

# Software Requirements Specification (SRS)

## Project: Food Delivery Service System

### Introduction

#### 1.1 Purpose

This Software Requirements Specification (SRS) defines the requirements for the Food Delivery Service system. The system is designed to provide customers with a convenient, reliable, and secure way to order food from multiple restaurants. It also streamlines order management for restaurants and delivery agents, while enabling administrators to oversee and manage the platform.

#### 1.2 Scope

The system allows customers to register, browse menus, add items to a cart, place orders, make payments, and track deliveries. Restaurants can update menus, accept/reject orders, and view analytics. Delivery agents receive assignments, update delivery status, and track routes using GPS. The platform excludes third-party food preparation but integrates with external payment gateways and notification services.

#### 1.3 Audience

This document is intended for:

- Developers: For system design and implementation.
- QA Engineers: For preparing test plans and validation.
- System Integrators: For integrating APIs and services.
- Maintenance Technicians: For system upkeep and troubleshooting.
- Business Stakeholders: For requirement validation and compliance.

#### 1.4 Definitions

API – Application Programming Interface

OTP – One-Time Password

ETA – Estimated Time of Arrival

HTTPS – Hypertext Transfer Protocol Secure

PCI-DSS – Payment Card Industry Data Security Standard

GDPR – General Data Protection Regulation

### Overall Description

#### 2.1 Product Perspective

The Food Delivery Service is a client-server platform accessible via web and mobile applications. It integrates with external services such as payment gateways, SMS/email APIs, and GPS for real-time delivery tracking. The solution provides interconnected modules for customers, restaurants, delivery agents, and administrators.

#### 2.2 Major Product Functions

- Customer account creation and OTP verification
- Restaurant and dish browsing with filters
- Shopping cart and order placement
- Secure payment processing

- Real-time order and delivery tracking
- Restaurant menu and pricing management
- Delivery agent assignment and navigation
- Analytics and reporting for administrators

## 2.3 User Roles and Characteristics

- Customers: Require intuitive UI and fast response.
- Restaurants: Need stable dashboards, easy menu updates, and order management.
- Delivery Agents: Require mobile tools for assignments, navigation, and delivery confirmation.
- Admins: Require monitoring, reporting, and user management tools.

## 2.4 Operating Environment

The system supports:

- Web: Chrome, Firefox, Edge, Safari
- Mobile: iOS 13+ and Android 8+
- Backend: Cloud-based Linux servers with HTTPS/TLS
- APIs: Payment, Notification, and GPS services

## 2.5 Constraints

- PCI-DSS compliance for payment security
- GDPR/local compliance for data privacy
- Multi-language support (English + regional)
- High traffic handling during peak loads- 99.9% system uptime SLA

# External Interface Requirements

## 3.1 User Interfaces

The system provides responsive, user-friendly designs:

- Customers: Menu browsing, cart management, checkout, order tracking
- Restaurants: Order dashboard, menu editor, analytics
- Delivery Agents: Mobile app with assignments, delivery status, GPS tracking
- Admins: Web console for monitoring and reporting

## 3.2 Hardware Interfaces

- Delivery Agents: Smartphones with GPS modules
- Restaurants: Optional barcode scanners for inventory
- Servers: Cloud infrastructure with load balancing

## 3.3 Software Interfaces

- Payment Gateway APIs (Stripe, Razorpay, PayPal)
- SMS/Email APIs (Twilio, SendGrid)
- Restaurant Management APIs
- Google Maps API for GPS services

## 3.4 Communications

- HTTPS/TLS 1.2+ for all communications
- JSON payloads for APIs
- WebSockets for real-time updates
- Push notifications for users and delivery agents

## 4. FUNCTIONAL REQUIREMENTS

Req ID	Type	Requirement	Priority	Acceptance Criteria
FR-001	Functional	Customers can create an account and verify their account with OTP	High	Only verified and logged in accounts can order
FR-002	Functional	Customers can search via restaurants / dish categories	Medium	Displaying filtered restaurants and categories
FR-003	Functional	Customers can search via dishes names	Medium	Displays searched dish from various restaurants
FR-004	Functional	Customers can add and remove items from cart	Medium	Cart is updated and shows total cost
FR-005	Functional	Customers should select quantity	High	Cost is calculated as per quantity
FR-006	Functional	Customers can choose add ons or extras	Medium	Extra costs are added
FR-007	Functional	Customers should provide address for delivery	High	Estimated delivery time is displayed
FR-008	Functional	Customers can apply discount coupon codes	Medium	Coupon code applied successfully
FR-009	Functional	Customers should select payments method and complete payment to place order	High	Displayed order placed message
FR-010	Functional	Customers can track estimated delivery time	Low	ETA is displayed below the order placed message
FR-011	Functional	Restaurants can either accept or reject the order	High	Order updates correctly
FR-012	Functional	Delivery agents can view their delivery details and location	High	Agents receive jobs

FR-013	Functional	Restaurants can alter dishes and their pricings	Medium	Menu is updated
FR-014	Functional	Restaurants can take ratings from customers	Medium	Rating saved
FR-015	Functional	Admin can generate order reports	High	Report generated

## 5. NON-FUNCTIONAL REQUIREMENTS

Req ID	Type	Requirement	Priority	Acceptance Criteria
NFR-001	Non-functional	Every order should not take longer than 30min	High	Helps in stock monitoring
NFR-002	Non-functional	80% payments within 3s	High	Highly responsive
NFR-003	Non-functional	Supports up-to 50,000 users at once	High	Scalable to real world
NFR-004	Non-functional	Responsive user interface for all devices	Medium	Good interface attracts customers
NFR-005	Non-functional	Multi language support	Medium	Users can switch between languages

