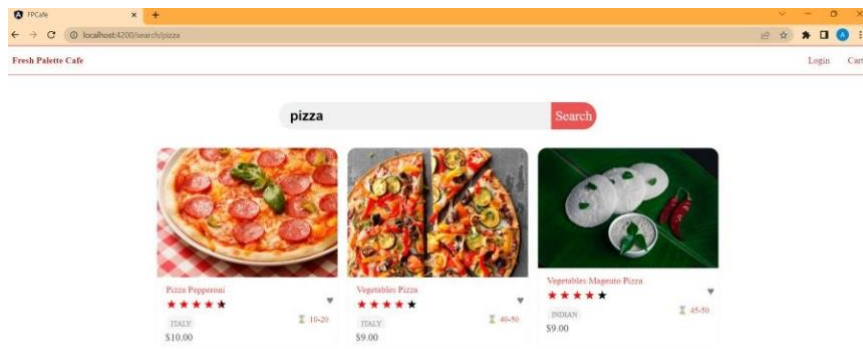
- Users should be able to modify the information on their profiles and have the changes remain in effect.
- Users who have registered should have access to their profile details, including name, contact information, and password.

**Menu display:** This function shows a menu with all the food options, their costs, and other information like ingredients, nutrition information, etc. Users may choose their preferred things and place orders with its assistance.



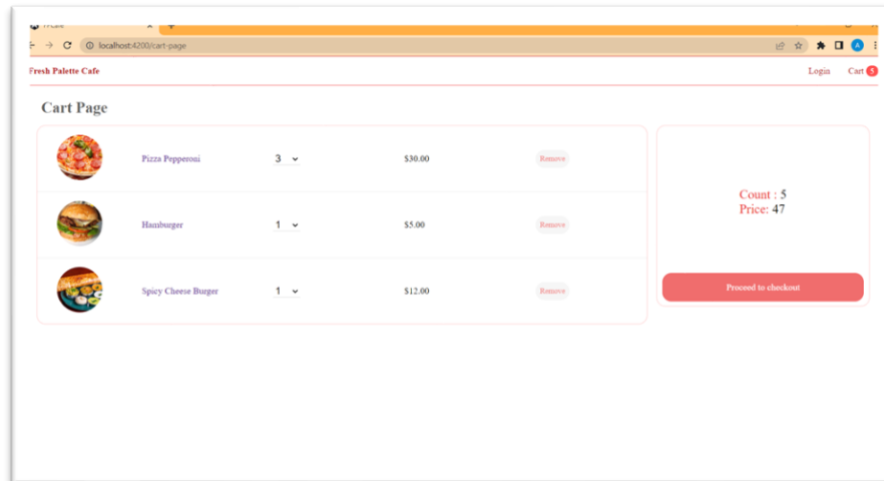
Menu display

**Search functionality:** Users may search the website for certain restaurants or food products using this tool. It saves customers time and enables them to rapidly discover their favourite things.



## Search sample

**Order placement and payment:** Users may securely place orders and make payments on the website thanks to this functionality. Users should have the ability to check the progress of their orders and, if necessary, cancel them.



## Cart Page

**Restaurant management:** This feature allows restaurant owners to update their profiles, change their menus, and monitor client orders. It makes it possible for restaurant owners to easily plan their operations, control their inventory, and monitor their sales

**Rating and feedback system:** Using this feature' users may rate and discuss their meal and delivery experiences. It helps other users make decisions while placing their orders and helps the website improve its offers in response to user input..

**Notifications:** This function notifies consumers of the status of their orders, updates on delivery, and other crucial website-related information. It. keeps website visitors informed and interested.

**Customer support:** Through a variety of channels, including chat, email, and phone, this service offers consumers customer help. It improves user experience and assists users in resolving website-related questions and problems.

**Social media integration:** This service allows users to submit details about their food orders and experiences on social media platforms like Facebook and Instagram.

It facilitates website promotion and increases system traffic.

## 9. User interface

### 1. Performance:

- The website must respond to user input in under two seconds.
- The website can support up to 500 users at once.
- The website must be accessible at least 99.9% of the time.

### 2. Security:

- User authentication should be implemented using secure methods, such as hashing and salting passwords, in all server-to-client communications on the website.
- To guarantee that users may only access the areas of the website that they are authorised to, the website must implement role-based access control.

### 3. Usability:

- The website must be responsive and mobile device-optimized, with a straightforward and user-friendly user interface.
- The website must give consumers brief, clear directions on how to execute activities.

### 4. Scalability:

- The website must be scalable horizontally, meaning it must be able to add more servers to meet rising traffic without noticeably degrading its performance.
- During peak periods, the website must be able to manage a tenfold increase in traffic.

### 5. Reliability: The website must be able to recover from server issues and failures within 30 seconds and have automated backups to avoid data loss in the case of a

disaster. The website needs a monitoring system so that it can detect errors and notify the development team.

#### **6. Maintainability:**

- o The website has to be modular, follow the SOLID principles, and have automated tests to ensure that changes don't break functionality that currently exists.
- o The website's rich documentation makes the code and system architecture simple for developers to understand.

To ensure that each of these non-functional criteria is addressed, it would be desirable to integrate exact measurements and measures. The document may also include diagrams of network topology or system architecture to illustrate the design decisions made in order to meet the non-functional requirements.

## **10. Testing**

### **Experimental Setup:**

The experimental setup section outlines the hardware and software requirements needed to successfully run the web application to order food from fresh palette cafe website project.

### **Hardware Requirements**

The web application to order food from fresh palette cafe website project may be operated on a number of hardware setups, however for best performance, the following recommendations are made:

- Display: 14 inches or bigger with a resolution of 1920 x 1080 pixels or higher
- Processor: Intel Core i5 or higher
- RAM: 8 GB or higher
- Storage: 256 GB or higher
- Network: Ethernet or Wi-Fi

### **Software Requirements**

The meal ordering online project needs the software and equipment listed below to function:



- Operating system: Linux, macOS, or Windows 10

Node.js must be version 14.17.0 or later, and Angular CLI must be version 12.2.0 or later.

- Version 4.4.9 or higher of MongoDB

The project's tools and software were all purchased from their official websites and set up in accordance with the installation manuals. On a machine with the suggested hardware and software specs, the experimental configuration was tested.

