International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

# Omsai Tour and Travels: Smart Tourism and School Van Management System

*A Web and Mobile Based Integrated Platform for Tour Booking and School Transportation Management*

**Shubham Yashwant Nayakwadi**

nayakwadi2001@gmail.com

Department of Computer Science, M.Sc. Computer Science

Roll No. 25 | Academic Year 2026–2027

---

**ABSTRACT**

Digitalization has changed the way tourism services and transportation operations are planned, booked, monitored, and communicated. Small tour operators frequently depend on telephone calls, messaging applications, notebooks, spreadsheets, and manually maintained booking records. School transportation may similarly depend on paper attendance, informal parent-driver communication, and limited visibility of vehicle movement. This paper proposes "Omsai Tour and Travels: Smart Tourism and School Van Management System", an integrated web and mobile platform designed to manage tour packages, customer bookings, school students, parents, drivers, vehicles, routes, pickup/drop status, notifications, and location-aware van monitoring.

The proposed architecture separates the presentation, application, and data layers. A React.js web client and a Flutter-based mobile application communicate with a Java Spring Boot REST API. MySQL provides centralized relational storage, while JWT-based authentication and role-based authorization control access. GPS/location services can transmit periodic vehicle coordinates, and notification services can inform authorized parents and administrators about important trip events. The system also provides an administrative dashboard for operational monitoring and reporting.

The research follows an applied software engineering methodology involving requirement analysis, review of existing approaches, system modelling, database design, prototype implementation, and evaluation planning. The uploaded reference paper uses a progression from problem motivation and related work to research gap, proposed framework, methodology, implementation, evaluation, limitations, and conclusion; this paper follows the same research pattern while developing original content for the Omsai project. The reference emphasizes separating proposed evaluation from completed experimental results.

***Keywords***-*smart tourism, tour management, school van management, mobile application, GPS tracking, Spring Boot, React.js, Flutter, MySQL, REST API, route management.*

## 1. INTRODUCTION

Tourism and local transportation are information-intensive activities. Customers require package details, prices, schedules, vehicle information, and booking confirmations, while operators need accurate records of customers, vehicles, routes, payments, and schedules. When these records are maintained across disconnected channels, information can become duplicated, outdated, or difficult to retrieve.

School transportation adds another coordination problem. Parents need timely information about pickup and drop activities; drivers require an organized route and student list; and administrators need centralized records of vehicles, drivers, students, routes, and trip history. Location-aware mobile services provide an opportunity to make this communication more systematic.

The proposed Omsai Tour and Travels system combines these two service domains into one modular platform. The tourism component manages travel packages and bookings, while the school van component manages students, parents, drivers, vehicles, routes, trip status, and optional GPS tracking. A common backend, database, authentication layer, and notification mechanism reduce duplication while keeping business workflows separate.

The main contributions are:

- Integrated architecture for tourism booking and school van management.
- Role-specific workflows for customers, parents, drivers, and administrators.
- Centralized relational data model for users, bookings, students, vehicles, routes, and trips.
- REST API communication between web/mobile clients and backend.
- Optional GPS and notification layer for school transportation monitoring.
- Evaluation framework covering functional correctness, performance, usability, security, and data integrity.

## 2. RELATED WORK

### 2.1 Digital Tourism and Online Booking

Online tourism systems commonly provide destination information, package discovery, availability, customer registration, reservation, and confirmation. Digital booking reduces dependence on telephone-based coordination and creates searchable historical records. Small operators may still use multiple disconnected tools for customer enquiries, vehicle allocation, payment tracking, and reporting. The proposed system treats package, customer, vehicle, and booking information as related records in one database.

### 2.2 School Transportation Management

School transportation applications commonly organize students, guardians, drivers, vehicles, routes, stops, and trip events. A location-enabled application can provide the current or last-known position of an assigned vehicle. Controlled visibility is essential: parents should see information relevant to their child and assigned route, while administrators require broader operational access.

### 2.3 Location-Based Services

GPS-enabled mobile devices can obtain geographic coordinates and transmit them to a server. A transportation application can store timestamped coordinates and present a vehicle's latest known position on a map. For an academic prototype, periodic updates are simpler to implement and evaluate than unrestricted continuous background tracking.

