Journey DNA: A Smart, Personalized Travel Planning and Bus Fleet Discovery Platform

NISHANTH S
[Student 1]
Dept. of Computer Science & Engg.
Sri Shakthi Inst. of Engg. & Tech.
Coimbatore, Tamil Nadu, India

NISHANTH C
[Student 2]
Dept. of Computer Science & Engg.
Sri Shakthi Inst. of Engg. & Tech.
Coimbatore, Tamil Nadu, India

MAHASAKTHI S
[Student 3]
Dept. of Computer Science & Engg.
Sri Shakthi Inst. of Engg. & Tech.
Coimbatore, Tamil Nadu, India

NISHANTH R K
[Student 4]
Dept. of Computer Science & Engg.
Sri Shakthi Inst. of Engg. & Tech.
Coimbatore, Tamil Nadu, India

Corresponding Author: Nishanth C | Email: nishanthsella2007cn@gmail.com / [student2.cse@siet.ac.in]

ABSTRACT

Modern leisure travel and collective group excursions are frequently hindered by severe information fragmentation, compelling travelers to juggle disconnected mapping tools, unverified tourist directories, private fleet operators, and manual spreadsheets. This paper introduces **Journey DNA**, an integrated, location-aware travel planning and charter bus discovery web platform designed to unify destination cataloging, spatial route synthesis, and private transport booking into a cohesive cyber-physical architecture. Engineered with a lightweight client-server model utilizing HTML5, modern CSS3, asynchronous JavaScript with Leaflet GIS mapping, a Python Flask micro-framework backend, and a transactional SQLite relational database, Journey DNA provides an end-to-end itinerary ecosystem. The system incorporates a 28-category spatial discovery engine computing real-time straight-line and detour metrics via the Haversine formulation, an automated passenger-capacity-to-bus classification algorithm, a dedicated hidden-gems discoverer, upcoming cultural event tracking, and an administrative governance dashboard enforced through session-based Role-Based Access Control (RBAC). Empirical evaluation and automated test suites confirm 100% compliance across authentication, spatial filtering, fleet reservation workflows, and data integrity without relying on costly closed-source platform dependencies. The proposed platform significantly diminishes cognitive friction and planning overhead, advancing the practical democratization of smart tourism ecosystems.

Keywords **-** Smart Tourism, Personalized Travel Planning, Web Application, Geographic Information Systems (GIS), Tourist Destination Discovery, Python Flask, SQLite, Bus Fleet Discovery, Travel Itinerary Management, Haversine Spatial Filtering.

1. INTRODUCTION

The digital revolution within the global tourism industry has fundamentally altered how modern consumers explore geographic regions, formulate itineraries, and secure transportation [1]. Concurrently, the proliferation of Geographic Information Systems (GIS), mobile location-based services (LBS), and interactive web technologies has fostered high consumer expectations for instant, hyper-localized destination intelligence [2]. Despite the maturity of commercial booking portals, individuals and collective group travelers (such as college student tours, corporate retreats, and extended family pilgrimages) continue to confront a highly fragmented planning landscape [3].

In customary travel planning, users are burdened with context-switching among multiple disjointed software applications: destination review blogs for sightseeing research, proprietary satellite map platforms for distance appraisal, local bus directories or informal telephone contacts for charter coach quotes, and manual note-taking tools for schedule consolidation. This siloed operational paradigm introduces substantial cognitive load, fosters scheduling miscalculations, and obscures transparent pricing and vehicle safety verification [4]. Group travel logistics present an even steeper hurdle, as charter bus fleets possess intricate operational constraints, varying seating capacities, and non-standardized reservation protocols.

To resolve these operational barriers, this research presents **Journey DNA**, an integrated smart web application designed to synthesize tourist destination discovery, interactive geographic visualization, multi-destination itinerary composition, and verified bus fleet reservation. Developed using open-standard technologies - HTML5, CSS3, ES6 JavaScript, Python Flask, and SQLite - the platform establishes a consolidated hub where users can explore regional highlights, calculate detour overhead, match group headcounts against certified vehicle classifications, and track booking lifecycle states seamlessly.

2. PROBLEM STATEMENT

Contemporary digital tourism architectures remain plagued by functional decoupling between destination discovery and charter transit acquisition. Commercial travel aggregators emphasize fixed point-to-point airline or intercity scheduled bus tickets; they largely neglect private charter fleets and custom multi-stop leisure circuits. Consequently, group excursion organizers encounter four critical systemic failures:

1. Information Asymmetry: Inadequate vehicle specifications, lack of verified onboard photos/videos, and hidden intermediary markups.

2. Disjointed Spatial Planning: Difficulty assessing whether intermediate points of interest (POIs) such as waterfalls, village temples, or scenic viewpoints can be incorporated into an ongoing road route without incurring excessive detour penalties.