## Implementation

### Models:

#### User Model:

```
src > app > shared > model > TS User.ts > User > id
1  export class User{
2      id!:string;
3      email!:string;
4      name!:string;
5      address!:string;
6      token!:string;
7      isAdmin!:boolean;
8  }
9
```

Figure 4.2.1.1: User Model

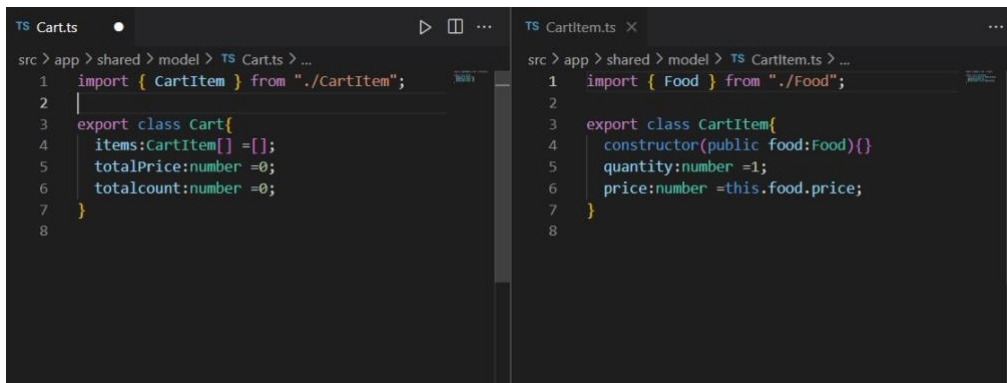
#### Food Model:

```
src > app > shared > model > TS Food.ts > Food
1  export class Food{
2      id!:string;
3      name!:string;
4      price!:number;
5      tags?:string[];
6      favorite!:boolean;
7      stars!:number;
8      imageUrl!:string;
9      origins!:string[];
10     cookTime!:string;
11 }
12
```

Figure: Food Model – All variables created for a food item **Cart**

#### Model:





**Figure :** Cart and CartItem Model

## Services

We have used total three services for user , cart and food details .

### User Service:

User support for the fresh palette cafe's online food ordering system The website project is in charge of overseeing user-related tasks including login and logout management. ToastrService and HttpClient are only two of the dependencies that are needed by this injectable class in Angular.

The login() method of the service sends a POST request and an IUserLogin object to the backend server using the HttpClient module. The setUserToLocalStorage() method sets the user information to the local storage if the login is successful. If the login attempt is unsuccessful, it shows an error message using the ToastrService.

User information is removed from the BehaviorSubject and deleted from local storage by the service's logout() function. In order to avoid any problems with the previous user's information, the window is also reloaded.

The User class's private BehaviorSubject, which contains the user's most current data, may be subscribed to by other components. Additionally, it has an Observable that the userSubject may utilise to emit events if the user data changes.

In general, the User Service plays a critical role in the authentication and authorisation procedures for the Fresh Palette Cafe Website project, providing the safe handling of the user's sensitive data.

```

TS user.service.ts X
src > app > services > TS user.service.ts > UserService > login > error
1  import { Injectable } from '@angular/core';
2  import { BehaviorSubject, Observable, tap } from 'rxjs';
3  import { User } from '../shared/model/User';
4  import { IUserLogin } from '../shared/interfaces/IUserLogin';
5  import { HttpClient } from '@angular/common/http';
6  import { USER_LOGIN_URL } from '../shared/constants/url';
7  import { ToastrService } from 'ngx-toastr';
8
9  const USER_KEY=''
10 @Injectable({
11   providedIn: 'root'
12 })
13 export class UserService {
14
15   private userSubject = new BehaviorSubject<User>(this.getUserFromLocalStorage());
16   public userObservable:Observable<User>
17   constructor(private http:HttpClient, private toastrService:ToastrService) {
18     this.userObservable=this.userSubject.asObservable();
19   }
20
21   login(userLogin:IUserLogin):Observable<User>{
22     return this.http.post<User>(USER_LOGIN_URL, userLogin).pipe(tap({
23       next: (user) => {
24         this.setUserToLocalStorage(user)
25         this.toastrService.success('Welcome to Fresh Pallete Cafe ${user.name}!',
26           'Login Successful !');
27       },
28       error: (errorResponse) => {
29         this.toastrService.error(errorResponse.error, 'Login Failed !');
30       }
31     }));
32   }
33   logout(){

```

```

TS user.service.ts X
src > app > services > TS user.service.ts > UserService > login > error
28     error: (errorResponse) => {
29       this.toastrService.error(errorResponse.error, 'Login Failed !')
30     }
31   }));
32 }
33 logout(){
34   this.userSubject.next(new User());
35   localStorage.removeItem(USER_KEY);
36   window.location.reload();
37 }
38 private setUserToLocalStorage(user:User){
39   localStorage.setItem(USER_KEY,JSON.stringify(user));
40 }
41 private getUserFromLocalStorage():User{
42   const userJason = localStorage.getItem(USER_KEY);
43   if(userJason)
44     return JSON.parse(userJason) as User;
45   return new User();
46 }
47 }
48

```

**Figure a- b:** User Service showing login and logout functionality

**Food Service:**

The FreshPalette Cafe's meals may be ordered using the Web application's FoodService section. The website project is responsible for providing the functionalities needed to obtain food items from the server. It provides three main ways to access food: `getAll`, `getAllFoodBySearchTerm`, and `getFoodById`.

By providing these options, the FoodService enables the other application components to get food items from the server and display them to the user. This service is crucial to the operation of the Web application to order food from the fresh palette cafe Website because it provides the consumer with the crucial information about food goods.

The `getAll` method retrieves every food item that is available and located on the server. The resulting Observable of type `Food[]`, which includes the food items, may be subscribed to by any component that wishes to display the food items.

The `getAllFoodBySearchTerm` method retrieves all of the food items using the user-provided search term. The query is prefixed in the server's URL, and the items of food that were located are then returned as an Observable of type `Food[]`.

The `getFoodById` method retrieves a single food item based on its ID. The food ID is added to the server's URL to generate a food item, which is then returned as an Observable of type `Food`.

```

TS food.service.ts ×
src > app > services > TS food.service.ts > FoodService > getFoodById
1  import { Injectable } from '@angular/core';
2  import { sample_foods } from 'src/data';
3  import { Food } from '../shared/model/Food';
4  import { HttpClient } from '@angular/common/http';
5  import { Observable } from 'rxjs';
6  import { FOODS_BY_ID_URL, FOODS_BY_SEARCH_URL, FOODS_URL } from '../
7
8
9  @Injectable({
10   providedIn: 'root'
11 })
12 export class FoodService {
13
14   constructor(private httpClient:HttpClient ) { }
15   getAll():Observable<Food[]>{ //to return the sample data
16     return this.httpClient.get<Food[]>(FOODS_URL) //now data of s
17   }
18   //Search Food
19   getAllFoodBySearchTerm(searchTerm:string){
20     return this.httpClient.get<Food[]>(FOODS_BY_SEARCH_URL + searchT
21   }
22   //get Food by ID
23   getFoodById(foodId:string):Observable<Food>{
24     return this.httpClient.get<Food>(FOODS_BY_ID_URL + foodId)
25   }
26 }
27

```

**Food Service** typescript file access all details of a food item for home page, at the time of search of an item and seeing a food item by its ID .

### Cart Service:

The CartService is in charge of overseeing the user's cart in the web application to order food from fresh palette cafe website project. It manages all cart activities, including the addition of food items, removal of food items, adjusting the number of food items, and clearing of carts.

To store the cart items and their data, including the food item, its amount, and its pricing, it employs a Cart model. To represent a specific item in the cart, it also employs a CartItem model.

The service tracks modifications to the cart using a BehaviorSubject and notifies any subscribers of these modifications. In order to maintain the cart information between page refreshes, it additionally keeps it in the localStorage section of the browser.

The CartService is responsible for managing the shopping cart on the meal ordering website. It keeps track of the state of the cart using a Cart object, which contains an array of CartItem objects that represent the items in the cart. Whenever the cart is modified, it also contains a BehaviorSubject that is used to emit the modified cart object.

