OPD Token Allocation System – Backend Design Report

1. Introduction

The OPD Token Allocation System is a backend service designed to manage patient tokens in a hospital OPD based on time slots, doctor availability, and patient priority. The aim of this system is to simulate real-world OPD behavior such as limited slot capacity, emergency handling, cancellations, and no-shows while keeping the design simple and efficient.

This project focuses on backend logic and algorithmic correctness rather than UI or database complexity.

---

2. System Overview

The system consists of:

- Doctors with predefined time slots
- Each slot having a fixed maximum capacity
- Tokens generated for patients based on their source and priority
- Priority-based replacement if a slot is already full
- APIs for token creation, cancellation, emergency insertion, and slot status
- A simulation module to demonstrate a full OPD day

All data is stored in-memory for simplicity.

---

3. Token Priority Strategy

Each token is assigned a priority based on the patient source:

Source	Priority
Emergency	1 (Highest)
Paid	2
Follow-up	3
Online	4
Walk-in	5 (Lowest)

Lower number means higher priority.

This ensures that:

- Emergency patients are always treated first
- Paid and follow-up patients are given preference

- Walk-in patients are considered only if capacity is available

---

4. Slot Allocation Logic

When a new token request comes:

1. The system checks if the doctor exists
2. It checks if the slot exists
3. It checks the slot capacity
4. If the slot has space:
    - Token is directly added
5. If the slot is full:
    - The system finds the lowest priority token in the slot
    - If the new token has higher priority:
        - The lower priority token is removed
        - The new token is inserted
    - Otherwise:
        - The request is rejected

This simulates real OPD behavior where critical patients replace less urgent ones.

---

5. Emergency Handling

Emergency tokens:

- Always use the highest priority (1)
- Can replace any existing token in a full slot
- Are inserted immediately into the system

This guarantees emergency patients are never blocked by normal bookings.

---

6. Cancellation and No-Show Handling

Two special cases are handled:

1. **Cancellation**
    - Token status is marked as CANCELLED
    - Token is removed from the slot

2. **No-Show**
    - Token status is marked as NO_SHOW
    - Token is removed from the slot

Both cases free up space in the slot for future patients.

---

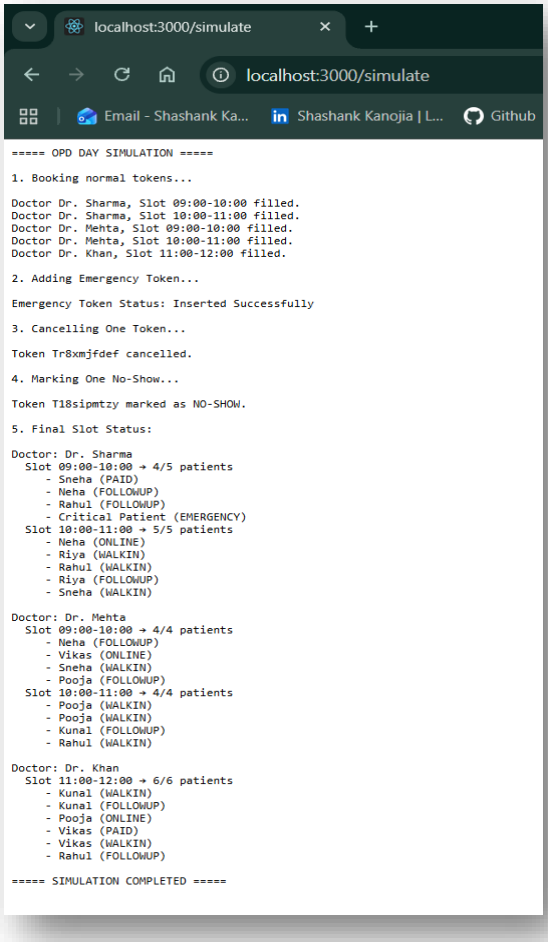
7. Simulation

A simulation module was created to represent a full OPD day:

- Normal tokens are booked until slots are full
- Emergency token is added
- One token is cancelled
- One token is marked as no-show
- Final slot status is printed

The simulation is shown in the browser at: http://localhost:3000/simulate

The following is the screenshot as proof of execution :



8. Design Decisions

Decision	Reason
In-memory storage	Keeps system simple and focused on logic
No database	Assignment is logic-focused, not infrastructure-focused
Express.js	Lightweight and suitable for backend APIs
Priority replacement	Closest to real hospital OPD behavior
No authentication	Outside scope of the assignment

9. Limitations

- Data resets when the server restarts
- No persistent storage
- Not optimized for concurrent users
- No frontend interface

These limitations are acceptable since the goal is algorithm and backend logic demonstration.

---

10. Conclusion

This project successfully demonstrates:

- Slot capacity enforcement
- Priority-based allocation
- Emergency handling
- Token cancellation and no-show handling
- API design and modular backend architecture
- Realistic OPD workflow simulation

It reflects practical backend problem-solving and aligns well with real-world hospital OPD token systems.

Github link : https://github.com/ShashankKan0jia/opd-token-engine-Medoc-Health

Made by :

Shashank Kanojia

Email : kanojiashashank87@gmail.com