OPD Token Allocation Engine Backend Intern Assignment – Technical Documentation

1. Project Overview

The OPD Token Allocation Engine is a backend service designed to manage patient token allocation for hospital OPDs. The system supports multiple doctors, fixed consultation slots, multiple patient sources, and real-world scenarios such as cancellations, no-shows, priority insertions, and concurrency handling.

2. Architectural Design

The application follows a layered architecture: Controller → Service → Token Engine → Repository → Database. Each layer has a well-defined responsibility. Controllers expose REST APIs, services orchestrate workflows, the TokenEngine contains the core allocation logic, and repositories handle persistence and concurrency.

3. Core Business Rules

- One token per patient per doctor per day. - Slots have a hard maximum capacity. - Tokens are prioritized by source. - Higher-priority tokens may preempt lower-priority ones. - Capacity is never exceeded under any circumstance.

4. Token Priority Model

Priority is calculated dynamically using a PriorityCalculator based on token source. Typical priority order: Emergency > Paid > Follow-up > Walk-in > Online Higher priority tokens are allowed to displace lower priority tokens.

5. Token Allocation Algorithm

When a token request is received, the system: 1. Identifies or creates the patient using phone number. 2. Enforces the one-token-per-day rule. 3. Searches for an available slot. 4. Allocates directly if capacity exists. 5. Attempts preemption if slot is full. 6. Adds the patient to the waitlist if allocation is not possible.

6. Preemption Logic

If no slots are available, the system identifies the lowest-priority token for the doctor on the same day. If the incoming token has a higher priority, the existing token is cancelled and the slot is reassigned.

7. Waitlist Management

When allocation is not possible, patients are added to a waitlist with priority and timestamp. Waitlisted patients are promoted automatically when a slot becomes available.

8. Cancellation Handling

When a token is cancelled: - Token status is updated to CANCELLED. - Slot capacity is reduced. - Slot is marked OPEN. - The highest priority waitlisted patient is promoted.

9. No-Show Handling

Patients are marked NO_SHOW only after a 15-minute grace period from slot start time. Once marked: - Slot capacity is freed. - Waitlist promotion is triggered.

10. Concurrency & Safety

The system uses transactional boundaries and database-level locking for token number generation. This prevents race conditions and ensures consistent allocation under concurrent requests.

11. Reporting APIs

The system provides paginated APIs to fetch booked tokens by date range. This supports operational dashboards and reporting needs.

12. Edge Cases Handled

- Duplicate token requests - Slot overbooking prevention - Priority inversion - Emergency insertions - Concurrent allocation requests - Stale waitlist entries

13. Trade-offs & Design Decisions

Fairness was intentionally sacrificed in favor of strict priority enforcement to reflect real hospital workflows. Overbooking was avoided to ensure predictability and prevent cascading delays.

14. Challenges Faced

Key challenges included handling dynamic reallocation safely, preventing race conditions, managing waitlists without duplication, and maintaining deterministic behavior under real-time constraints.

15. Conclusion

The OPD Token Allocation Engine provides a robust, scalable, and realistic solution for OPD scheduling. It enforces strict constraints while remaining flexible enough to handle real-world hospital scenarios.