The removeFromCart method eliminates an item from the cart by filtering the items array in the Cart object to remove the item with the given foodId. Following that, the setCartToLocalStorage method is called, which sends the changed cart through the BehaviorSubject.

The addToCart method adds an item to the cart by creating a new CartItem object from the Food object that was sent to the function and placing it in the items array of the Cart object. If the item is already in the cart, the method does nothing and exits. The updated cart is then emitted through the BehaviorSubject when the setCartToLocalStorage method has been invoked to update the cart in local storage.

By locating the CartItem object with the given foodId and changing its amount and price values, the changeQuantity method modifies the quantity of an item in the cart. The modified cart is then sent through the BehaviorSubject by calling the setCartToLocalStorage function.

Use the setCartToLocalStorage method to modify the cart in local storage and emit it through the BehaviorSubject. It calculates the total cost and total number of things in the cart by using the reduce method on the items array of the Cart object. The modified Cart object is then saved locally as a JSON string after being communicated via the BehaviorSubject.

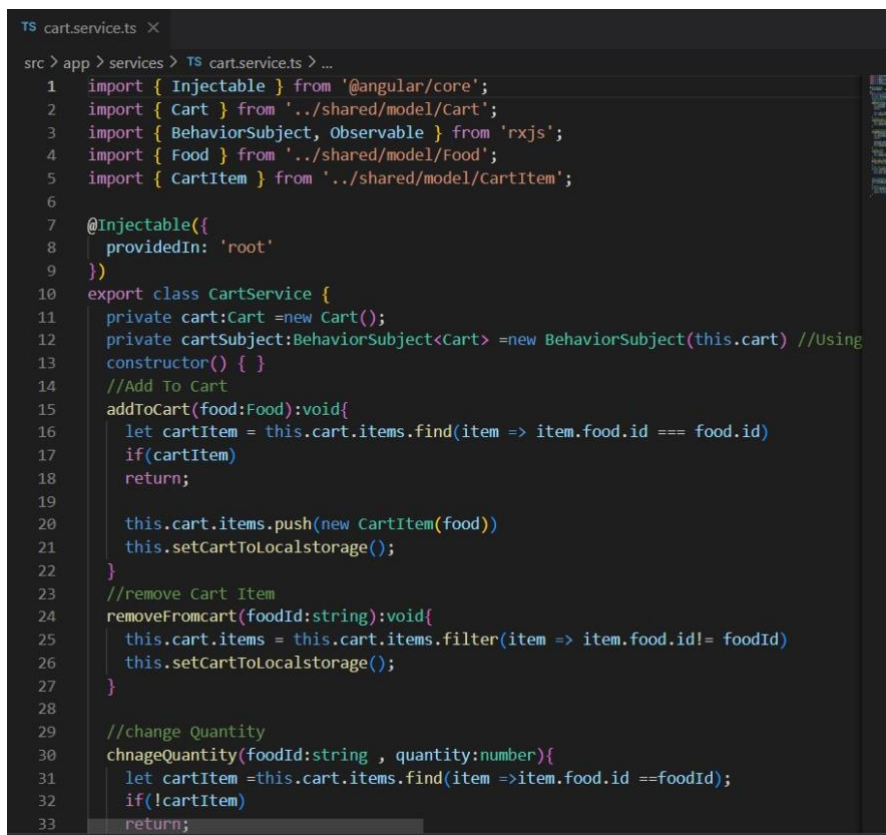
The clearCart function changes the cart in local storage, resets the Cart object to an empty cart, and emits the updated cart via the BehaviorSubject.

The function getCartFromLocalStorage retrieves the Cart object's JSON text representation from local storage and parses it to create a new Cart object. If there

are no cart records in the local storage, a new empty Cart object is returned. This function is used to initialise the Cart object in the CartService's constructor.

In order for other components to subscribe to it and get updates anytime there is a change to the cart, the getCartObservable function returns an observable of the Cart object.

Overall, the CartService is crucial to producing a faultless and user-friendly online application for the project's process of ordering meals from the Fresh Palette Cafe.



```
TS cart.service.ts X
src > app > services > TS cart.service.ts > ...
1  import { Injectable } from '@angular/core';
2  import { Cart } from '../shared/model/Cart';
3  import { BehaviorSubject, Observable } from 'rxjs';
4  import { Food } from '../shared/model/Food';
5  import { CartItem } from '../shared/model/CartItem';
6
7  @Injectable({
8    providedIn: 'root'
9  })
10 export class CartService {
11   private cart: Cart = new Cart();
12   private cartSubject: BehaviorSubject<Cart> = new BehaviorSubject(this.cart) //Using
13   constructor() { }
14   //Add To Cart
15   addToCart(food: Food): void {
16     let cartItem = this.cart.items.find(item => item.food.id === food.id)
17     if (cartItem)
18       return;
19
20     this.cart.items.push(new CartItem(food))
21     this.setCartToLocalStorage();
22   }
23   //remove Cart Item
24   removeFromCart(foodId: string): void {
25     this.cart.items = this.cart.items.filter(item => item.food.id !== foodId)
26     this.setCartToLocalStorage();
27   }
28
29   //change Quantity
30   changeQuantity(foodId: string, quantity: number) {
31     let cartItem = this.cart.items.find(item => item.food.id === foodId);
32     if (!cartItem)
33       return;
```

Figure 4.2.2.3 a

```

src > app > services > TS cart.service.ts > ...
34
35     cartItem.quantity=quantity;
36     cartItem.price = quantity*cartItem.food.price;
37     this.setCartToLocalStorage();
38 }
39 //Clear Cart
40 clearCart(){
41     this.cart= new Cart();
42     this.setCartToLocalStorage();
43 }
44 //get Cart Observable means Check Observable data
45 getCartObservable():Observable<Cart>{
46     return this.cartSubject.asObservable();
47 }
48 //now set local storage data
49 private setCartToLocalStorage():void{
50     this.cart.totalPrice = this.cart.items.reduce((prevSum,currentItem)
51     this.cart.totalcount=this.cart.items.reduce((prevSum , currentItem)
52
53
54     const cartJson =JSON.stringify(this.cart);
55     localStorage.setItem('Cart',cartJson);
56     this.cartSubject.next(this.cart)
57 }
58 //Whenever set Local Storage data then also get data
59 private getCartFromLocalStorage():Cart{
60     const cartJson =localStorage.getItem('Cart');
61     return cartJson?JSON.parse(cartJson):new Cart();
62 }
63 }
64

```

**Figure a-b:** Cart Service- All the functionality of cart to add item, to remove item , increase quantity , reduce quantity , clear cart .

