KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE – 638 060

DEPARTMENT OF INFORMATION TECHNOLOGY

22ITL42 – WEB TECHNOLOGY LABORATORY

PROJECT REPORT

IYYAPPAN R – 23ITR063

ONLINE FOOD DELIVERY WEBSITE

OBJECTIVES:

Our online food delivery website aims to simplify the way people order food by providing a seamless and user-friendly experience. We connect users with a wide range of local restaurants, offering diverse cuisines to suit every taste. Customers can easily browse menus, place orders, and track their deliveries in real time. We focus on fast and reliable delivery, secure payment options, and accurate order fulfillment. Our platform is designed to make food ordering more convenient, efficient, and accessible. Customer satisfaction and high-quality service are at the core of our mission.

TECHNOLOGY STACK

Backend Development

- Programming Language: TypeScript
- Framework: Angular 16 (with Node.js/Express for API)
- Database: MongoDB (Non-relational)
- API Architecture: RESTful services

Frontend Development

- ❖ Core Technologies: HTML5, CSS3, SCSS
- ❖ UI Framework: Angular Material + Bootstrap 5
- ❖ State Management: NgRx (for complex state)
- ❖ Form Handling: Reactive Forms

MODULE DESCRIPTION:

- o Home page
- o Oder page
- o Dinning page
- o Login/Signup page

ADVANTAGES:

- ❖ Enhanced User Experience
- ❖ Secure & Reliable
- ❖ Cost-Effective & Fast Development
- ❖ Competitive Advantage

SOURCE CODE:

home.component.html:

```
<div class="home-container">
  <!-- Hero Banner -->
  <div class="hero-banner">
    <div class="hero-content">
      <h1>Delicious Food Delivered To Your Doorstep</h1>
      <div class="search-container">
        <input
          type="text"
          [(ngModel)]="searchQuery"
          (keyup.enter)="searchRestaurants()"
          placeholder="Search for restaurants or cuisines..."
          class="search-input"
        >
        <button (click)="searchRestaurants()" class="search-btn">
          <i class="fas fa-search"></i>
        </button>
      </div>
    </div>
  </div>

  <!-- Category Tabs -->
  <div class="category-tabs">
    <div class="container">
      <div class="tabs-scroll">
        <button
          *ngFor="let category of categories"
          [class.active]="activeCategory === category"
          (click)="filterByCategory(category)"
          class="tab-btn"
        >
          {{ category }}
        </button>
      </div>
    </div>
  </div>

  <!-- Featured Restaurants -->
  <section class="featured-section" *ngIf="featuredRestaurants.length > 0">
    <div class="container">
      <h2 class="section-title">Featured Restaurants</h2>
      <div class="row">
        <div
        *ngFor="let restaurant of featuredRestaurants"
        class="col-md-4 mb-4"
      >
        <div class="restaurant-card featured">
          <div class="card-img">
            <img [src]="restaurant.image" [alt]="restaurant.name">
            <div class="rating-badge">
              <i class="fas fa-star"></i> {{ restaurant.rating }}
            </div>
          </div>
          <div class="card-body">
            <h3>{{ restaurant.name }}</h3>
            <div class="cuisine">{{ restaurant.cuisine }}</div>
            <div class="details">
              <span class="delivery-time">
                <i class="fas fa-clock"></i> {{ restaurant.deliveryTime }}
              </span>
              <span class="min-order">
                <i class="fas fa-shopping-bag"></i> ${{ restaurant.minOrder }} min
              </span>
            </div>
            <button
              (click)="viewRestaurant(restaurant._id)"
              class="btn view-btn"
            >
              View Menu
            </button>
          </div>
        </div>
      </div>
    </div>
  </div>
```

```
</section>

<!-- All Restaurants -->
<section class="restaurants-section">
  <div class="container">
    <div class="section-header">
      <h2 class="section-title">All Restaurants</h2>
      <div class="sort-options">
        <select [(ngModel)]="sortOption" (change)="sortRestaurants()" class="form-select">
          <option value="rating-desc">Rating: High to Low</option>
          <option value="delivery-asc">Delivery Time: Fastest</option>
          <option value="min-order-asc">Min. Order: Low to High</option>
        </select>
      </div>
    </div>

    <div *ngIf="isLoading" class="loading-spinner">
      <div class="spinner-border text-primary" role="status">
        <span class="visually-hidden">Loading...</span>
      </div>
    </div>

    <div *ngIf="!isLoading && filteredRestaurants.length === 0" class="no-results">
      <i class="fas fa-utensils-slash"></i>
      <h3>No restaurants found</h3>
      <p>Try adjusting your search or filters</p>
    </div>

    <div class="row">
      <div
        *ngFor="let restaurant of filteredRestaurants"
        class="col-md-6 col-lg-4 mb-4"
  >
   <div class="restaurant-card">
    <div class="card-img">
     <img [src]="restaurant.image" [alt]="restaurant.name">
     <div class="rating-badge">
      <i class="fas fa-star"></i> {{ restaurant.rating }}
     </div>
    </div>
    <div class="card-body">
     <h3>{{ restaurant.name }}</h3>
     <div class="cuisine">{{ restaurant.cuisine }}</div>
     <div class="details">
      <span class="delivery-time">
       <i class="fas fa-clock"></i> {{ restaurant.deliveryTime }}
      </span>
      <span class="min-order">
       <i class="fas fa-shopping-bag"></i> ${{ restaurant.minOrder }} min
      </span>
     </div>
     <button
      (click)="viewRestaurant(restaurant._id)"
      class="btn view-btn"
     >
      View Menu
     </button>
    </div>
   </div>
  </div>
 </div>

 <!-- Pagination -->
 <div *ngIf="totalPages > 1" class="pagination-container">
```

```
        <nav aria-label="Restaurant pagination">
          <ul class="pagination">
            <li class="page-item" [class.disabled]="currentPage === 1">
              <a class="page-link" (click)="changePage(currentPage - 1)">Previous</a>
            </li>

            <li
              *ngFor="let page of getPageNumbers()"
              class="page-item"
              [class.active]="page === currentPage"
            >
              <a class="page-link" (click)="changePage(page)">{{ page }}</a>
            </li>

            <li class="page-item" [class.disabled]="currentPage === totalPages">
              <a class="page-link" (click)="changePage(currentPage + 1)">Next</a>
            </li>
          </ul>
        </nav>
      </div>
    </div>
</section>

<!-- Promo Banner -->
<div class="promo-banner">
  <div class="container">
    <div class="promo-content">
      <h2>Get 20% Off Your First Order!</h2>
      <p>Use code <strong>FOODIE20</strong> at checkout</p>
      <button class="btn order-now-btn">Order Now</button>
    </div>
  </div>
  </div>
</div>
```

home.component.css:

```
.home-container {
  font-family: 'Poppins', sans-serif;

  .hero-banner {
    background: linear-gradient(rgba(0,0,0,0.5), url('/assets/images/hero-bg.jpg');
    background-size: cover;
    background-position: center;
    color: white;
    padding: 5rem 0;
    text-align: center;

    .hero-content {
      max-width: 800px;
      margin: 0 auto;

      h1 {
        font-size: 2.5rem;
        margin-bottom: 1.5rem;
      }
    }
  }

  .restaurant-card {
    border-radius: 10px;
    overflow: hidden;
    box-shadow: 0 3px 10px rgba(0,0,0,0.1);
    transition: transform 0.3s ease;
    height: 100%;

    &:hover {
      transform: translateY(-5px);
    }

    .card-img {
      position: relative;
      height: 180px;
      overflow: hidden;

      img {
        width: 100%;
        height: 100%;
        object-fit: cover;
      }
    }
  }

  /* Add more styles for other elements */
}
```

auth.service.ts:

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { BehaviorSubject, Observable, tap } from 'rxjs';
import { Router } from '@angular/router';

