****

**Created by : Nour Othman**

**ArtiKeys Company**

**1. Project Overview**

* **Project Name:** SUFRA.COM \_ Home Cooking Platform
* **Project Description:** This project is a web-based platform that enables home-based chefs to showcase their culinary skills and sell homemade dishes online. The platform allows users to browse categories, explore different chefs, view dishes, and place orders. Chefs can register, manage their dishes, and receive reviews from customers.
* **Goals:**
  + Create a dynamic and user-friendly platform where home chefs can promote their products.
  + Provide a seamless experience for customers to browse and purchase dishes.
  + Enable chefs to manage their profiles and dish offerings through an admin dashboard.
* **Key Features:**
  + Product listing and browsing by category and chef.
  + User registration and login system for both customers and chefs.
  + Add to cart and order placement functionality.
  + Chef dashboard to manage dishes and profile.
  + User ratings and reviews for dishes and chefs.

**2. Development Environment**

* **Programming Languages:** ASP.NET Core (Backend), HTML, CSS, JavaScript (Frontend)
* **Database:** SQL Server
* **Tools/Frameworks Used:**
  + Bootstrap for responsive design and layout.
  + Entity Framework Core for ORM and database interactions.
  + ASP.NET Identity for user authentication and authorization.
* **Server Environment:** IIS, Azure, or any other hosting environment.

**3. Project Structure**

* **Main Folders and Files:**
  + **Controllers:** Handles the logic of the application.
    - ChefsController: Manages chef profiles and related operations.
    - CategoriesController: Handles food categories and subcategories.
    - DishesController: Manages individual dishes and their details.
    - AuthController: Handles user authentication (login, registration).
  + **Models:** Represents the data structure.
    - Chef: Contains information about the chef (name, experience, ratings, etc.).
    - Category: Represents a food category (e.g., meals, desserts).
    - SubCategory: Represents subcategories within a category.
    - Dish: Contains details of the individual dishes (name, description, price, etc.).
    - Review: Stores reviews left by customers on dishes or chefs.
  + **Views:** Handles the front-end user interface.
    - HomeView: Displays the main categories and featured chefs.
    - ChefProfileView: Displays individual chef profiles, including their dishes.
    - DishDetailsView: Displays detailed information about a dish, including reviews.
    - LoginView, RegisterView: User authentication pages.
  + **Areas:**
    - Admin: The area used for managing chef registrations, category management, and order processing.

**4. Database Design**

* **Database Schema:**
  + Chefs: Stores chef information such as name, location, experience, and profile image.
  + Categories: Stores the main categories (e.g., Meals, Desserts, Snacks).
  + SubCategories: Stores subcategories under each category (e.g., Traditional Meals, Pasta).
  + Dishes: Stores information about each dish, including price, description, and availability.
  + Reviews: Stores ratings and reviews provided by customers for each dish or chef.
* **Relationships:**
  + One-to-Many between Categories and SubCategories.
  + One-to-Many between Chefs and Dishes.
  + Many-to-Many between Customers and Dishes for reviews and ratings.

**5. Main Components**

* **Authentication & Authorization:**
  + Chef and customer registration are handled through separate forms.
  + After chef registration, the admin must approve the chef before they can access the dashboard.
  + Customers can directly register and place orders after signing in.
* **Category and Product Management:**
  + The admin can create, edit, and delete categories and subcategories.
  + Chefs can manage their own dishes by adding new ones, updating existing ones, and deleting them.
* **Rating and Review System:**
  + Customers can leave reviews and rate both dishes and chefs.
  + Reviews are displayed publicly on the dish and chef profile pages.

**6. User Interface**

* **Home Page:**
  + Displays a selection of featured categories and chefs.
  + Offers filters for browsing dishes by category or chef.
* **Chef Profile Page:**
  + Displays a chef's profile, including their biography, reviews, and list of available dishes.
  + Customers can view detailed information about each dish and leave reviews.
* **Dish Details Page:**
  + Displays the name, price, description, and reviews for a specific dish.
  + Customers can add dishes to their cart and proceed to checkout.
* **Cart and Checkout Pages:**
  + Allows customers to review their selected dishes, adjust quantities, and complete orders.

**7. Admin Dashboard**

* **Chef Management:**
  + Admins can view and approve chef registrations.
  + Admins can deactivate chefs or remove them from the platform.
* **Category and SubCategory Management:**
  + Admins can create, edit, or delete categories and subcategories from the dashboard.
* **Order Management:**
  + Admins and chefs can view, process, and track orders from the dashboard.

**8. Challenges Faced**

* **Database Design:** Ensuring that categories, subcategories, and dishes are flexible enough to accommodate various types of foods and chefs.
* **Dynamic Content:** Implementing dynamic content loading for chefs, dishes, and categories based on admin and chef inputs.
* **Authentication:** Managing multiple user roles (admin, chef, customer) and ensuring the proper flow of permissions.

**9. Future Enhancements**

* **Payment Integration:** Add online payment gateways for seamless transactions.
* **Order Tracking:** Allow customers to track the status of their orders in real-time.
* **Chef Performance Analytics:** Provide analytics tools for chefs to track their sales and reviews over time.

**10. Installation & Setup**

* **Prerequisites:**
  + .NET Core SDK
  + SQL Server (or any compatible database system)
  + Node.js (for frontend dependencies)
* **Installation Steps:**
  + Clone the repository.
  + Install the necessary NuGet packages by restoring the project dependencies.
  + Set up the database using Entity Framework migrations (dotnet ef database update).
  + Build and run the project using Visual Studio or the command line (dotnet run).
  + Access the platform at http://localhost:{port}.