## Data Base

### User Data Base :

```

TS datats
src > TS datats > 100 sample_users
90 },
91 ],
92
93 export const sample_users: any[] = [
94 {
95     name: "Atishya Jain",
96     email: "atishya123@gmail.com",
97     password: "atishya123",
98     address: "Roorkee",
99     isAdmin: true,
100 },
101 },
102 },
103 },
104 {
105     name: "Harsh Srivastav",
106     email: "harsh123@gmail.com",
107     password: "harsh23",
108     address: "lucknow",
109     isAdmin: true,
110 },
111 },
112 {
113     name: "Navya Yadav",
114     email: "gr1m69@gmail.com",
115     password: "nav02",
116     address: "lucknow",
117     isAdmin: true,
118 },
119 {
120     name: "Narendar Verma",
121     email: "Modiji123@gmail.com",
122     password: "Modi",
123     address: "lucknow",
124     isAdmin: true,
125 },
126 {
127     name: "Shubham Singh",
128     email: "csksubham@gmail.com",
129     password: "ladak101",
130     address: "bihar",
131     isAdmin: true,
132 },
133 },
134 {
135     name: "Kaushik Deka",
136     email: "cr7deka@gmail.com",
137     password: "chinki",
138     address: "Assam",
139     isAdmin: true,
140 },
141 ],
142

```

User data stored which is checked on time of login

### Food Item Data Base:



```

src > TS datas > [0] sample_foods
1
2 export const sample_foods: any[] = [
3
4   {
5     id: '1',
6     name: 'Pizza Pepperoni',
7     cookTime: '10-20',
8     price: 10,
9     favorite: false,
10    origins: ['ITALY'],
11    stars: 4.5,
12    imageUrl: 'assets/food-1.jpg',
13    tags: ['FastFood', 'Pizza', 'Lunch'],
14  },
15  {
16    id: '2',
17    name: 'Meatball',
18    price: 20,
19    cookTime: '20-30',
20    favorite: true,
21    origins: ['PERSIA', 'MIDDLE EAST', 'CHINA'],
22    stars: 4.7,
23    imageUrl: 'assets/food-2.jpg',
24    tags: ['SlowFood', 'Lunch'],
25  },
26  {
27    id: '3',
28    name: 'Hamburger',
29    price: 5,
30    cookTime: '10-15',
31    favorite: false,
32    origins: ['GERMANY', 'US'],
33    stars: 3.5,
34    imageUrl: 'assets/food-3.jpg',
35    tags: ['FastFood', 'Hamburger'],
36  },
37  {
38    id: '4',
39    name: 'Fried Potatoes',
40    price: 2,
41    cookTime: '15-20',
42    favorite: true,
43    origins: ['BELGIUM', 'FRANCE'],
44    stars: 3.3,
45    imageUrl: 'assets/food-4.jpg',
46    tags: ['FastFood', 'Fry'],
47  },
48  {
49    id: '5',
50    name: 'Chicken Soup',
51    price: 11,
52    cookTime: '40-50',
53    favorite: false,
54    origins: ['INDIA', 'ASIA'],
55    stars: 3.0,
56    imageUrl: 'assets/food-5.jpg',
57    tags: ['SlowFood', 'Soup'],
58  },
59  {
60    id: '6',
61    name: 'Vegetables Pizza',
62    price: 9,
63    cookTime: '40-50',
64    favorite: false,
65    origins: ['ITALY'],
66    stars: 4.0,
67    imageUrl: 'assets/food-6.jpg',
68    tags: ['FastFood', 'Pizza', 'Lunch'],
69  },
70  {
71    id: '7',
72    name: 'Spicy Cheese Burger',
73    price: 12,
74    cookTime: '20-30',

```

**Figure :** All details of Food Items that are being displayed on the home screen.

## Backend Server

```

TS server.ts
src > TS server.ts > ...
1 import express from "express";
2 import cors from "cors";
3 import { sample_foods , sample_users} from "../data";
4 import jwt from "jsonwebtoken";
5 const bodyParser = require('body-parser').json();
6
7 const app=express();
8 app.use(bodyParser)
9 app.use(express.json())
10 app.use(cors({
11   credentials:true,
12   origin:["http://localhost:4200"]
13 }));
14
15 app.get("/api/foods", ( req,res)=>{
16   res.send(sample_foods)
17 })
18
19 //get food via serchTerm
20 app.get("/api/foods/search/:searchTerm",(req,res)=>{
21   // console.log("req from serachTerm",req.params.searchTerm)
22   const searchTerm = req.params.searchTerm;
23   const foods= sample_foods.filter((food)=>food.name.toLowerCase().includes(searchTerm));
24   res.send(foods);
25 })
26
27 //get food by ID
28 app.get("/api/foods/:foodId",(req,res)=>{
29   const foodId=req.params.foodId;
30   const food=sample_foods.find((food)=> food.id==foodId)
31   res.send(food);
32 })
33
34 //LOGIN API
35 app.post("/api/users/login",(req, res)=>{
36   // console.log("REQ from login ", req.body);
37   const {email , password} = req.body;
38   const user = sample_users.find((user) => user.email === email && user.password === password);

```



```

TS server.ts
src > TS server.ts > ...
27 app.get("/api/foods/:foodId", (req, res) => {
28   const foodId = req.params.foodId;
29   const food = sample_foods.find((food) => food.id === foodId);
30   res.send(food);
31 })
32
33 // LOGIN API
34 app.post("/api/users/login", (req, res) => {
35   // console.log("REQ from login ", req.body);
36   const {email, password} = req.body;
37   const user = sample_users.find((user) => user.email === email && user.password === password);
38   if(user){
39     res.send(generateTokenResponse(user));
40   }
41   else{
42     res.status(400).json({message: "User Name or Password is incorrect"});
43   }
44 }
45 )
46
47 const generateTokenResponse = (user: any) => {
48   const token = jwt.sign({
49     email: user.email, isAdmin: user.isAdmin
50   }, "A3002", {
51     expiresIn: "30d"
52   });
53   user.token = token;
54   return user;
55 }
56
57 const port = 5000;
58 app.listen(port, () => {
59   console.log(`Website is Running on http://localhost:${port}`);
60 })

```

**Figure a-b:** All API and backend setup .

## App Module

This is the AppModule file for the web application to order food from fresh palette cafe website's Angular project. It has all the settings, imports, and declarations required for the programme to function.

All of the application's components, such as HeaderComponent, HomeComponent, SearchComponent, FoodPageComponent, CartPageComponent, TitleComponent, and

LoginComponent, are listed in the declarations section.

The BrowserModule, which enables the application to run in the browser, the AppRoutingModule for routing, the RatingModule for the star rating feature, the HttpClientModule for sending HTTP requests, the ReactiveFormsModule for using reactive forms, the ToastrModule for displaying toast messages, and the BrowserAnimationsModule for animation are all listed in the imports section.

The bootstrap section identifies the application's primary component, in this instance AppComponent.

This is the web application to order food from fresh palette cafe website project's Angular app module file (app.module.ts). It specifies and imports several modules and project-related components.

Declarations for each component and module are included in the declarations and imports arrays, respectively. Use the providers array to add providers for dependency injection. In this case, we haven't created any special suppliers.

The bootstrap array is then used to bootstrap the AppComponent as the application's root component.

The Angular application's fundamental functionality is provided by the BrowserModule and HttpClientModule modules, while the RatingModule module is utilised to provide ratings.

For developing reactive forms that are utilised in the login component for user authentication, the ReactiveFormsModule module is imported.

Use the ToastrModule to show alerts on the screen in response to user input or server-side events.

The AppComponent is the primary component of the application, and there are other defined components for the header, home, search, food page, cart page, title, and login.