@Injectable({
  providedIn: 'root'
})
```

```
export class AuthService {
  private apiUrl = 'http://localhost:3000/api/auth';
  private currentUserSubject = new BehaviorSubject<any>(null);
  public currentUser = this.currentUserSubject.asObservable();

  constructor(private http: HttpClient, private router: Router) {
    const user = localStorage.getItem('currentUser');
    if (user) {
      this.currentUserSubject.next(JSON.parse(user));
    }
  }

  login(email: string, password: string): Observable<any> {
    return this.http.post(`${this.apiUrl}/login`, { email, password }).pipe(
      tap((response: any) => {
        if (response.token && response.user) {
          localStorage.setItem('currentUser', JSON.stringify(response.user));
          localStorage.setItem('token', response.token);
          this.currentUserSubject.next(response.user);
        }
      })
    );
  }

  signup(userData: any): Observable<any> {
    return this.http.post(`${this.apiUrl}/signup`, userData);
  }

  logout() {
    localStorage.removeItem('currentUser');
    localStorage.removeItem('token');
    this.currentUserSubject.next(null);
    this.router.navigate(['/login']);
  }

  getToken(): string | null {
    return localStorage.getItem('token');
  }

  isAuthenticated(): boolean {
    return !!this.getToken();
  }
}
```

auth.interceptor.ts:

```
import { Injectable } from '@angular/core';
import {
  HttpRequest,
  HttpHandler,
  HttpEvent,
  HttpInterceptor
} from '@angular/common/http';
import { Observable } from 'rxjs';
import { AuthService } from './auth.service';

@Injectable()
export class AuthInterceptor implements HttpInterceptor {
  constructor(private authService: AuthService) {}

  intercept(request: HttpRequest<unknown>, next: HttpHandler):
Observable<HttpEvent<unknown>> {
    const token = this.authService.getToken();
    if (token) {
      request = request.clone({
        setHeaders: {
          Authorization: `Bearer ${token}`
        }
      });
    }
    return next.handle(request);
  }
}
```

restaurant.service.ts:

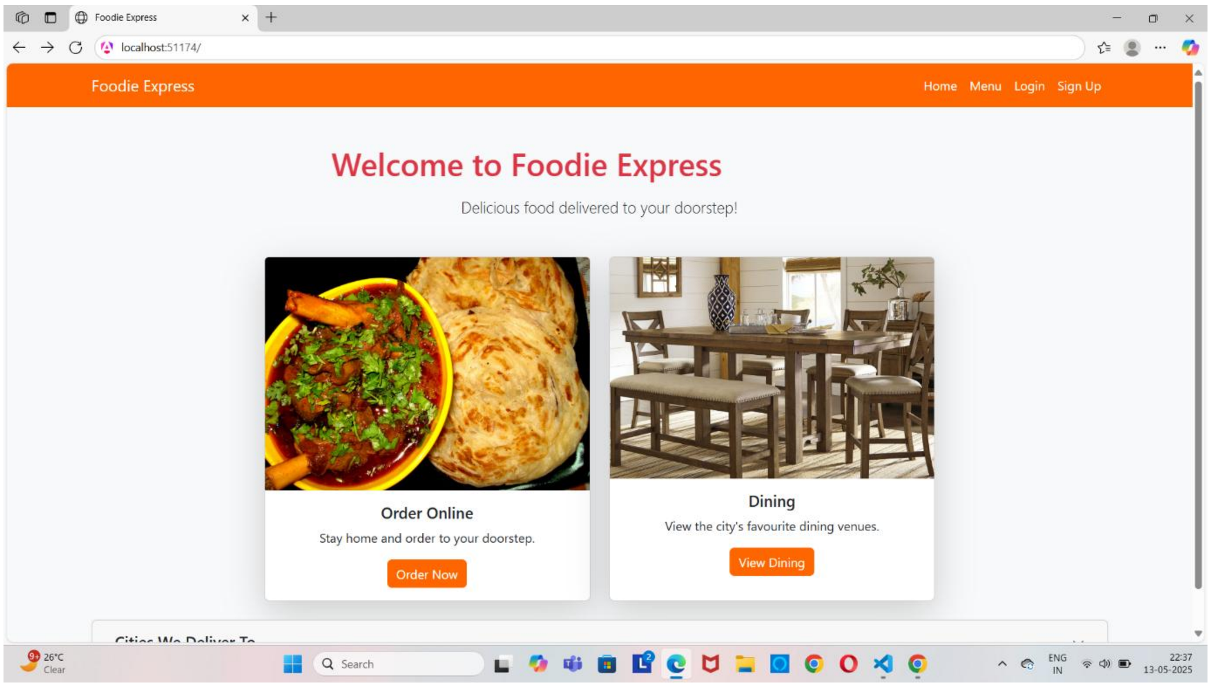
```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable, BehaviorSubject } from 'rxjs';
import { map } from 'rxjs/operators';

