Backend Completion - Report

Cab Booking Management System

---

1. Introduction

This report certifies the successful completion of the **backend development** for the *Cab Booking Management System*.

The backend has been designed, implemented, and validated to support all core business workflows required for a functional cab booking platform.

The system addresses real-world use cases involving **individual customers, corporate customers, vendors, and drivers**, ensuring correct handling of bookings, payments, notifications, and lifecycle management.

---

2. Objective of the Backend System

The primary objectives of the backend were:

- To implement a **robust booking lifecycle** from request to completion
- To support **role-based access control** for customers, vendors, and drivers
- To differentiate logic between **individual and corporate customers**
- To ensure **data integrity, validation, and auditability**
- To provide a **scalable MVP architecture** suitable for academic and demo use

All objectives have been fully met.

---

3. System Architecture Overview

The backend follows a **modular REST API architecture**, implemented using:

- **Node.js + Express.js** for API services
- **PostgreSQL** as the relational database
- **JWT-based authentication** for secure access
- **Role-based middleware** for authorization
- **Cron jobs** for time-based booking state transitions

The architecture cleanly separates concerns into:

- Controllers
- Routes
- Middleware
- Jobs (background tasks)
- Utility modules

This ensures maintainability and extensibility.

---

4. User Roles and Responsibilities

4.1 Customer

- Create bookings (individual or corporate)
- View booking history
- Cancel bookings (when allowed)
- Perform payments (online or cash-based rules)

4.2 Vendor

- View prioritized booking requests
- Accept or reject bookings
- Assign drivers and vehicles
- Manage booking fulfillment

4.3 Driver

- View assigned trips
- Start trips (with payment validation for corporate bookings)
- End trips and complete journeys

Each role is strictly enforced using authorization middleware.

---

5. Booking Lifecycle Implementation

The backend correctly implements the **complete booking lifecycle**:

1. Booking creation
2. Vendor review and prioritization
3. Acceptance or rejection
4. Driver and vehicle assignment
5. Trip start
6. Trip completion
7. Payment handling
8. Notifications and audit logging

All state transitions are validated to prevent illegal or inconsistent operations.

---

6. Individual vs Corporate Booking Logic

The system includes **advanced business logic** to differentiate booking behavior:

Individual Customers

- Can choose **cash or online payment**
- Can start trips without pre-payment
- Payment is finalized after trip completion

Corporate Customers

- Forced **online payment only**
- Must complete payment **before trip starts**
- Booking priority increases automatically when target time is near

This logic is dynamically enforced without manual intervention.

---

7. Priority Handling and Vendor Visibility

The backend dynamically computes booking priority using SQL logic:

- **High Priority**
    - Corporate bookings within 3 hours of target time
- **Normal Priority**
    - Regular bookings
- **Low Priority**
    - Individual bookings when urgent corporate bookings exist

This ensures vendors always see the **correct booking order** without modifying stored data.

---

8. Payment Management

The payment system supports:

- Individual online payments
- Individual cash payments
- Mandatory corporate online payments

Strict routing rules ensure:

- Corporate bookings cannot use individual payment endpoints
- Invalid payment flows are blocked
- Payment status is always consistent with booking state

---

9. Notifications and Audit Logging

Notifications

- Booking acceptance
- Driver assignment
- Trip completion
- Payment instructions

Audit Logs

- Every critical state transition is recorded:
    - Who performed the action
    - Role of the user
    - Previous and new status
    - Timestamp

This provides **traceability and accountability**, essential for production-quality systems.

---

10. Data Integrity and Validation

The backend enforces strong integrity through:

- Database constraints and checks
- Controlled state transitions
- Ownership validation
- Availability checks for drivers and vehicles

No invalid or inconsistent records can be created through the API.

---

11. Testing and Verification

All major scenarios have been verified, including:

- Individual booking flow
- Corporate booking flow
- Priority conflicts
- Vendor acceptance and rejection
- Driver trip start and end
- Payment confirmation
- Notification delivery
- Audit log creation

All tests confirm expected outcomes.

---

12. Scope of Completion (MVP)

The backend fully satisfies the **Minimum Viable Product (MVP)** scope:

✅ Core business logic

✅ All user roles

✅ Booking lifecycle

✅ Payments

✅ Notifications

✅ Audit logging

✅ Data integrity

No critical features are missing.

---

13. Future Enhancements (Out of Scope)

The following are **optional enhancements**, not required for MVP:

- Real-time tracking (WebSockets)
- Maps and distance calculation
- Payment gateway integration
- Admin analytics dashboard
- Push notifications

These can be added later without changing the core backend.

---

14. Final Certification Statement

This report certifies that the backend implementation of the Cab Booking Management System is complete, stable, and production-quality for academic and demonstration purposes.
All MVP requirements have been fully implemented and validated.