Overall, this file sets the program's core structure together with all necessary dependencies and modules. It is a significant file that is necessary for the efficient running of the programme.

```

src > app > TS app.module.ts > TS AppModule
7  import { AppComponent } from './app.component';
8  import { HeaderComponent } from './components/header/header.component';
9  import { HomeComponent } from './components/pages/home/home.component';
10 import { SearchComponent } from './components/pages/search/search.component';
11 import { FoodPageComponent } from './components/pages/food-page/food-page.component';
12 import { CartPageComponent } from './components/pages/cart-page/cart-page.component';
13 import { TitleComponent } from './components/pages/title/title.component';
14 import { LoginComponent } from './components/pages/login/login.component';
15 import { ReactiveFormsModule } from '@angular/forms';
16 import { ToastrModule } from 'ngx-toastr';
17 import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
18
19 @NgModule({
20   declarations: [
21     AppComponent,
22     HeaderComponent,
23     HomeComponent,
24     SearchComponent,
25     FoodPageComponent,
26     CartPageComponent,
27     TitleComponent,
28     LoginComponent
29   ],
30   imports: [
31     BrowserModule,
32     BrowserAnimationsModule,
33     AppRoutingModule,
34     RatingModule

```

**Figure :** App Module showing all the components created

## Angular Project Framework and Configuration

The build and development parameters for an Angular application are defined and customised in the angular.json configuration file. An Angular project's root directory has the file, which provides details about the project's dependencies, settings, and organisational structure.

The parts of the angular.json file are as follows:

- version: Identifies the configuration file's version.

The root directory for new projects is defined by the newProjectRoot setting.

Projects: This section includes the workspace's setup for each project.

- Schematics: Provides settings for the code snippet generators known as schematics.
- cli: the Angular CLI's options directory.

The name, root folder, source root, and prefix are just a few of the features that make up the project section. For the project's numerous build and development activities, including constructing, testing, and serving, the architect property includes configurations.

The output path, index file, main file, and assets are just a few examples of the settings that the build configuration offers. The styles field defines the global styles

to incorporate into the build, including the primary scss file and any extra styles from external dependencies. The scripts field defines any JavaScript files to include in the build..

The configurations property lists many build configurations, including "production" and "development" configurations. Options for each configuration may include source maps, build output budget sizes, and build optimisations.

The build configuration to use is one of the choices specified in the serve configuration. To extract translations from the project for localization, use the extract-i18n option.

The test configuration details the assets, styles, scripts, and polyfills that need be present in the test environment in order to execute the tests.

Finally, Angular schematics—code generators used to produce files like components, services, and directives—have choices in the schematics section. In conclusion, the angular.json file, which specifies the project structure, dependencies, and many build and development options, is a crucial configuration file for an Angular project.

## Events List

Every web application needs events because they allow for dynamic and interesting user experiences. Our online application for ordering food from the fresh palette cafe website may utilise events to launch activities, such as updating the shopping cart, displaying alerts when an order is successfully placed, or tracking user activity for analytics. In this section, we'll go through the various events used in our application and how Angular handles them.

1. **User Login Event:** A user's login to the website initiates this event. It is possible to track user behaviour and tailor their online experience with it.
2. **User Signup Event:** A new user's registration on the website causes this event to occur. It may be utilised to keep track of the quantity of signups as well as other pertinent details like their preferences and location.

3. **Item Added to Cart Event:** This event is launched when a user adds an item to their online Food ordering cart. It has the ability to gauge consumer interest in a certain product and recommend comparable items to them.
4. **Order Placed Event:** Each time someone makes an order through our website it sparks something called an Order Placed Event that provides us with valuable data such as what was bought how much of it was ordered and where our customers are located.
5. **Item Removed from Cart Event:** Should any customer decide to remove something from their virtual cart before completing checkout we'll receive notification through something called an Item Removed from Cart Event – enabling us to better cater to each individuals preferences moving forward..
6. **Order Delivered Event:** As soon as we've successfully delivered someones order we're alerted by means of an Order Delivered Event which tracks delivery progress and ensures accuracy on our part when fulfilling requests..
7. **Order Cancelled Event:** Anyone who cancels their online purchase will invoke whats known as an Order Cancelled Event – allowing us to understand why this may have occurred while simultaneously keeping track of how often cancellations happen overall.
8. **Search Event:** Finally whenever someone uses the search bar on our website it triggers a Search Event thats incredibly useful in monitoring interest levels for various items and finding ways to improve our online search capabilities and overall user experience
9. **Delivered Order:** This event is started when an order is delivered to the user..It might be applied to track the status of delivery and confirm that the correct order was sent.
10. **Payment Processed Event:** This event is started when a user's payment is processed on the website. It may be used to monitor both the quantity of

payments that have been received successfully and any possible issues with payments.

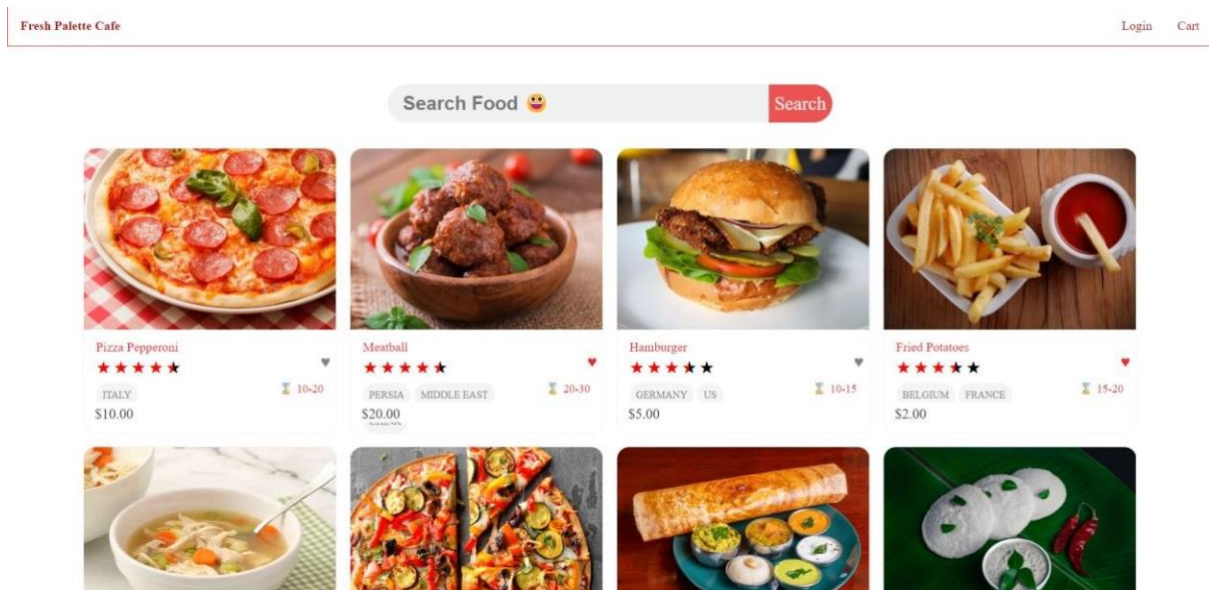
**11. Review Submitted Event:** When a person reviews a product or service and submits it on the website, this happens. It may be utilised to keep an eye on customer feedback and improve the level of the services the website offers.