interface Restaurant {
  _id: string;
  name: string;
  cuisine: string;
  deliveryTime: string;
  minOrder: number;
  rating: number;
  image: string;
}

interface MenuItem {
  _id: string;
  name: string;
  description: string;
  price: number;
  category: string;
  restaurantId: string;
  image: string;
}

@Injectable({
  providedIn: 'root'
})
export class RestaurantService {
  private apiUrl = 'http://localhost:3000/api/restaurants';
  private restaurantsSubject = new BehaviorSubject<Restaurant[]>([]);
  public restaurants$ = this.restaurantsSubject.asObservable();

  constructor(private http: HttpClient) {
    this.loadRestaurants();
  }

  private loadRestaurants(): void {
    this.http.get<Restaurant[]>(this.apiUrl).subscribe({
      next: (restaurants) => this.restaurantsSubject.next(restaurants),
      error: (err) => console.error('Failed to load restaurants:', err)
    });
  }

  getRestaurantById(id: string): Observable<Restaurant> {
    return this.http.get<Restaurant>(`${this.apiUrl}/${id}`);
  }

  getMenu(restaurantId: string): Observable<MenuItem[]> {
    return this.http.get<MenuItem[]>(`${this.apiUrl}/${restaurantId}/menu`);
  }

  searchRestaurants(query: string): Observable<Restaurant[]> {
    return this.http.get<Restaurant[]>(`${this.apiUrl}/search?q=${query}`);
  }
}
```

cart.service.ts:

```
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';

interface CartItem {
  _id: string;
  name: string;
  price: number;
  quantity: number;
  restaurantId: string;
  image: string;
}

@Injectable({
  providedIn: 'root'
})
export class CartService {
  private cartItemsSubject = new BehaviorSubject<CartItem[]>([]);
  public cartItems$ = this.cartItemsSubject.asObservable();
  private cartTotalSubject = new BehaviorSubject<number>(0);
  public cartTotal$ = this.cartTotalSubject.asObservable();

  constructor() {
```

```
  const savedCart = localStorage.getItem('cart');
  if (savedCart) {
    this.cartItemsSubject.next(JSON.parse(savedCart));
    this.updateTotal();
  }
}

addToCart(item: CartItem): void {
  const currentItems = this.cartItemsSubject.value;
  const existingItem = currentItems.find(i => i._id === item._id);

  if (existingItem) {
    existingItem.quantity += 1;
  } else {
    currentItems.push({ ...item, quantity: 1 });
  }

  this.cartItemsSubject.next(currentItems);
  this.updateTotal();
  this.saveCart();
}

removeFromCart(itemId: string): void {
  const updatedItems = this.cartItemsSubject.value.filter(item => item._id !== itemId);
  this.cartItemsSubject.next(updatedItems);
  this.updateTotal();
  this.saveCart();
}

