SOFTWARE DESIGN DOCUMENT (SDD)

Project: Real-Time Queue & Appointment Optimizer System

1. Introduction

This Software Design Document (SDD) presents the overall design architecture of the proposed software system. The project is currently in the design phase, and therefore this document focuses on explaining the planned structure of the system rather than implementation details. It describes the major components of the system, the relationships between these components, and the flow of data within the system.

The purpose of this document is to provide a clear and detailed understanding of how the system will be designed and how different modules will interact with each other. By defining the architecture and design decisions at an early stage, this document helps developers, reviewers, and stakeholders understand the system behaviour, ensure design consistency, and serve as a reference during the development and testing phases.

2. System Architecture

The system follows a client-server architecture model.

Clients:

- Customer Interface – allows users to join the queue
- Counter Interface – allows staff to serve customers
- Display Interface – shows live queue status

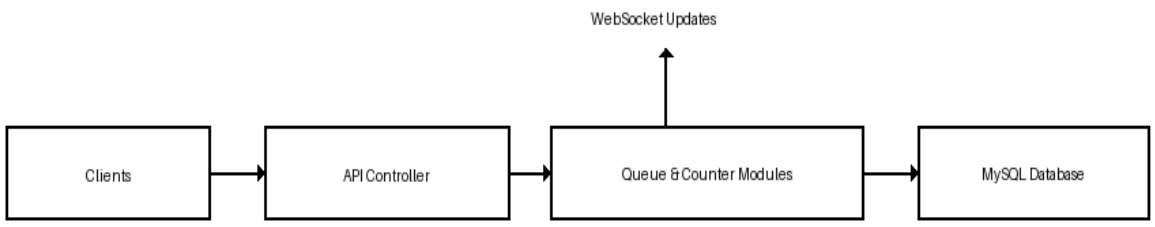
Backend:

- API Controller
- Queue Management Module
- Counter Service Module
- Real-Time Communication Module

Database:

- MySQL database for persistent data storage

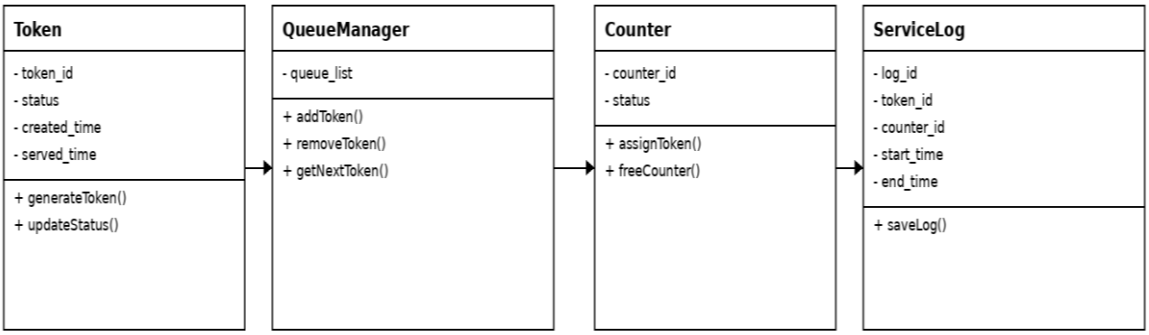
Architecture Diagram:



3. Detailed Design

The detailed design explains the internal working of each module.

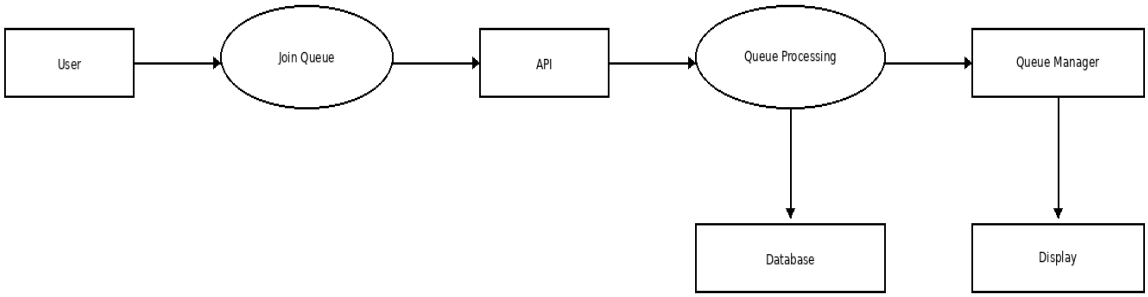
Class Diagram:



Data Flow:

Requests move from client interfaces to the backend API. The queue manager processes logic, and updates are sent back using real-time communication.

1. User joins the queue using the client interface.
2. Request is sent to the backend API.
3. Queue Manager generates and stores a token.
4. WebSocket broadcasts updated queue to all clients.
5. Counter becomes free and requests next token.
6. Queue Manager assigns next token automatically.



Logic Description:

Tokens are assigned in a sequential order based on arrival. When a service counter becomes free, the system automatically assigns the next available token from the queue. The queue status is then updated and reflected on all clients in real time.

4. Database Design

The database uses a relational model implemented using MySQL.

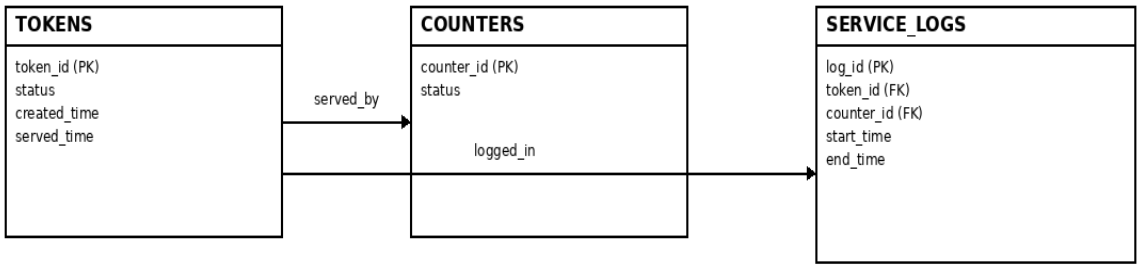
Entities:

- Tokens – stores token number, status, and timestamps
- Counters – stores counter ID and availability status
- Service Logs – stores service history

Relationships:

- Each token is served by one counter
- Each counter serves multiple tokens over time

ER Diagram:



5. Design Constraints

Hardware Constraints:

- Must run on standard desktop or server hardware
- No specialized hardware required

Software Constraints:

- Backend developed using Python and FastAPI
- MySQL used for database management
- WebSocket support required for real-time updates

Regulatory Constraints:

- Basic data privacy guidelines must be followed
- Secure handling of user data