## 11.Screenshot and demo

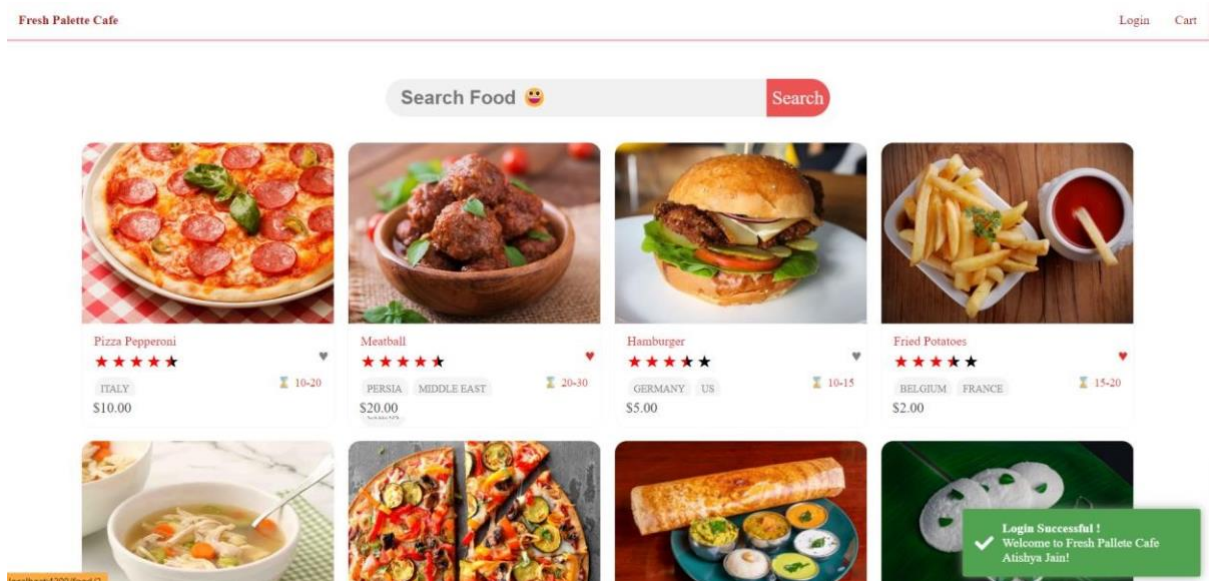
A comprehensive list of the test cases developed for the application must be included in this section. It should include both positive and negative test cases as well as cutting-edge situations that might jeopardise the system. Additionally, the expected outcomes for each test scenario must be stated.

### Positive Test Case:

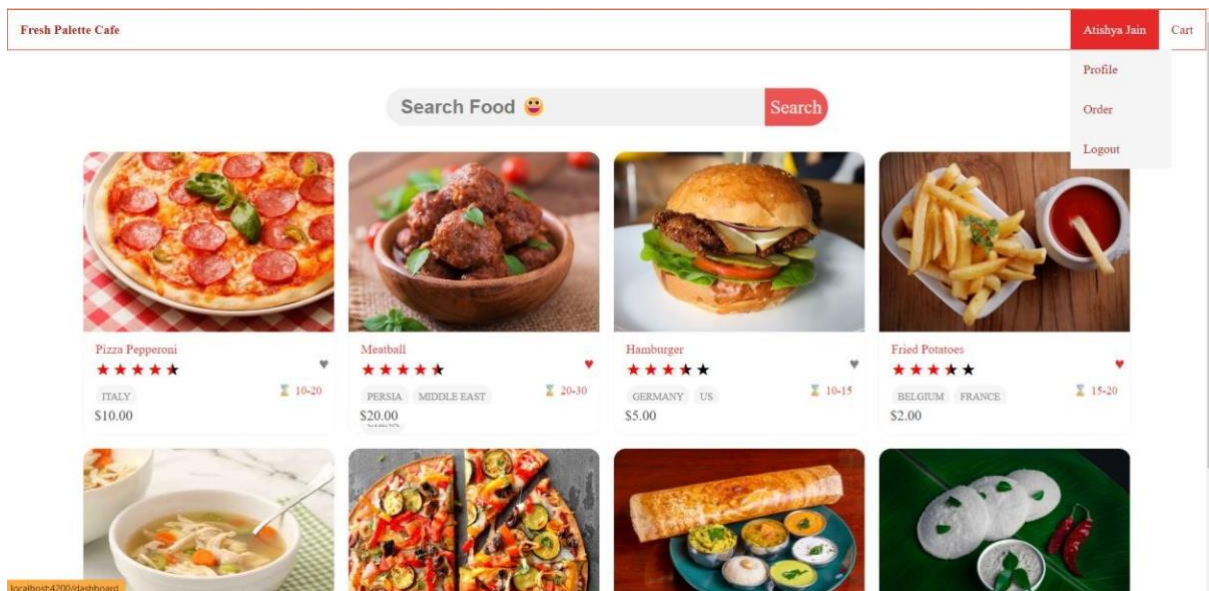
First will be showing the Page without Login. After that Screenshot of Successful login.



This is the Page when we open the Website . In the Header section it will be showing only the Login option and cart option.