updateQuantity(itemId: string, quantity: number): void {
    const updatedItems = this.cartItemsSubject.value.map(item => {
      if (item._id === itemId) {
        return { ...item, quantity };
      }
      return item;
    });
    this.cartItemsSubject.next(updatedItems);
    this.updateTotal();
    this.saveCart();
  }

  clearCart(): void {
    this.cartItemsSubject.next([]);
    this.cartTotalSubject.next(0);
    localStorage.removeItem('cart');
  }

  private updateTotal(): void {
    const total = this.cartItemsSubject.value.reduce(
      (sum, item) => sum + (item.price * item.quantity),
      0
    );
    this.cartTotalSubject.next(total);
  }

  private saveCart(): void {
    localStorage.setItem('cart', JSON.stringify(this.cartItemsSubject.value));
  }
}
```

OUTPUT:



Welcome to Foodie Express

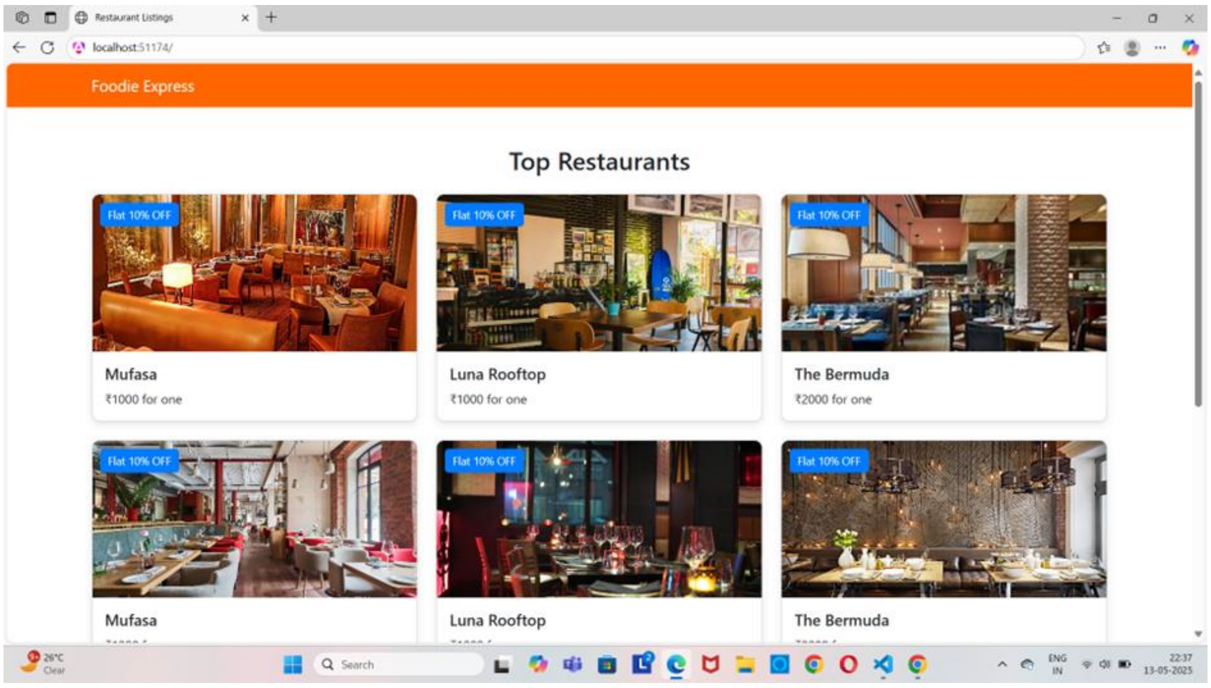
Delicious food delivered to your doorstep!

Order Online

Stay home and order to your doorstep.

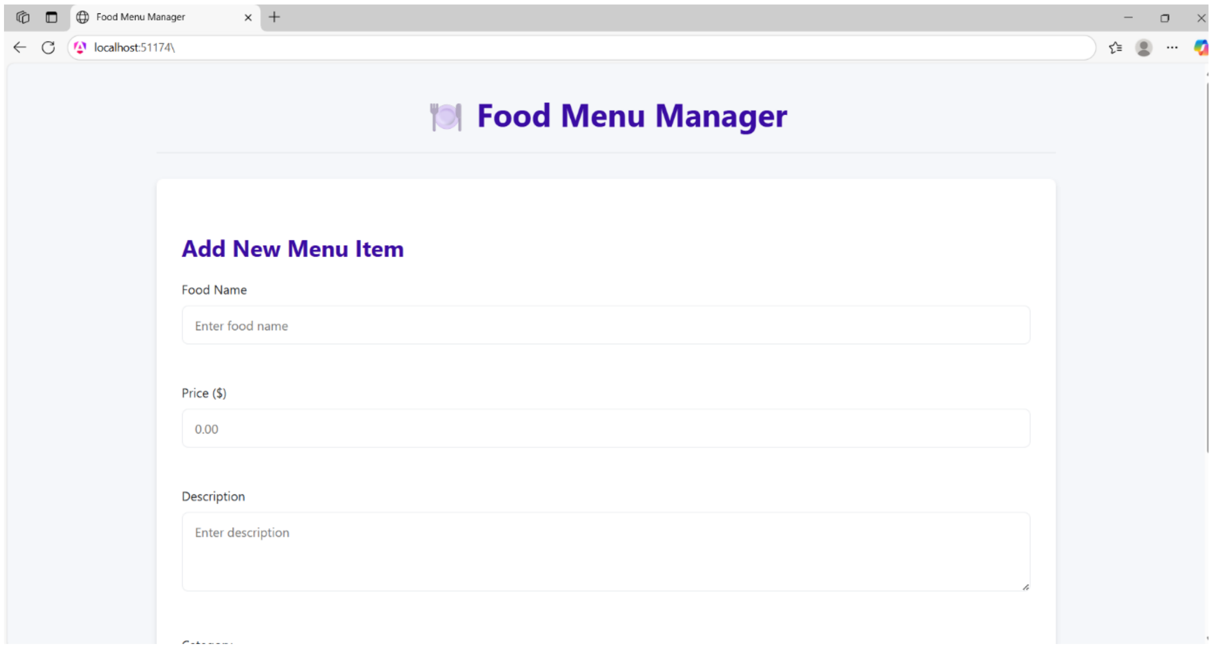
Dining

View the city's favourite dining venues.



Top Restaurants

Flat 10% OFF	Flat 10% OFF	Flat 10% OFF
Mufasa	Luna Rooftop	The Bermuda
₹1000 for one	₹1000 for one	₹2000 for one
Flat 10% OFF	Flat 10% OFF	Flat 10% OFF
Mufasa	Luna Rooftop	The Bermuda



Food Menu Manager

Add New Menu Item

Food Name

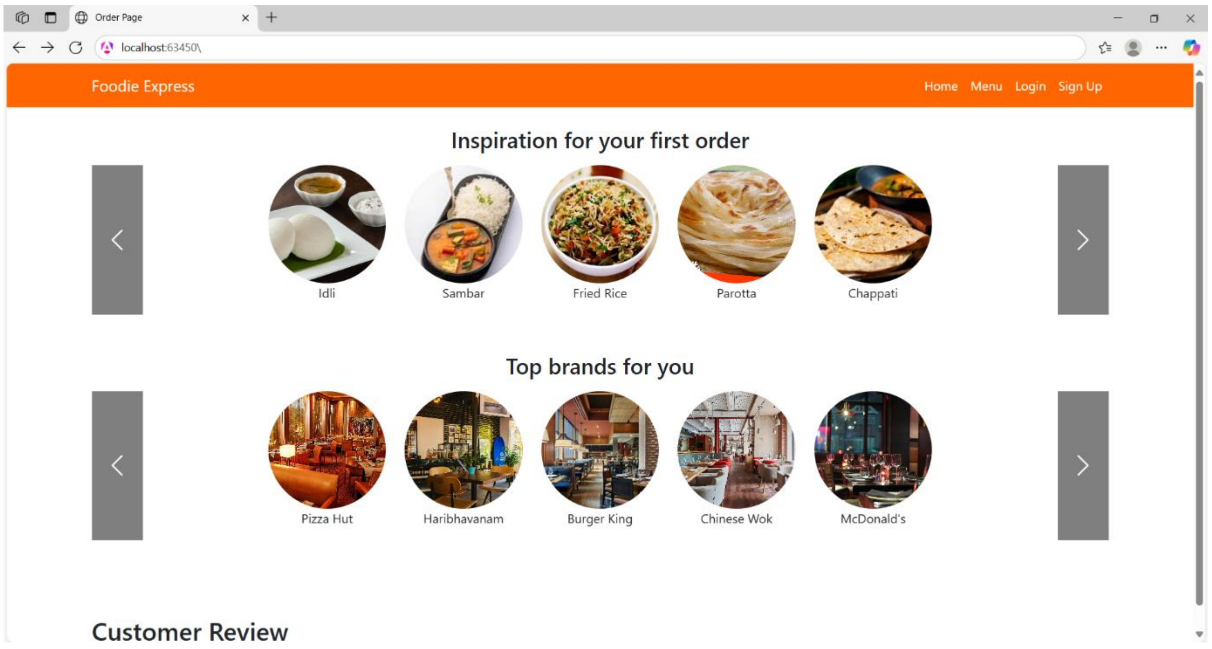
Enter food name

Price ($)

0.00

Description

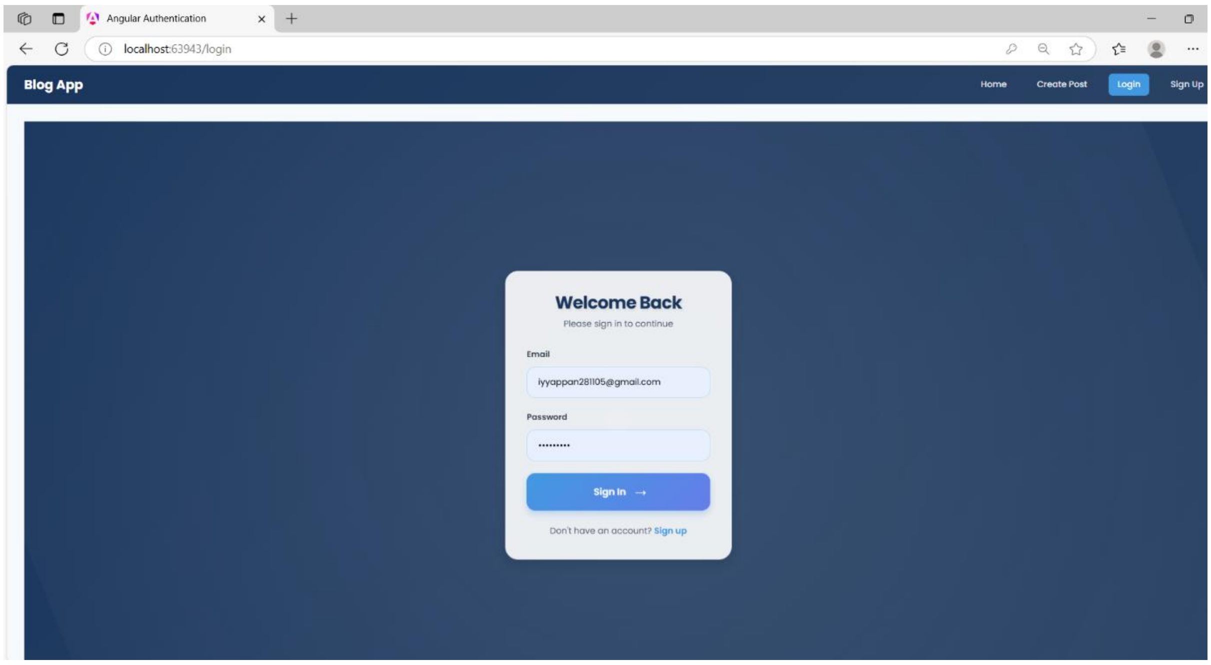
Enter description



