Software Architecture and Design Specification

Project: Online Food Delivery System

Version: 1.3

Authors: JAYPAL REDDY, NAMITH B U, MANOJ R, MANUPRASAD H S

Date: 01-10-2025

Status: Draft

Approvals

Role	Name	Responsibility
QA Lead	JAYPAL REDDY	Prepare plan, coordinate execution
Test Engineer	MANOJ R	Design & execute test cases, log defects
Developer	NAMITH B U	Support defect fixes and triage
Product Owner	MANUPRASAD H S	Approve test results, sign-off readiness

1. Introduction

1.1 Purpose

The purpose of this Software Architecture Document (SAD) is to define the architectural components, system structure, data flow, integration points, and design decisions for the **Food Delivery System (FDS)**.

This SAD directly corresponds to the requirements defined in the **SRS document** and provides the **design specification codes (DS-XXX)** used in the **RTM**.

1.2 Scope

Architecture overview
Component-level design
API design
UML diagrams
Module descriptions

Security architecture
Mapping of design elements to SRS requirements

1.3 Audience

Developers
QA/Testers
Architects
Project Managers

System Administrators

1.4 Definitions

- **FDS:** Food Delivery System
- **API:** Application Programming Interface
- **OTP:** One-Time Password
- **Microservice:** A small, autonomous service that is independently deployable and scalable.
- **RBAC:** Role-Based Access Control.
- **JWT:** JSON Web Token, used for securely transmitting information.

2. Document Overview

2.1 How to use this document

This document provides the key architectural and design deliverables. It should be used as the primary guide for development teams to ensure the system is built according to the specified architecture. It includes component diagrams, architectural decisions, API designs, and security models.

2.2 Related Documents

- FDS SRS v1.0
- Software Test Plan (STP) v1.0
- Design Specifications v1.0 .

3. Architecture

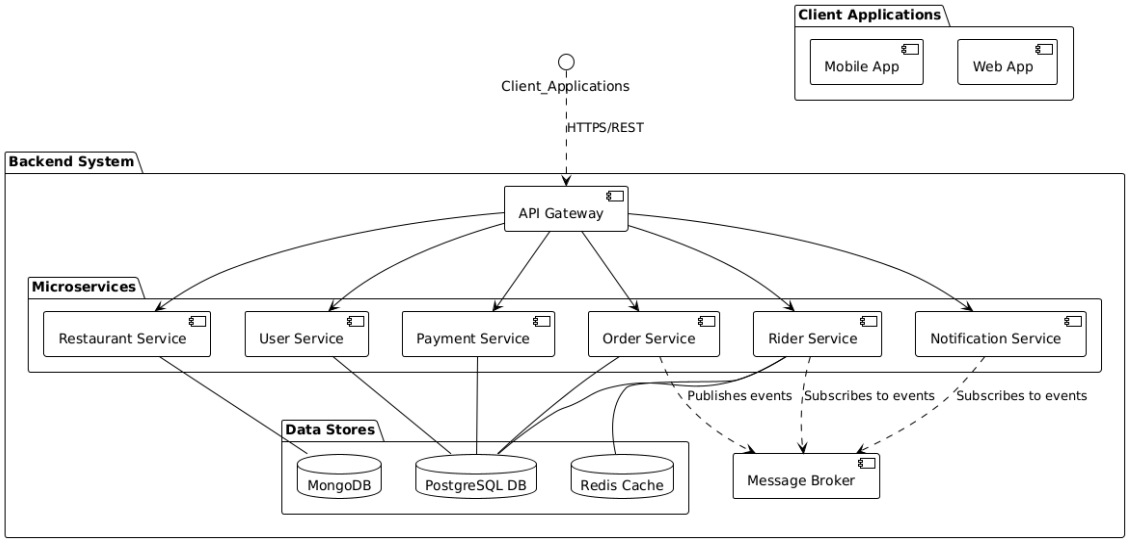
3.1 Goals & Constraints

- **Goals:**
    - **High Availability:** The system must achieve 99.9% availability.
    - **Scalability:** The architecture must scale horizontally to handle fluctuating loads.
    - **Low Latency:** Critical API response times should be under 500ms.
- **Constraints:**
    - Compliance with PCI-DSS standards for payment processing.
    - Integration with third-party services for mapping and payment processing.

3.2 Stakeholders & Concerns

- **Customers:** Security of payment data, ease of use, real-time order status.
- **Restaurant Partners:** Reliability of order notifications, ease of menu management.
- **Delivery Riders:** Efficient route planning, accurate location tracking.
- **Platform Administrators:** System maintainability, scalability, monitoring, and security.

