**Software Requirements Specification (SRS) Template**

**Project: Food Delivery System**

**Version:** 1.0

**Authors:** < Namith B U, Manuprasad H S, Jaypal Reddy, Manoj R / Bytezz >

**Date:** 03-09-2025

**Status:** Draft / For Review

**Revision history**

| Version | Date | Author | Change summary | Approval |
| --- | --- | --- | --- | --- |
| 1.0 | 03-09-2025 | < Namith B U, Manuprasad H S, Jaypal Reddy, Manoj R > | Initial draft of SRS for Food Delivery System |  |

**Approvals**

| Role | Name | Signature / Email | Date |
| --- | --- | --- | --- |
| Course Coordinator |  |  |  |
| Project Guide |  |  |  |

**Table of Contents**

1. Introduction
2. Overall description
3. External interfaces
4. System features (detailed)
5. Non-functional requirements (detailed)
6. Quality attributes & Acceptance tests
7. UML Use-Case Diagram
8. Requirements Traceability Matrix (RTM)

**1. Introduction**

**1.1 Purpose**

This document is a Software Requirements Specification (SRS) for a Food Delivery System (FDS). It defines the functional and non-functional requirements, interfaces, and verification criteria for the system. It is intended for developers, QA engineers, project managers, and stakeholders to serve as a foundational agreement on what the system will do.

**1.2 Scope**

This SRS covers the core functionalities of the food delivery platform, including user registration and management for customers, restaurants, and delivery partners. It encompasses features such as searching for restaurants, placing orders, processing payments, real-time order tracking, and a rating/review system. It also includes the administrative backend for monitoring and managing the platform's operations.

The scope excludes internal kitchen management systems for restaurants, payroll and HR systems for delivery partners, and advanced financial accounting beyond basic transaction settlement reports.

**1.3 Audience**

This document is intended for Development Teams, QA & Testing Engineers, Product Managers, Restaurant Partners, and System Administrators.

**1.4 Definitions**

List of acronyms:

* **FDS:** Food Delivery System
* **PII:** Personally Identifiable Information
* **UI:** User Interface
* **API:** Application Programming Interface
* **GPS:** Global Positioning System
* **OTP:** One-Time Password
* **PCI-DSS:** Payment Card Industry Data Security Standard
* **TLS:** Transport Layer Security

**2. Overall description**

**2.1 Product perspective**

The Food Delivery System is a comprehensive platform connecting customers with local restaurants through a seamless digital experience. It consists of a customer-facing mobile/web application, a restaurant management dashboard, a mobile application for delivery partners, and an administrative backend. The system will operate on a cloud-based infrastructure, leveraging third-party services for payments, mapping, and notifications to ensure a reliable and scalable service.

**2.2 Major product functions (detailed)**

* **User Authentication:** Secure registration and login for all user roles (Customer, Restaurant, Delivery Partner).
* **Restaurant & Menu Management:** Restaurants can manage their profile, menu items, prices, and availability.
* **Order Placement & Payment:** Customers can browse restaurants, add items to a cart, and securely pay for their orders.
* **Order Management:** Restaurants can accept or reject incoming orders and update their status (e.g., "Preparing", "Ready for Pickup").
* **Delivery & Tracking:** Assigns delivery partners to orders and provides real-time GPS tracking for customers.
* **Ratings & Reviews:** Allows customers to rate and review their food and delivery experience.
* **Admin Dashboard:** System administrators can manage users, oversee operations, and generate reports.

**2.3 User roles and characteristics (expanded)**

* **Customer:** A user with basic tech proficiency who wants to order food easily and track it until delivery.
* **Restaurant Manager:** Staff at partner restaurants who need a simple interface to manage online orders, update menus, and track earnings.
* **Delivery Partner:** An individual who uses a mobile app to receive delivery tasks, navigate to the restaurant and customer, and manage their earnings.
* **System Administrator:** A technical user responsible for platform health, user management, and resolving disputes.

**2.4 Operating environment**

The system will be a cloud-native application, hosted on platforms like AWS, Azure, or GCP. It will require a modern web browser for web-based dashboards and iOS/Android for mobile applications. A stable internet connection is mandatory for all users. The system must operate 24/7, with planned downtime for maintenance.

**2.5 Constraints**

* The system must comply with data privacy regulations (e.g., GDPR, CCPA) regarding user data.
* All payment processing must adhere to PCI-DSS standards.
* The system will depend on third-party APIs (Google Maps, Payment Gateways), and their availability and terms of service may impact functionality.
* Mobile applications must be compatible with the last two major versions of iOS and Android.

**3. External interface requirements**

**3.1 User interfaces**

* **Customer App (Mobile/Web):** Intuitive UI for searching, ordering, tracking, and payments.
* **Restaurant Dashboard (Web):** A clear and simple interface for order management, menu updates, and viewing reports.
* **Delivery Partner App (Mobile):** A streamlined, map-centric UI for accepting jobs, navigation, and updating delivery status.
* **Accessibility:** Key user flows should aim for compliance with WCAG 2.1 AA standards, including high-contrast mode and screen-reader support.

