Software Requirements Specification (SRS)

Project: Intelligent Ticketing and Scheduling System for Intercity Bus Services

Version: 1.0
Prepared by: Ekra Hossain (ID: 0122420004)
Date: July 4, 2025

Table of Contents

1. Introduction

1.1 Purpose

This document defines the software requirements for the Intelligent Ticketing and Scheduling System, a cloud-based Software-as-a-Service (SaaS) platform designed specifically for small and medium-sized intercity bus companies. The solution aims to offer an affordable, efficient, and scalable way to manage ticket sales, scheduling, passenger tracking, and revenue monitoring with integrated blockchain security.

1.2 Scope

The system provides:

- A white-labeled ticketing portal for each bus company
- Passenger mobile apps for real-time tracking and digital ticketing
- An operator dashboard for fleet and revenue management
- Blockchain-based fraud-proof ticketing and transparent revenue tracking
- Real-time GPS tracking and analytics to optimize routes and improve services

This SRS defines functional and non-functional requirements to guide development and ensure the system aligns with business needs.

2. Overall Description

2.1 Product Perspective

This platform is a multi-tenant SaaS solution, meaning multiple bus companies can operate independently within the same system while maintaining data and branding isolation. It is hosted on a scalable cloud infrastructure and follows microservices-based architecture.

2.2 Business Context and Problem

Small and medium-sized bus companies often cannot afford dedicated IT systems due to high upfront costs and lack of technical staff. As a result, they face:

- Inefficient manual ticketing processes
- Revenue loss due to fraud and leakages
- Lack of route planning and demand insights
- Poor passenger satisfaction due to limited services

This system solves these challenges by providing a low-cost, customizable, secure platform as a subscription service, increasing productivity and operational transparency.

3. Functional Requirements

3.1 Passenger Module

- Book tickets via the web or mobile app
- Select seats and pay online
- Receive QR code-based e-tickets
- Track bus location in real time
- View travel history

3.2 Operator Dashboard

- Manage bus schedules and seat maps
- Monitor real-time bus location
- View analytics reports on routes and sales
- Validate e-tickets via QR code scanning
- Configure company branding and pricing

3.3 Admin and System Management

- Manage tenants (bus companies)
- Control API usage and permissions
- Perform compliance and security audits
- View overall system health and logs

3.4 Ticketing and Scheduling Engine

- Check availability and pricing rules
- Assign tickets and track occupancy
- Handle peak/off-peak rates dynamically

3.5 Payment Gateway Integration

- Secure transactions via Stripe, PayPal
- Refund processing
- Automatic reconciliation with blockchain ledger

4. Non-Functional Requirements

4.1 Security

- Blockchain-backed ticket issuance and validation
- TLS encryption for data in transit; AES-256 for data at rest
- OAuth 2.0-based authentication and authorization
- Tenant data isolation enforced through schemas and smart contracts

4.2 Scalability & Performance

- Kubernetes-powered horizontal scaling
- Redis caching for quick response
- Kafka-based streaming for real-time GPS updates
- Load balancing across services

4.3 Reliability & Availability

- AWS-based high availability (multi-AZ)
- Backup and disaster recovery strategy with snapshots
- Downtime incident logging and root cause analysis

4.4 Usability

- Modern and responsive UI for web and mobile
- Multi-language support for regional customers
- Accessibility support (WCAG 2.1)

5. Architecture Overview

5.1 Software Architecture

- **Frontend:** React.js web apps; Flutter or React Native mobile apps
- **Backend:** Node.js / Spring Boot microservices
- **Infrastructure:** AWS-based with EKS, API Gateway, S3, RDS, and Lambda
- **Blockchain Layer:** Hyperledger Fabric per tenant for fraud-proof logging
- **Database:** PostgreSQL for structured data, MongoDB for analytics
- **Monitoring:** Prometheus, Grafana, ELK for logs and alerts

6. Quality and Compliance

The platform adheres to modern software quality standards and includes:

- Automated CI/CD pipeline for consistent delivery

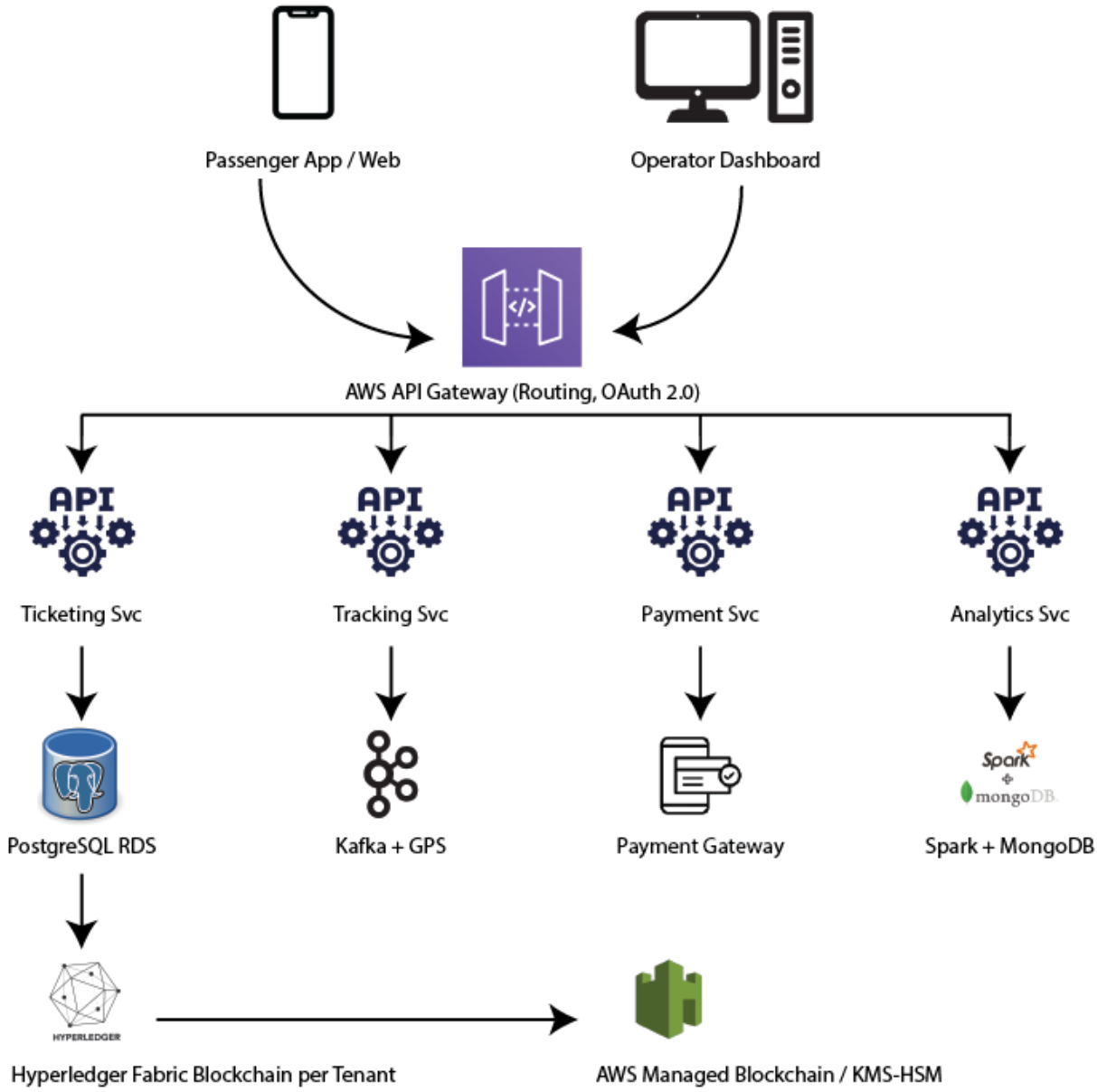
- Regression testing, unit and integration tests
- Security audits, version control, and rollback options
- API documentation and user guides for clients

7. External Interfaces

Interface Type	Description
Web Interface	White-labeled web portal for passengers
Mobile App	Cross-platform mobile app for booking and tracking
API Gateway	RESTful APIs for front-end-backend communication
Payment Gateway	Stripe/PayPal APIs for payment processing
GPS Interface	MQTT/WebSocket interface to receive live GPS data
Blockchain	Smart contract interactions for ticket and revenue logging

8. Network Architecture

8.1 Architecture Diagram



8.2 Key Network Components

Component	Description
API Gateway	Acts as a secure entry point for all client requests, enforces OAuth 2.0.
Kubernetes (EKS)	Orchestrates microservices with load balancing and scaling.

Component	Description
Microservices	Independent services for auth, scheduling, ticketing, payment, analytics.
Database Layer	PostgreSQL (relational, per tenant schemas), MongoDB (analytics).
Blockchain Layer	Hyperledger Fabric with tenant-specific channels for ticket validation.
Message Broker	Kafka for GPS data, ticket events, and microservice communication.
Cache Layer	Redis via ElastiCache for session/schedule caching.
Monitoring Stack	Prometheus, Grafana, ELK for real-time logs, metrics, and alerts.

9. Conclusion

This system is built to transform intercity bus operations through automation, transparency, and cost-effectiveness. By eliminating manual inefficiencies and introducing advanced digital features like blockchain, real-time tracking, and analytics, it empowers bus operators to grow sustainably and improve customer satisfaction — all with minimal IT overhead.