## 6. Security Objectives & Requirements:

Objective ID	Security Objective	Requirement ID	Security Requirement
SO1	Protect customer personal & payment data	SR1	All communication must use SSL/TLS (HTTPS)

<b>SO2</b>	<b>Ensure authentication &amp; authorization</b>	<b>SR2</b>	<b>Implement Multi-Factor Authentication (MFA) for admins &amp; delivery partners</b>
<b>SO3</b>	<b>Secure payment transaction</b>	<b>SR3</b>	<b>Integrate PCI-DSS compliant payment gateways</b>
<b>SO4</b>	<b>Maintain data integrity</b>	<b>SR4</b>	<b>Use Role-Based Access Control (RBAC) to restrict unauthorized data modifications</b>
<b>SO5</b>	<b>Ensure availability &amp; reliability</b>	<b>SR5</b>	<b>Perform regular audits and protect system against DDoS, SQL injection, brute force</b>

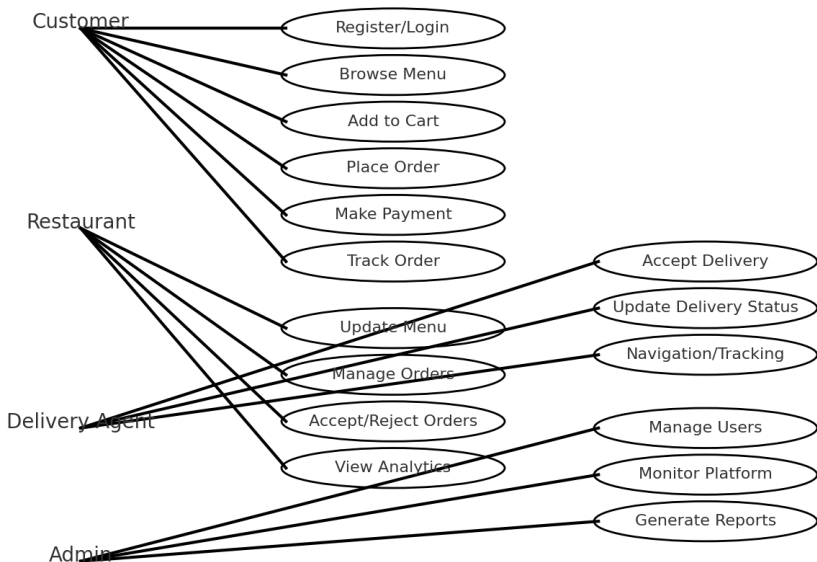
## 7. Quality Attributes & Acceptance Tests:

<b>Attribute ID</b>	<b>Quality attribute</b>	<b>Description</b>	<b>Acceptance</b>
<b>QA1</b>	<b>Performance</b>	<b>System must handle multiple requests quickly</b>	<b>Simulate 5000 concurrent orders → average response time &lt; 3 seconds</b>
<b>QA2</b>	<b>Usability</b>	<b>User interface should be simple and intuitive</b>	<b>Simulate 5000 concurrent orders → average response time &lt; 3 seconds</b>
<b>QA3</b>	<b>Reliability</b>	<b>System should ensure order consistency and high uptime</b>	<b>Simulate server restart → active orders and payments remain intact</b>
<b>QA4</b>	<b>Scalability</b>	<b>System must scale during peak</b>	<b>Test with 10x normal traffic →</b>

		<b>loads (festivals, weekends)</b>	<b>system remains stable without downtime</b>
<b>QA5</b>	<b>Maintainability</b>	<b>System must scale during peak loads (festivals, weekends)</b>	<b>Deploy a new version update without interrupting existing active user sessions</b>

## 8. UML Use-Case Diagram

The UML Use-Case diagram visually represents the interactions between system actors and the system functionalities. The diagram identifies four main actors: Customer, Restaurant, Delivery Agent, and Admin. Each actor has distinct responsibilities and system interactions.



**Figure 1: UML Use-Case Diagram for Food Delivery System**

## 9. Requirements Traceability Matrix (RTM)

The Requirements Traceability Matrix (RTM) provides a mapping between the defined requirements and their corresponding system modules and test cases. This ensures that all requirements are covered, traceable, and verifiable.

Req. ID	Requirement Description	Source	System Module	Test Case ID
FR-1	Customer Registration/Login	Stakeholder	Auth Service	TC-01
FR-2	Browse Menus with Filters	Stakeholder	Menu Module	TC-02
FR-3	Add Items to Cart	Stakeholder	Cart Module	TC-03
FR-4	Place Food Order	Stakeholder	Order Module	TC-04
FR-5	Secure Payment	PCI-DSS	Payment Gateway	TC-05
FR-6	Track Delivery in Real-Time	Stakeholder	GPS/Tracking Module	TC-06
FR-7	Restaurant Menu Management	Stakeholder	Restaurant Dashboard	TC-07
FR-8	Delivery Agent Assignment	Stakeholder	Delivery Module	TC-08
FR-9	Delivery Status Update	Stakeholder	Delivery Module	TC-09
FR-10	Admin User Management	Stakeholder	Admin Console	TC-10
NFR-1	System Uptime 99.9%	Business Req	Infrastructure	TC-11
NFR-2	Data Privacy (GDPR)	Legal Req	Security	TC-12
NFR-3	Multi-Language Support	Stakeholder	UI/UX	TC-13
NFR-4	High Traffic Handling	Business Req	Cloud Infra	TC-14
NFR-5	Secure Communication (HTTPS)	Security Req	API Layer	TC-15