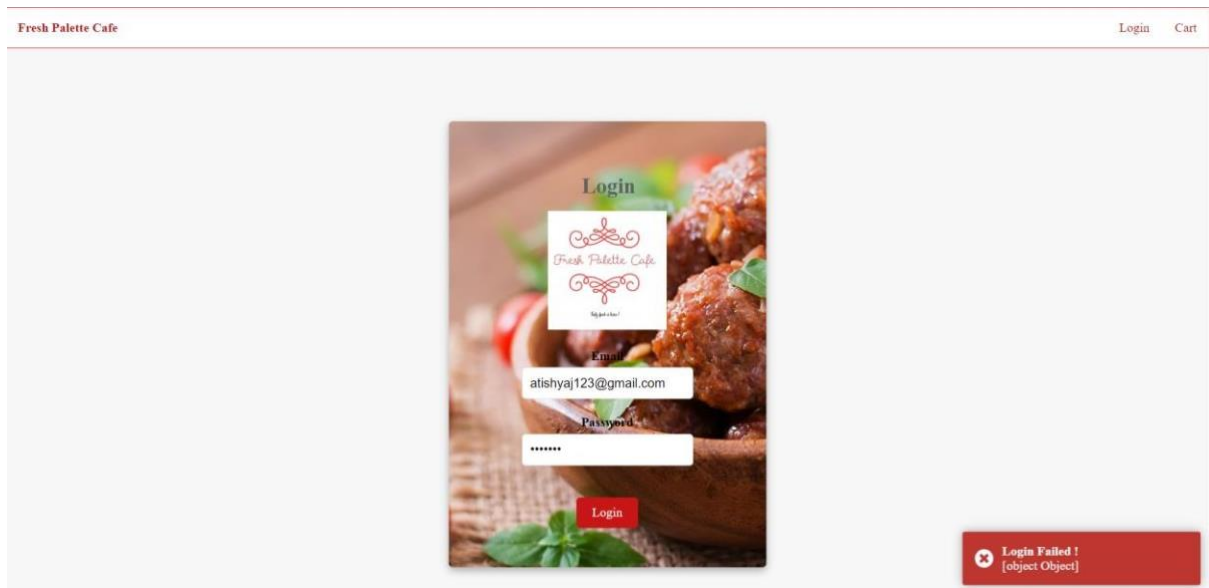


Shows the pop up of successful login

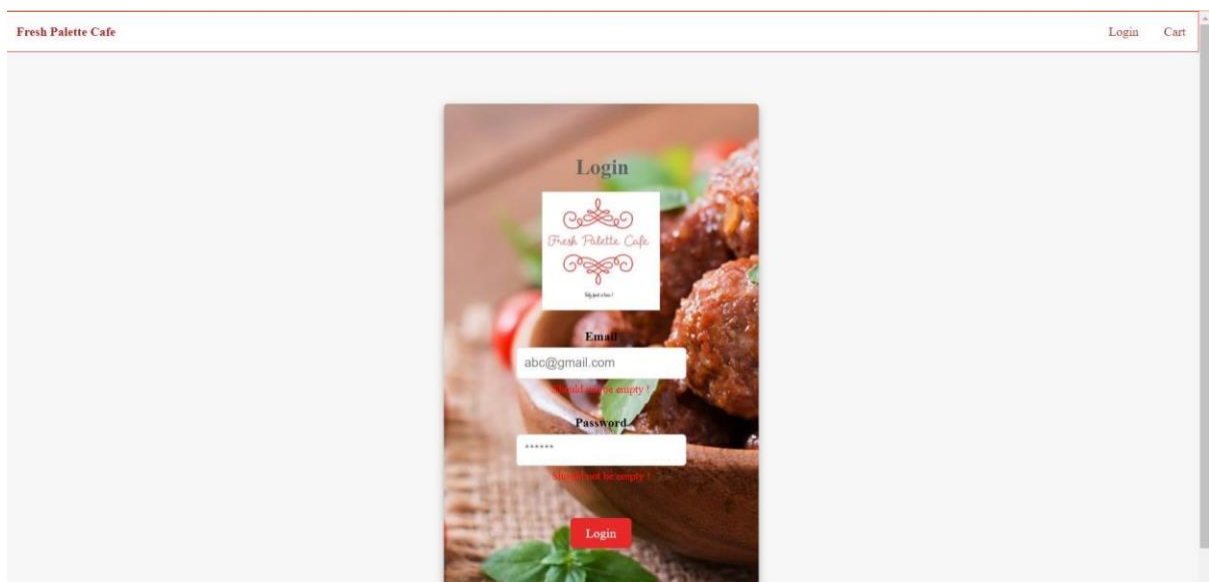


After successful Login in header it will show the user name and have a drop down option in which we will be option to visit profile , order details and to LogOut.





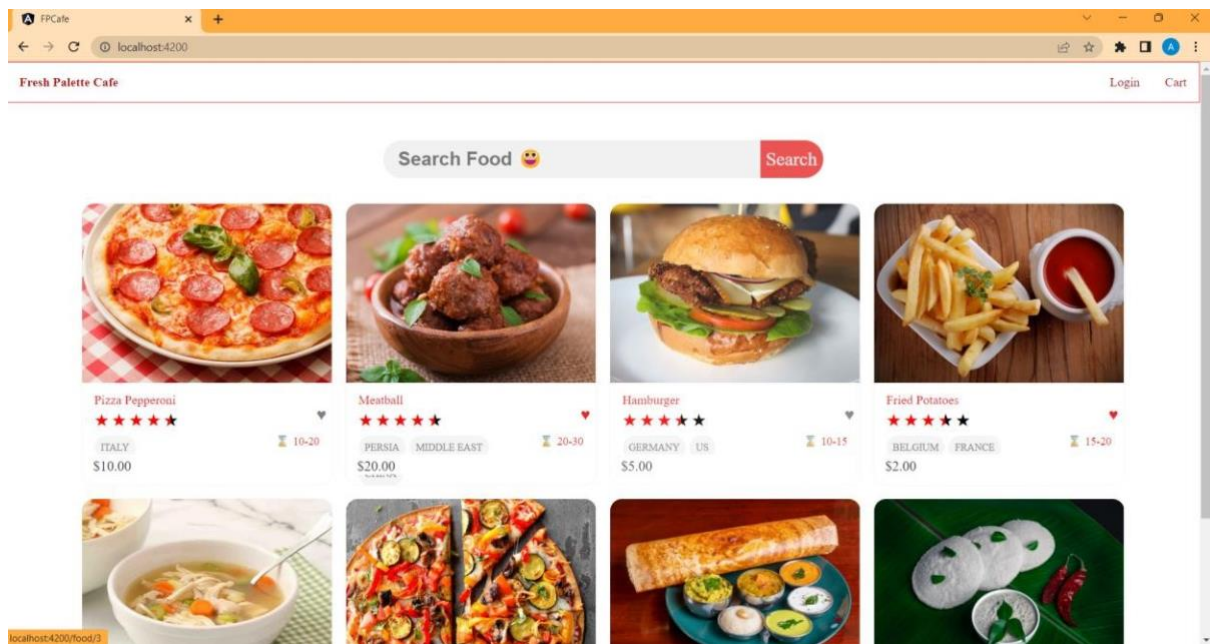
In this wrong details have be entered in the login therefore a popup of Login Failed! is shown.



Should not be Empty warning is shown to login if nothing is filled and login is clicked

