**Software Requirements Specification (SRS)**

**ChefConnect**

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to outline the software requirements for the ChefConnect platform. ChefConnect aims to connect customers and chefs, allowing customers to browse menus, order food, and communicate directly with local chefs. The document defines the scope, functionality, and specifications required for the Minimum Viable Product (MVP).

**1.2 Scope**

ChefConnect is a web-based application that enables:

* Customers to browse local chefs, view menus, and place food orders.
* Chefs to manage their profiles, menus, and process customer orders.
* Administrators to oversee user activity, orders, and system performance.

The platform will be developed using ASP.NET Core with Razor Pages and SQL Server as the database. It will be deployed within 1 month with a single developer.

**1.3 Definitions, Acronyms, and Abbreviations**

* **MVP**: Minimum Viable Product
* **Admin**: Administrator responsible for managing the platform
* **SRS**: Software Requirements Specification

**1.4 References**

* Microsoft ASP.NET Core Documentation
* SQL Server Documentation
* Razor Pages Documentation

**2. Overall Description**

**2.1 Product Perspective**

ChefConnect will be an independent web application that integrates standard authentication, food ordering, and management functionalities. The MVP will focus on essential features with a scalable architecture for future enhancements.

**2.2 User Classes and Characteristics**

* **Customer**: End users who browse chefs, view menus, and place orders.
* **Chef**: Service providers who manage menus and handle customer orders.
* **Admin**: Overseers of platform operations and management.

**2.3 Operating Environment**

* **Platform**: Web-based (PC and Mobile compatible).
* **Technology Stack**: ASP.NET Core, Razor Pages, SQL Server, Bootstrap for UI.
* **Hosting**: Cloud-hosted on Azure or AWS.

**2.4 Constraints**

* The platform must be delivered within one month.
* Single developer working on the project.
* Limited to core features (MVP scope).

**2.5 Assumptions and Dependencies**

* Users will have access to the internet.
* Third-party services (email/SMS notifications) will be integrated if necessary.

**3. Specific Requirements**

**3.1 Functional Requirements**

**3.1.1 Customer Features**

* **Account Management**:
  + Register a new account.
  + Log in and log out.
  + Reset password.
* **Browse Chefs**:
  + View list of chefs by categories (e.g., cuisine, location, ratings).
  + Search chefs or dishes.
* **Order Food**:
  + Add dishes to a cart.
  + Place orders with specific instructions and delivery/pickup time.
  + View order status (e.g., Processing, Completed).
* **Order History and Reviews**:
  + View past orders.
  + Rate and review chefs/dishes.

**3.1.2 Chef Features**

* **Account Management**:
  + Register as a chef.
  + Manage profile information (name, description, location, photos).
* **Menu Management**:
  + Add, edit, or delete menu items.
  + Include details such as name, description, price, and photos.
* **Order Management**:
  + View new orders and customer details.
  + Update order status (e.g., Accepted, Preparing, Completed).
* **Feedback Management**:
  + View and respond to customer reviews.

**3.1.3 Admin Features**

* **User Management**:
  + View and manage customer and chef accounts.
  + Block/unblock users if needed.
* **Order Monitoring**:
  + View all active and completed orders.
  + Generate basic reports (e.g., order volume, revenue).
* **Content Moderation**:
  + Monitor and remove inappropriate reviews or feedback.
* **System Monitoring**:
  + Track application logs for security and performance.

**3.2 Non-Functional Requirements**

* **Performance**: Ensure response time under 3 seconds for common operations.
* **Scalability**: The platform should be designed to handle up to 500 concurrent users.
* **Usability**: Intuitive UI design with mobile-first responsiveness.
* **Security**:
  + Passwords stored with encryption (e.g., ASP.NET Identity).
  + Secure communication using HTTPS.
* **Availability**: 99.5% uptime during operational hours.

**4. System Design Constraints**

* **Database**: SQL Server to store user data, menu items, and orders.
* **Framework**: ASP.NET Core with Razor Pages for fast development.
* **Deployment**: Cloud-hosted to ensure scalability and availability.

**5. Appendices**

**5.1 Future Enhancements**

* Real-time chat between customers and chefs.
* Integrated payment gateway for online payments.
* Delivery tracking for real-time updates.

**5.2 References**

* ASP.NET Core Identity Documentation
* Entity Framework Core Guide