### 2.4 Mobile and Web Architecture

A separate frontend and backend architecture allows multiple clients to reuse the same business services. A web client can serve administrators and customers, while a mobile client can serve parents and drivers. REST APIs provide a standardized communication layer and make additional clients easier to add later.

### 2.5 Research Gap

The research gap addressed here is not the absence of tourism booking systems or school transportation systems individually. It is the lack of a single academic platform designed around the combined requirements of a local tour operator and school van service. The project investigates how shared identity, vehicle, route, notification, and reporting services can support both domains without mixing their operational rules.

| Area | Existing Approach | Proposed Gap Addressed |
|---|---|---|
| Tour booking | Package and reservation management | Integrated vehicle/service records |
| School transport | Student/route tracking | Parent, driver and admin workflow |
| Communication | Calls/messages/manual updates | Centralized status and notifications |
| Data | Separate files/apps | Centralized relational database |
| Clients | Single interface | Web + role-specific mobile clients |
| Monitoring | Manual reports | Dashboard and structured trip history |

## 3. PROPOSED SYSTEM

### 3.1 System Overview

The system is organized as a multi-client service platform. Customers use tourism services through web/mobile interfaces. Parents use school transportation functions, drivers use mobile workflows for assigned trips, and administrators use a dashboard to manage both service areas. Protected requests pass through authentication and authorization.

**[FIGURE: Figure1.OverallSystemArchitecture]** *Customer / Parent / Driver / Admin → React Web / Flutter Mobile → Spring Boot REST API → MySQL; optional Maps/GPS and Notification services.*

### 3.2 Major Components

| Component | Responsibilities |
|---|---|
| Web Frontend | Tour pages, booking, customer account, admin dashboard, reports |
| Mobile App | Parent and driver workflows; optional customer access |
| Authentication | Registration, login, token validation, role permissions |
| Tour Module | Packages, destinations, schedules, bookings |
| School Module | Students, parents, drivers, vehicles, routes, stops, trip status |
| Tracking | Periodic GPS coordinates and latest vehicle position |
| Notifications | Booking confirmation and pickup/drop/trip events |
| Database | Persistent storage and relationships |
| Admin Module | Master data, monitoring and reports |

### 3.3 User Roles

- Customer: registers, browses packages, books tours and views booking history.
- Parent: views assigned child, route/trip status and authorized location information.
- Driver: views route/students, starts trips, updates pickup/drop status and shares location.
- Administrator: manages users, tours, bookings, students, drivers, vehicles, routes and reports.

## 4. METHODOLOGY

### 4.1 Problem Formulation

Let U represent users, P tour packages, B bookings, S students, V vehicles, D drivers, R routes, T trips, and L location records. The system receives requests through authenticated clients and processes them using role-specific business rules. The principal objective is to maintain consistent records and expose only authorized information.

*Simplified flow: Request → Authentication → Authorization → Validation → Business Logic → Database/External Service → Response.*

### 4.2 Requirement Analysis

| Type | Requirement |
|---|---|
| Functional | Registration and login |
| Functional | Tour package creation, update and search |
| Functional | Tour booking and cancellation/status management |
| Functional | Student, parent, driver and vehicle management |
| Functional | Route and pickup-point management |
| Functional | Trip start/end and pickup/drop status |
| Functional | Location updates and authorized tracking |
| Functional | Notifications and reports |
| Non-functional | Security and role-based access |
| Non-functional | Responsive UI and mobile usability |
| Non-functional | Data integrity and validation |
| Non-functional | Maintainability and modularity |

### 4.3 Database Methodology

MySQL is proposed as the relational database. Primary and foreign keys represent entity relationships. Master data such as users, students, vehicles, routes and packages is separated from operational records such as bookings, trips, tracking events and notifications.