**3.2 Hardware interfaces**

* **GPS Module:** The mobile devices of customers and delivery partners will be used for location tracking.
* **Printers:** Restaurants may use thermal receipt printers to print order dockets, interfaced via their local computer or tablet.

**3.3 Software interfaces**

* **Payment Gateway API:** Integration with a third-party payment provider (e.g., Stripe, Razorpay) via REST API over HTTPS to handle all transactions.
* **Mapping Service API:** Integration with a mapping service (e.g., Google Maps API) for address autocompletion, calculating distances, and displaying routes.
* **Notification Service API:** Integration with services like Firebase Cloud Messaging (FCM) or Twilio for sending push notifications and SMS alerts.

**3.4 Communications**

* All communication between client applications and the backend server shall be encrypted using TLS 1.2+.
* The system shall implement a reliable mechanism for handling intermittent network connectivity on mobile devices, such as retrying requests and local caching.

**4. System features (detailed)**

Each requirement below includes acceptance criteria and a reference test case. IDs follow FDS-F-###.

**4.1 User Authentication & Profile Management**

Description: Allow users to register, log in, and manage their profiles securely.

| Req ID | Requirement | Type | Priority | Source/Stakeholder | Acceptance criteria / Test case ref | Comments / Dependencies |
| --- | --- | --- | --- | --- | --- | --- |
| FDS-F-001 | The system shall allow new users to register as a Customer, Restaurant, or Delivery Partner with a unique email address. | Functional | High | Product Manager | AC-FDS-F-001: User can complete registration and receives a verification email. Test: TC-Auth-01 |  |
| FDS-F-002 | The system shall authenticate users using their email and a securely stored password. | Functional | High | Security | AC-FDS-F-002: Authenticated user is granted access to their respective dashboard/app. Test: TC-Auth-02 | Passwords must be hashed. |
| FDS-F-003 | The system shall provide a "Forgot Password" functionality that allows users to reset their password via a secure link sent to their registered email. | Functional | High | Customer | AC-FDS-F-003: User successfully resets password and can log in with the new one. Test: TC-Auth-03 |  |

**4.2 Restaurant Search and Ordering**

Description: Enable customers to find restaurants, browse menus, and place orders.

| Req ID | Requirement | Type | Priority | Source/Stakeholder | Acceptance criteria / Test case ref | Comments / Dependencies |
| --- | --- | --- | --- | --- | --- | --- |
| FDS-F-010 | The system shall allow customers to search for restaurants by name, cuisine, or location. | Functional | High | Customer | AC-FDS-F-010: Search returns relevant results that can be sorted or filtered. Test: TC-Search-01 |  |
| FDS-F-011 | The system shall display detailed restaurant information, including menu, prices, ratings, and estimated delivery time. | Functional | High | Customer | AC-FDS-F-011: All specified details are visible and accurate on the restaurant page. Test: TC-Menu-01 |  |
| FDS-F-012 | The system shall allow customers to add/remove items from a shopping cart and view the total cost, including taxes and fees. | Functional | High | Business | AC-FDS-F-012: Cart updates correctly in real-time. Test: TC-Order-01 |  |
| FDS-F-013 | The system shall process payments through an integrated payment gateway. | Functional | High | Business | AC-FDS-F-013: Successful payment results in order confirmation. Test: TC-Payment-01 | Requires Payment Gateway API. |
| FDS-F-014 | The system shall send an order confirmation to the customer and the restaurant via notification and email. | Functional | High | Operations | AC-FDS-F-014: Both parties receive confirmation immediately after a successful order. Test: TC-Notify-01 |  |

**4.3 Order and Delivery Management**

Requirements: Allow restaurants to manage orders and enable real-time tracking for customers. IDs: FDS-F-020..FDS-F-025.

* **FDS-F-020:** Restaurants shall be able to accept or reject new orders from their dashboard.
* **FDS-F-021:** The system shall automatically assign a nearby, available delivery partner to an accepted order.
* **FDS-F-022:** Customers shall be able to track the status of their order in real-time (e.g., "Accepted", "Preparing", "Out for Delivery").
* **FDS-F-023:** Customers shall be able to view the delivery partner's location on a map once the order is "Out for Delivery".

**4.4 Ratings and Admin Functions**

Requirements: Allow customers to provide feedback and administrators to manage the platform. IDs: FDS-F-030..FDS-F-033.

* **FDS-F-030:** After delivery, customers shall be able to rate the restaurant and the delivery partner on a scale of 1-5.
* **FDS-F-031:** Administrators shall have a dashboard to view system-wide statistics (e.g., total orders, revenue).
* **FDS-F-032:** Administrators shall be able to manage user accounts (e.g., activate/deactivate restaurants).