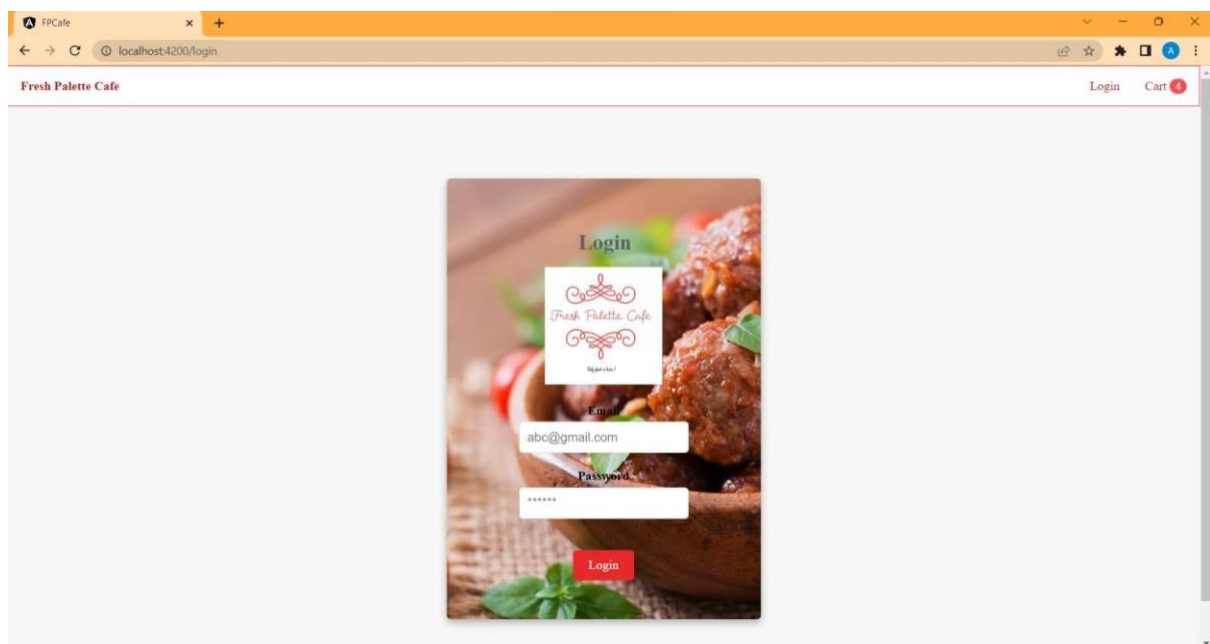
**Results:**  
**Home Page :**





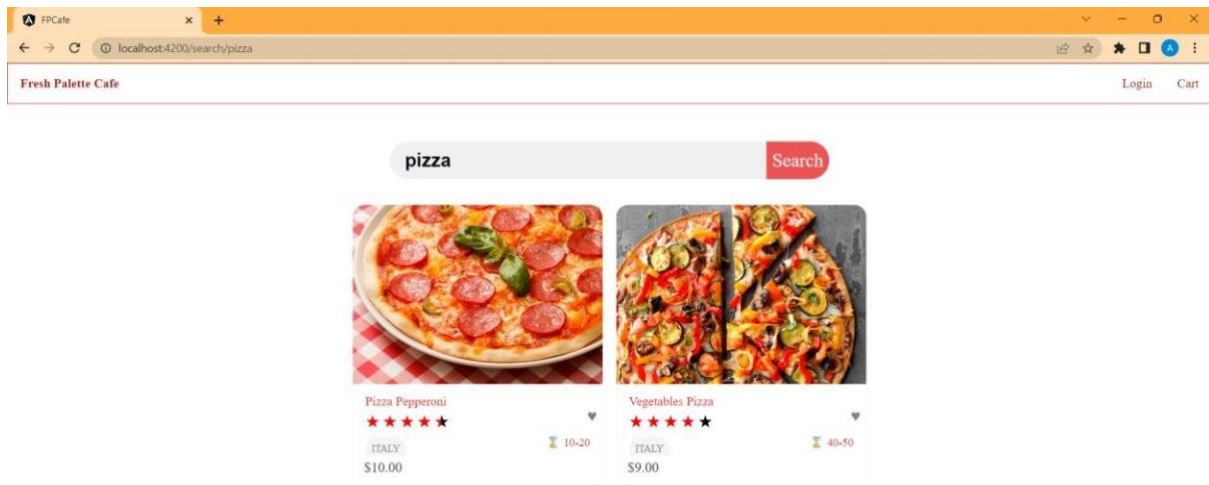
Home Page

## Login Page :



## Login Page

### Search :



## 12. Known issues

### Conclusion :

We initially determined the functional and non-functional requirements for the website before building the crucial features and functionalities to meet those objectives. Some of these features include a user-friendly interface for browsing restaurants and menu items, a shopping cart for adding and managing products for orders, an order history section for looking over past orders, and a restaurant administration system for controlling the menu and orders. We also set up a number of security precautions to ensure the protection of user data and transactions.

In conclusion, our proposal to create a website for ordering meals was successfully developed and evaluated. We utilised Angular as the main front-end framework throughout the project, coupled with a number of back-end tools like Node.js, Express, and MongoDB. The project's objectives included making it easier for people to purchase meals online and helping restaurant operators manage customer orders and menu items.

We utilised a range of dependency injection techniques, including as constructor injection, property injection, and method injection, to manage and deliver dependencies to the numerous components and services across the project. We also incorporated a number of events, including click events, form events, and lifecycle events, to increase interactivity and improve the website user experience.

We looked at a variety of performance indicators, including page load time, server response time, and database query time, and created performance graphs and charts to display the results. Based on our research, we identified areas that needed to be improved and provided suggestions for more work that might be done to improve the website's operation.

The Angular framework has shown to be effective for accelerating development and streamlining maintenance. The whole user experience has been enhanced by the addition of services including menu customization, user registration and authentication, restaurant management, and order tracking. The effort has also raised awareness of how important it is to manage and build databases effectively for data storage. Similar to every software project, there is always room for improvement. Performance improvement for this project could be advantageous, particularly for the queries the database and backend server make.

Overall, it was challenging but rewarding to design the website project for the web application to purchase food from Fresh Palette Cafe. We were able to provide a platform that is both functional and user-friendly for the web application to order food from Fresh Palette Cafe. This website should be beneficial to both users and restaurant owners, and we want to continue growing and improving this endeavour. Overall, the project to create an online web application to order food from fresh palette cafe website was successful in meeting its goals of offering a practical and user-friendly platform for customers to place orders and for restaurants to manage their orders.

Working with modern web development tools and methodologies has been a good learning experience thanks to the project to build an online web application to purchase meals from the fresh palette cafe website. Additionally, it has aided in the development of expertise in project management, database administration, frontend, and backend programming, and UI/UX design.

## 13.Future Enhancement

There is always space for progress and improvement in software projects. In this part, we will talk about the possible expansion of the online meal ordering website application's capability as well as any prospective improvements that may be done to improve the user experience. These upgrades can help the platform draw in more users and maintain its position as a leader in the rapidly developing online meal ordering market. Let us look more closely at the anticipated future developments below.

- **Mobile App Development:** It is possible to convert the online web application to order food from fresh palette cafe website into a mobile application, increasing its accessibility to a larger audience. Having a mobile app will be a fantastic method to reach out to more clients with the increasing number of smartphone users.
- **Integration with Third-party Payment Gateways:** Multiple payment channels, including PayPal, Stripe, and others, can be supported by the project in the future.  
This will provide customers more alternatives for paying for their products and streamline the ordering process..
- **Integration with Social Media Platforms:** It is possible to combine the project with social networking sites like Facebook, Twitter, and Instagram. Customers will be able to do this and share their experiences with their social media followers and friends, perhaps promoting the business..

- **Customization of User Interface:** It is possible to expand the project so that users may personalise the user interface to suit their tastes. This might involve things like altering the website's layout, text size, and colour palette.
- **Integration with Delivery Services:** Extensions to the project allow for integration with outside delivery services like UberEats, Grubhub, etc. Customers will have additional delivery alternatives as a result, and the ordering procedure will be easier.
- **Implementation of Advanced Analytics:** It is possible to expand the project to incorporate sophisticated analytics tools that will track website performance, reveal user preferences and behaviours, and offer new information on how to make changes in the future.
- **Integration with Chatbots:** It is possible to expand the project to incorporate chatbots that may assist clients with order placement, order tracking, and general customer service. This will lessen the workload of the customer care team and improve the client experience.
- **Implementation of Loyalty Programs:** The project may be expanded to incorporate consumer incentive schemes for making purchases, posting reviews, and introducing friends. Both retaining current clients and luring new ones will benefit from this.
- **Expansion to Multiple Locations:** Adding more sites to the project will help it reach a larger audience and grow its clientele. In addition to recruiting more people, this calls for the setup of new servers and databases.
- **Integration with AI and ML technologies:** It is possible to expand the project to incorporate AI and ML technologies, such as customised suggestions, predictive ordering, and automated customer assistance.

Customers will have a better customised experience as a result, and this will aid in streamlining corporate procedures..

These are only a few of the project's potential future scope areas. The precise areas to be addressed will rely on the objectives and needs of the organisation, as well as the money and resources at hand.