Inspiration for your first order

Top brands for you

Customer Review



Welcome Back

Please sign in to continue

Email

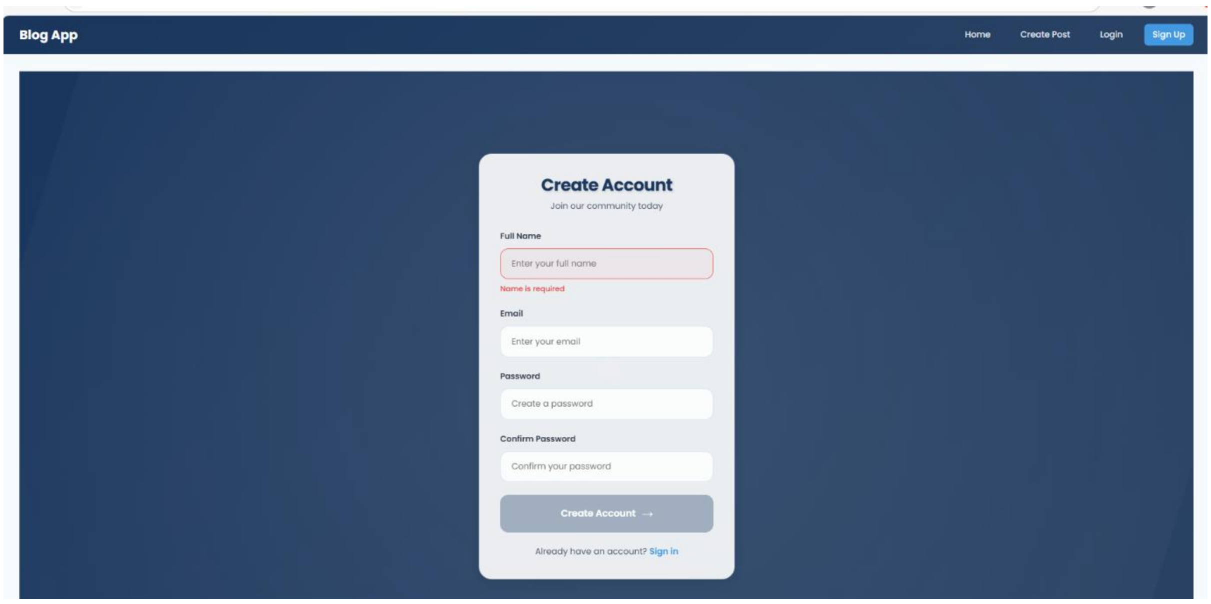
iyyappan281105@gmail.com

Password

•••••••••

Sign In →

Don't have an account? Sign up



Create Account

Join our community today

Full Name

Enter your full name

Name is required

Email

Enter your email

Password

Create a password

Confirm Password

Confirm your password

Create Account →

Already have an account? Sign in

CONCLUSION:

This comprehensive food delivery application provides a robust and scalable solution built with modern technologies like Angular 16 for the frontend and Node.js with MongoDB for the backend. The system delivers an exceptional user experience through intuitive interfaces, real-time order tracking, and secure authentication, while offering powerful business tools like an admin dashboard for efficient management. Designed with performance and flexibility in mind, the modular architecture ensures easy maintenance and future expansion. With features like payment integration, review systems, and responsive design, this application not only meets current market demands but also provides a competitive advantage in the food delivery industry

REFERENCE:

- ***"Angular Up & Running"* (2023) - Amazon link**
- ***"Node.js Design Patterns"* (2020) - Packt link**
- **https://www.mongodb.com/docs/drivers/node/current/**
- **https://www.w3schools.com/angular/default.asp**