3.3 Component (UML) Diagram



3.4 Component Descriptions

Auth Module

Design Spec: DS-Auth-01 / DS-Auth-02
Maps to SRS:

- FDS-F-001 (Signup/Login)
- FDS-F-003 (RBAC)
- FDS-SR-002 (Security RBAC)

Functions:

- JWT authentication
- Password hashing (bcrypt)
- Email verification
- Reset password

Menu & Cart Module

Design Spec: DS-Menu-01
Maps to SRS:

- FDS-F-010 (Browse Menu)
- FDS-F-011 (Cart CRUD)
- FDS-F-012 (Delivery Fee 15%)

Order Management Module

Design Spec: DS-Order-01
Maps to SRS:

- FDS-F-013 (Place Order)

- FDS-F-014 (Order History)

Payment Module

Design Spec: DS-Payment-01
Maps to SRS:

- FDS-F-020 (Mock Payment Integration)

Delivery Partner Assignment & Tracking Module

Design Spec: DS-Track-01
Maps to SRS:

- FDS-F-030 (Partner Assignment, Tracking)

Admin Module

Design Spec: DS-Admin-01
Maps to SRS:

- FDS-F-040 (Manage Catalogs)
- FDS-F-041 (Manage Order Status)
- FDS-F-050 (Logs & Alerts)

3.5 Chosen Architecture Pattern and Rationale

A Microservices Architecture has been chosen. This pattern is ideal for the FDS as it offers scalability, flexibility, and resilience. Each service can be scaled independently, developed and deployed by separate teams, and the failure of one non-critical service will not bring down the entire application.

3.6 Technology Stack & Data Stores

Layer	Tech
Frontend	React / Flutter
Backend	Node.js or FastAPI
DB	MongoDB / PostgreSQL
Auth	JWT, bcrypt
Notifications	FCM / Twilio
Maps	Google Maps API

3.7 Risks & Mitigations

- Risk: Delay in receiving a stable build for testing.
    - Mitigation: Request early and frequent smoke builds from the development team.
- Risk: Downtime of third-party payment or mapping APIs.
    - Mitigation: Utilize mock APIs and sandbox environments for testing to isolate dependencies.
- Risk: Network instability affecting real-time GPS tracking.
    - Mitigation: Implement retry logic and local caching on the client-side application.

3.8 Traceability to Requirements

The architecture components map to key business requirements as defined in the SRS and tracked via the RTM.

SRS Req ID	SRS Feature	SAD Module	Design Spec

FDS-F-001	Login/Signup	Auth	DS-Auth-01
FDS-F-003	RBAC	Auth	DS-Auth-02
FDS-F-011	Cart	Menu/Cart	DS-Menu-01
FDS-F-013	Order Placement	Order Module	DS-Order-01
FDS-F-020	Payment	Payment Module	DS-Payment-01
FDS-F-030	Delivery Assign/Track	Tracking Module	DS-Track-01
FDS-N-001	Performance	Core	DS-Perf-01
FDS-SR-002	RBAC Security	Auth	DS-Auth-02

3.9 Security Architecture

The security design adheres to the OWASP Testing Guide and focuses on the STRIDE threat model.

- TLS 1.2+ (SRS FDS-SR-001)
- RBAC (SRS FDS-SR-002)
- Protected APIs
- Hashed passwords
- OWASP Top-10 Protection