| Table | Important Fields |
|---|---|
| users | user_id, name, email, phone, password_hash, role |
| students | student_id, name, class, parent_id, pickup_point_id |
| parents | parent_id, user_id, relationship |
| drivers | driver_id, user_id, license_no |
| vehicles | vehicle_id, vehicle_number, capacity, status |
| routes | route_id, route_name, school_id |
| pickup_points | point_id, route_id, latitude, longitude, sequence_no |
| tour_packages | package_id, destination, duration, price, description |
| bookings | booking_id, user_id, package_id, travel_date, status |
| trips | trip_id, route_id, vehicle_id, driver_id, start_time, end_time, status |
| tracking | tracking_id, trip_id, latitude, longitude, timestamp |
| student_trip_status | trip_id, student_id, pickup_status, drop_status |
| notifications | notification_id, user_id, type, message, read_status |

### 4.4 Backend Methodology

Java Spring Boot is proposed for server-side implementation. Controllers expose REST endpoints, services contain business rules, repositories communicate with MySQL, and security components validate protected requests. DTOs can separate API payloads from database entities and centralized exception handling can provide consistent error responses.

Example API groups:

- /api/auth – registration and authentication
- /api/tours – package management
- /api/bookings – booking creation and history
- /api/students and /api/parents – school records
- /api/drivers and /api/vehicles – transport resources
- /api/routes and /api/trips – routes and trips
- /api/tracking – location updates
- /api/notifications – event messages
- /api/admin – dashboards/reports

### 4.5 Frontend and Mobile Methodology

React.js is proposed for the responsive web interface, while Flutter can provide a cross-platform mobile application. Both clients consume the same REST APIs so that authentication, validation, business rules and database operations remain centralized.

### 4.6 GPS and Notification Methodology

When tracking is enabled, the driver application obtains a location coordinate at an approved interval and submits it with a trip identifier and timestamp. The server validates the driver's authorization before storing it. Authorized parent clients request the latest position. Notifications can be triggered when a trip starts or when pickup/drop events are recorded.

### 4.7 Security Methodology

- Use secure password hashing rather than plain-text passwords.
- Use JWT or secure sessions for protected API requests.

- Apply role-based authorization.
- Validate input at client and server layers.
- Prevent cross-parent access to protected student records.
- Define retention rules for location and student information.
- Maintain audit logs for important administrative actions.

## 5. SYSTEM IMPLEMENTATION

### 5.1 Technology Stack

| Layer | Technology |
|---|---|
| Web frontend | React.js, HTML5, CSS3, JavaScript |
| Mobile | Flutter / Dart |
| Backend | Java, Spring Boot, Spring MVC |
| Data access | Spring Data JPA / Hibernate |
| Database | MySQL |
| Authentication | Spring Security + JWT |
| API | REST / JSON |
| Maps/location | GPS + selected map/routing API |
| Notifications | Firebase Cloud Messaging or equivalent |
| Version control | Git / GitHub |

### 5.2 Procedural Flow

**[FIGURE: Figure 2. Tour Booking Flow]** *Login → Browse Packages → Select Package → Date/Travellers → Validate → Confirm/Payment → Booking → Notification → History.*

**[FIGURE: Figure 3. School Van Flow]** *Driver Login → Assigned Route → Start Trip → Location Updates → Pickup Student → Next Stop → School → Drop Student → End Trip.*

### 5.3 Use Case Design

| Actor | Use Cases |
|---|---|
| Customer | Register, Login, Browse Tours, Book, Cancel/View Booking |
| Parent | Login, View Child, Route, Trip, Notifications |
| Driver | Login, Route, Start Trip, Pickup/Drop, Location, End Trip |
| Admin | Manage users, tours, bookings, students, drivers, vehicles, routes and reports |

### 5.4 E-R Design

**[FIGURE: Figure 4. Conceptual E-R Design]** *USER–BOOKING–TOUR_PACKAGE; PARENT–STUDENT; DRIVER–VEHICLE; ROUTE–PICKUP_POINT; ROUTE–TRIP–VEHICLE; TRIP–STUDENT through STUDENT_TRIP_STATUS; TRIP–TRACKING.*

### 5.5 Data Flow Design

At context level, customers, parents, drivers and administrators exchange data with the Omsai system. The system exchanges data with the database and, when enabled, map and notification services. Level 1 processes are authentication, tourism management, school transport management, tracking, notifications and reporting.

