Software Requirements Specification (SRS)

Project Title: KhanaKhazana – Food Ordering & Delivery Web Application

Prepared by: Syntax-slayer23

Date: September 2025

1. Introduction

1.1 Purpose
The purpose of this document is to define the requirements for the KhanaKhazana web application. The system allows users to browse menus, place food orders, and make payments. Vendors (via the admin panel) can manage dishes, update prices, and process orders.

1.2 Scope
KhanaKhazana is a full-stack food ordering and delivery system that consists of:
- Frontend (User Side): HTML, CSS, JavaScript
- Backend: Node.js, Express.js
- Database: MongoDB
- Payment Gateway: Stripe (Card/UPI) & COD

The system supports two main user roles:
- Customers: Browse, order, pay, and track their food orders.
- Vendors/Admins: Add, update, and delete dishes, manage incoming orders, and update statuses.

1.3 Definitions, Acronyms, and Abbreviations
- COD – Cash on Delivery
- UPI – Unified Payments Interface
- Stripe – Third-party payment processing gateway
- CRUD – Create, Read, Update, Delete

1.4 References
- Stripe API Documentation: https://stripe.com/docs
- MongoDB Documentation: https://www.mongodb.com/docs

2. Overall Description

2.1 Product Perspective
KhanaKhazana is a standalone web-based system built using the MERN stack (MongoDB, Express, Node.js + vanilla JS for frontend).

2.2 Product Functions

- User Side:
  - * User Registration/Login
  - * Browse menu and view dish details
  - * Add dishes to cart and checkout
  - * Multiple payment options (COD, UPI, Card)
  - * Order tracking

- Admin Panel:
  - * Add, update, and delete dishes
  - * Manage menu categories
  - * View and update order status
  - * Remove unavailable dishes

2.3 User Characteristics

- Customers: Basic internet users familiar with food delivery apps.
- Vendors/Admins: Restaurant staff with access to admin panel.

2.4 Constraints

- Requires internet connection.
- Payment gateway integration relies on Stripe APIs.
- Compatible with modern browsers (Chrome, Firefox, Edge).

2.5 Assumptions and Dependencies

- Users have access to devices with internet browsers.
- Stripe and MongoDB services are up and running.

3. System Features

3.1 User Module

- Login/Signup
- Explore menu
- Add to cart
- Place orders
- Payment via Stripe / COD / UPI
- Track order status

3.2 Admin Module

- Secure login for admin
- Add/Update/Delete dishes
- Manage dish details (Name, Description, Price)
- View customer orders
- Update order status (Processing - Out for delivery - Delivered)

4. External Interface Requirements

4.1 User Interfaces

- Frontend: Responsive web pages (HTML, CSS, JavaScript)
- Admin Panel: Dashboard for managing dishes and orders

4.2 Hardware Interfaces

Runs on standard web servers and user devices (PC, laptop, mobile)

4.3 Software Interfaces

- Node.js runtime environment
- MongoDB for database
- Stripe API for payment processing

4.4 Communication Interfaces

- HTTP/HTTPS requests between frontend and backend
- REST API for order and menu management

5. Non-Functional Requirements

- Performance: Handle up to 1000 concurrent users
- Security: User authentication & payment gateway encryption
- Reliability: 99.5% uptime expected
- Scalability: Easily scalable to add more restaurants or features
- Usability: Simple, intuitive UI

6. Other Requirements

- Source code hosted on GitHub for version control
- Documentation provided (README + SRS)
- Future enhancements may include:
  - Delivery partner tracking system
  - Push notifications for order updates
  - Multi-language support