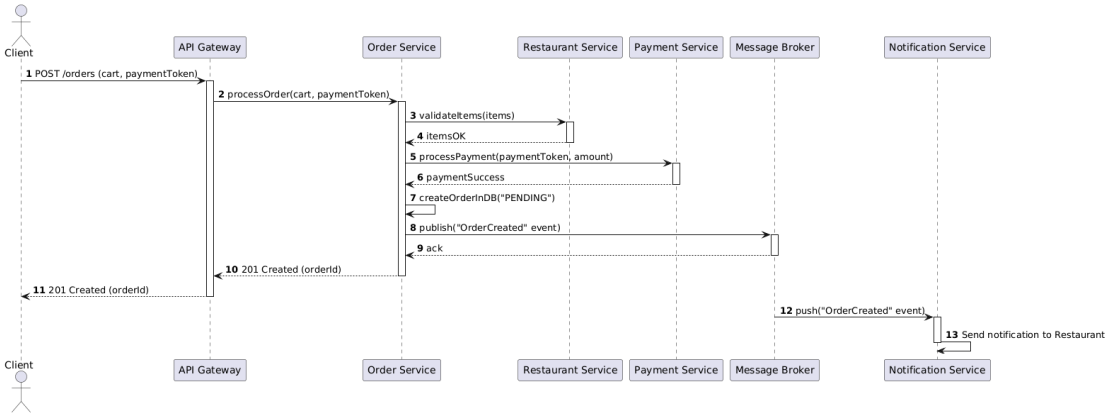
4. Design

4.1 Design Overview

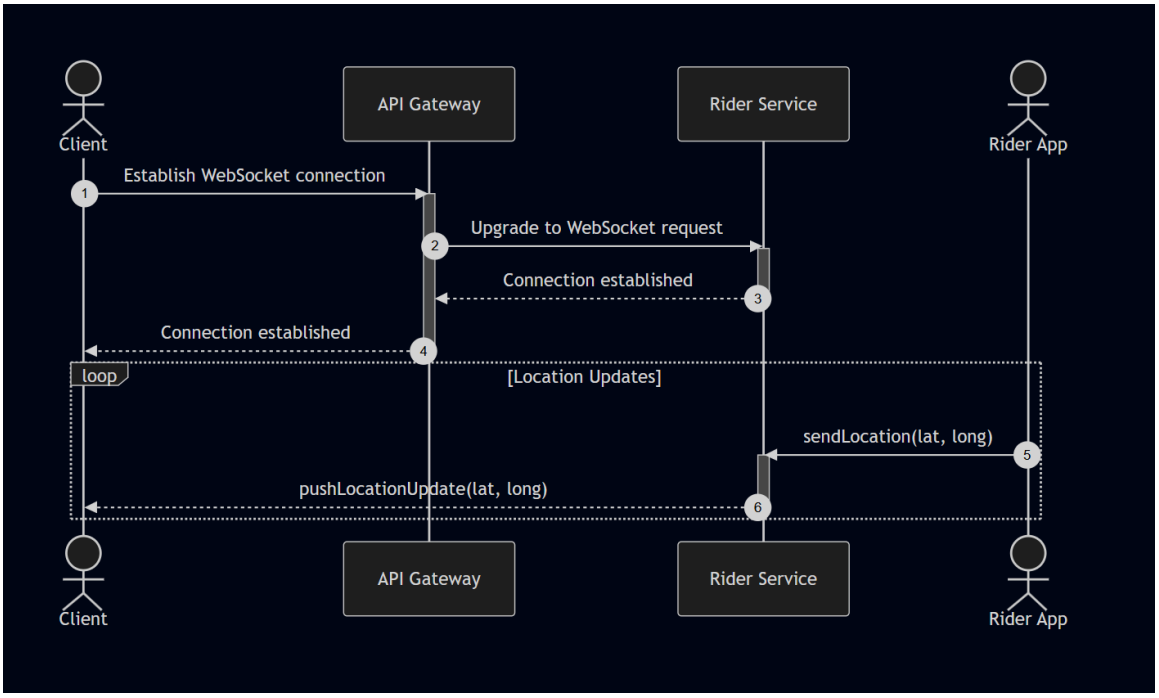
The system is designed using Domain-Driven Design (DDD) principles, where each microservice corresponds to a Bounded Context. This ensures a clean separation of concerns and aligns the software design with the business domain.

4.2 UML Sequence Diagrams

User Places an Order



Real-time Order Tracking



4.3 Authentication Service Design (DS-Auth-01)

This module handles the complete workflow of **user registration and login**, including validation, password hashing, and token generation. The system securely stores user credentials using a hashing algorithm and issues JWT tokens for session management. It also manages error handling for duplicate accounts or invalid credentials.
Maps to SRS: FDS-F-001, FDS-F-002.

4.4 Menu Management Module Design (DS-Menu-01)

This module is responsible for displaying **restaurant details and menu items**. It fetches information such as item name, price, availability, and restaurant metadata. The design includes API endpoints for retrieving menu lists and ensuring fast, structured data delivery to the frontend.
Maps to SRS: FDS-F-011.

4.5 Cart Service Design (DS-Cart-01)

This module manages **cart operations** such as adding items, updating quantities, and removing items. It recalculates totals in real-time, maintains the cart state for each user, and ensures consistency even with multiple updates. The service integrates with menu data to validate item availability and pricing.
Maps to SRS: FDS-F-012.

4.6 Order Confirmation Module Design (DS-OrderConf-01)

This module finalizes the ordering process by validating user and cart data, creating a new order entry, generating a unique order ID, and sending a confirmation response. It ensures that all order details (items, total amount, address) are stored securely and ready for further processing.
Maps to SRS: FDS-F-014.

5. Appendices

5.1 Glossary:

- **FDS:** Food Delivery System
- **API:** Application Programming Interface
- **PCI-DSS:** Payment Card Industry Data Security Standard
- **RTM:** Requirements Traceability Matrix
- **TLS:** Transport Layer Security

5.2 References: IEEE 42010, OWASP, NIST SP 800-160.

5.3 Tools: PlantUML, draw.io, Swagger.