**5. Non-functional requirements (detailed)**

NFRs below are measurable and tied to test plans. IDs FDS-NF-###.

| Req ID | Requirement | Category | Priority | Acceptance criteria / Measurement |
| --- | --- | --- | --- | --- |
| FDS-NF-001 | API response time for critical functions (search, order placement) shall be ≤ 500ms for the 95th percentile under normal load. | Performance | High | 95th percentile ≤ 500ms in load tests. Test: TC-Perf-01 |
| FDS-NF-002 | The system shall maintain 99.9% availability, excluding scheduled maintenance windows announced 24 hours in advance. | Reliability | High | Uptime monitoring reports show ≥99.9% monthly availability. |
| FDS-NF-003 | All sensitive user data (PII, passwords) must be encrypted at rest and in transit. Payment card data must not be stored on the system. | Security | High | Security audit confirms encryption and PCI-DSS compliance. Test: TC-Sec-01 |
| FDS-NF-004 | The mobile applications must be intuitive enough for a new user to place their first order in under 3 minutes without assistance. | Usability | High | 9/10 users in a usability test complete the task successfully within the time limit. Test: TC-UX-01 |
| FDS-NF-005 | The system architecture must be able to scale horizontally to handle a 3x increase in user traffic during peak hours without performance degradation. | Scalability | High | System maintains performance targets (FDS-NF-001) during a load test simulating 3x peak traffic. Test: TC-Scale-01 |

**5.1. Security**

**5.1.1 Security Objectives**

1. **Confidentiality:** To protect all Personally Identifiable Information (PII) and payment details from unauthorized disclosure.
2. **Integrity:** To ensure that all order details, transaction amounts, and user-generated content (reviews) are accurate and cannot be modified by unauthorized parties.

**5.1.2 Security Requirements**

| Req ID | Requirement | Type | Priority | Acceptance criteria / Test case ref |
| --- | --- | --- | --- | --- |
| FDS-SR-001 | All communication between clients and servers shall be encrypted using TLS 1.2 or higher. | Security | High | Penetration test confirms no unencrypted traffic. Test: TC-SEC-02 |
| FDS-SR-002 | The system shall enforce Role-Based Access Control (RBAC) to ensure users can only access data and functions appropriate for their role. | Security | High | Test cases verify that users cannot perform actions outside their defined permissions. Test: TC-SEC-03 |
| FDS-SR-003 | The system must implement protection against the OWASP Top 10 vulnerabilities, including SQL Injection, XSS, and CSRF. | Security | High | Vulnerability scan and code review show no critical vulnerabilities. Test: TC-SEC-04 |
| FDS-SR-004 | The system shall hash user passwords using a modern, salted hashing algorithm (e.g., bcrypt). | Security | High | Database audit confirms passwords are not stored in plaintext. Test: TC-SEC-05 |
| FDS-SR-005 | The system shall provide an audit trail by logging all critical actions (e.g., login attempts, order placement, profile changes). | Security | High | Logs are generated for all critical events and stored securely. Test: TC-SEC-06 |

**6. Quality attributes & Acceptance tests**

* **Exit criteria for acceptance:** All high-priority functional requirements (e.g., ordering, payment, tracking) are implemented and verified. No critical NFR failures are observed in performance, security, or reliability tests. The RTM shows 100% pass rate for all critical-path test cases.
* **Acceptance test suites:** End-to-end Order Flow, Payment Gateway Integration, Real-time Tracking, User Role Permissions, Performance & Load Testing, and Security Vulnerability Assessment.

**7. System models and diagrams**

**7.1 UML Use-Case diagram**

At least two UML use-case diagrams will be created: one detailing the "Customer" actor's interactions (e.g., Search, Place Order, Track Order) and another for the "Restaurant Manager" actor (e.g., Manage Menu, Process Orders).

**8. Requirements Traceability Matrix (RTM)**

| Req ID | Requirement short | Section ref / Design Spec | Module | Test case(s) | Status (N/P/A) | Comments |
| --- | --- | --- | --- | --- | --- | --- |
| FDS-F-002 | User login | 4.1 / DS-Auth-01 | AuthModule | TC-Auth-02 | N |  |
| FDS-F-013 | Process payment | 4.2 / DS-Payment-01 | PaymentModule | TC-Payment-01 | N | Depends on Gateway API |
| FDS-F-022 | Real-time order status | 4.3 / DS-Track-01 | OrderModule | TC-Track-01 | N |  |
| FDS-NF-001 | API response time | 5 / DS-Perf-01 | API Gateway / Core | TC-Perf-01 | N |  |
| FDS-SR-002 | Role-Based Access | 5.1.2 / DS-Auth-02 | AuthModule | TC-SEC-03 | N |  |