### 5.6 Sequence Design

1. Driver authenticates with the mobile application.
2. Application requests an access token.
3. Driver starts the assigned trip.
4. Mobile device obtains coordinates.
5. Coordinates are sent to the tracking API.
6. Backend validates driver/trip authorization.
7. Location is stored.
8. Authorized parent request's latest location.
9. Backend returns authorized location and status.

10. Mobile application displays the trip status.

## 6. EVALUATION METHODOLOGY

The reference paper distinguishes planned evaluation from completed experimental results. This project follows the same principle: the following metrics are an evaluation plan and should only be reported as measured results after implementation and testing.

### 6.1 Functional Testing

| Test Area | Metric |
|---|---|
| Authentication | Successful/failed login cases |
| Tour booking | Valid/invalid booking cases |
| Student management | Create/update/delete validation |
| Trip management | Trip start/end/status correctness |
| Tracking | Accepted/rejected location updates |
| Authorization | Unauthorized access attempts |
| Notifications | Correct event triggering/delivery |

### 6.2 Performance Metrics

- API response time
- Database query response time
- Tracking processing latency
- Mobile screen loading time
- Controlled concurrent request handling

### 6.3 Usability and Reliability

Usability can be assessed by representative users performing predefined tasks. Reliability can be assessed through repeated booking, login, trip and API tests. Failures and corrective actions should be documented instead of claiming improvements without measurements.

### 6.4 Security Evaluation

Testing can include authentication, authorization, invalid-input handling, token/session management, password-storage verification, and checks that one parent cannot access another student's protected records. OWASP guidance can be used as a general security checklist.

## 7. LIMITATIONS AND FUTURE WORK

### 7.1 Limitations

- GPS depends on permissions, connectivity, battery and server availability.
- Map/notification services may have quotas or costs.
- Prototype data may not represent production scale.
- Advanced route optimization is outside the minimum implementation.
- Payment may initially be simulated.
- Student and location data require careful privacy and access control.
- Production deployment requires operational and legal review.

### 7.2 Future Work

- AI-assisted route optimization
- ETA prediction using historical trip data
- QR/NFC student verification
- Multi-school and multi-branch support
- Analytics for package demand, revenue and vehicle utilization
- Vehicle maintenance reminders
- English/Marathi/Hindi interfaces
- Cloud deployment and monitoring
- Advanced communication integrations
- Privacy-preserving location-data retention

## 8. CONCLUSION

This paper proposed Omsai Tour and Travels: Smart Tourism and School Van Management System, an integrated web and mobile platform for tourism booking and school transportation management. The system connects customers, parents, drivers and administrators through centralized services and a relational database.

The proposed architecture uses React.js for the web interface, Flutter for mobile clients, Java

Spring Boot for REST services and MySQL for structured storage. GPS and notification services extend the school transportation module toward a location-aware smart service. The project demonstrates M.Sc. Computer Science concepts including requirements engineering, system architecture, database modelling, API development, mobile computing, authentication, security and evaluation.

At the proposal stage, no production-level improvement is claimed. The next step is to implement the prototype, prepare test data, execute functional and performance tests, collect usability feedback, and report measured results. This distinction keeps proposed design separate from experimental evidence, consistent with the reference research-paper approach.

**ACKNOWLEDGMENT-**

I thank Ms. Ahinsa Gaikwad, Department of Computer Science, Patkar-Varde College, for guidance on the project that this paper condenses, and the college for laboratory access. We also thank the Research Methodology faculty for the chapter structure that separated problem, method, and deferred work.

**REFERENCES**

[1] P. Lewis et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” NeurIPS, 2020.
[2] Spring, “Spring Boot Reference Documentation.”
[3] Spring, “Spring Security Reference Documentation.”
[4] React, “React Documentation.”
[5] MySQL, “MySQL Documentation.”
[6] Flutter, “Flutter Documentation.”
[7] OWASP Foundation, “OWASP Top 10: The Ten Most Critical Web Application Security Risks.”
[8] Object Management Group, “Unified Modeling Language (UML) Specification.”
[9] R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner’s Approach, McGraw-Hill.
[10] I. Sommerville, Software Engineering, Pearson.
[11] R. Elmasri and S. B. Navathe, Fundamentals of Database Systems, Pearson.