3. Absence of Unified Itinerary State Management: Disconnection between the traveler's planned sightseeing list and the transport provider's contract.

4. Heavy Computational Dependencies: Modern travel portals frequently mandate commercial proprietary API keys with restrictive billing quotas, hindering deployment for academic, municipal, and regional travel collectives.

Journey DNA explicitly addresses these deficiencies by establishing a unified, self-contained open-stack software architecture.

3. OBJECTIVES

The overarching engineering and research objectives of the Journey DNA initiative comprise:

- **Centralized POI Cataloging:** Establish a verified database of regional tourist destinations spanning major heritage sites, nature spots, and curated hidden gems with structured operational metadata.
- **Interactive GIS Visualization:** Implement a resilient, responsive map interface that synchronizes dynamically with destination cards, supporting zero-blank-canvas rendering across desktop and mobile form factors.
- **Live Spatial Proximity Discovery:** Develop a 28-category spatial discovery engine capable of operating upon live GPS coordinates or manual search hubs with real-time Haversine distance computations.
- **Automated Fleet Capacity Matching:** Formulate an algorithmic heuristic matching passenger passenger counts to suitable bus vehicle archetypes (e.g., Tempo Traveler, Mini Bus, Deluxe Coach, Luxury Multi-Axle).
- **Verified Operator Showcase:** Construct a dedicated Bus Directory hosting authentic fleet specifications, amenities, licensing credentials, and high-resolution multimedia tours.
- **Robust Administrative Governance:** Provide session-authenticated administrative tooling for itinerary auditing, fleet moderation, and real-time trip status lifecycle tracking.

4. LITERATURE REVIEW

Academic discourse surrounding smart tourism ecosystems emphasizes the convergence of physical tourism infrastructure with advanced ICT platforms. Buhalis and Amaranggana [1] examined smart tourism destinations, establishing that context-awareness and dynamic personalization are paramount for enriching traveler satisfaction. They posited that smart destinations must leverage real-time spatial data to interconnect tourists with local service providers. Gretzel et al. [2] expanded upon this theoretical foundation, delineating smart tourism as a cyber-physical system integrating data collection, processing, and service delivery layers. Their research identified that the democratization of tourism tech relies upon open, interoperable architectures.

Algorithmic tour synthesis and personalized itinerary recommendation have been extensively explored in spatial computing literature. Lim et al. [3] surveyed tour recommendation models utilizing location-based social networks, illustrating that greedy heuristics and orienteering problem formulations successfully balance user preferences against temporal and spatial budget constraints. De Choudhury et al. [4] investigated automatic itinerary construction from geospatial breadcrumbs, underscoring the necessity of calculating detour penalties when sequencing multi-stop excursions. Xiang et al. [5] demonstrated that travelers perceive higher utility when mapping interfaces provide immediate visual anchoring alongside descriptive metadata.

In the domain of passenger transport logistics, Werthner and Klein [6] provided classical taxonomies illustrating that independent charter carriers are historically disenfranchised in global distribution systems (GDS) due to high connectivity fees. Sinnott [7] formulated the computational virtues of the Haversine trigonometric equation, proving its numerical stability for short-to-medium range spherical geodesic calculations. Journey DNA synthesizes these foundational insights: it operationalizes Sinnott's spatial formula for real-time proximity discovery, embodies Gretzel's architectural principles using open-standard web technologies, and establishes an equitable direct-discovery bridge for regional charter bus fleets.

5. EXISTING SYSTEM & COMPARATIVE ANALYSIS

Current travel planning practices predominantly rely on heterogeneous, commercially isolated platforms. Users frequently consult search engines and crowd-sourced rating sites (e.g., TripAdvisor, Google Reviews) for discovery, transition to dedicated routing applications (e.g., Google Maps) for navigation, and subsequently communicate with local travel agents or browse intercity bus aggregators (e.g., RedBus) for transport. Crucially, scheduled intercity ticket aggregators operate on rigid route inventories; they do not cater to customized group hire where the vehicle remains chartered for the tour's entirety. Table I provides a structured comparative analysis contrasting legacy multi-app workflows with Journey DNA.

Evaluation Dimension	Conventional Fragmented Workflow	Journey DNA Proposed Platform
Destination Discovery	Dispersed across promotional blogs, third-party review sites, and social media; prone to outdated information.	Centralized verified database of regional destinations, nature spots, and curated hidden gems with structured timings.
GIS & Map Integration	Separate navigation software; pins must be manually re-entered or exported across applications.	Embedded responsive Leaflet GIS engine; bi-directionally synchronized with live cards and cluster markers.
Charter Fleet Discovery	Unregulated offline brokerage, classified telephone ads, lack of vehicle spec and interior photo verification.	Integrated Bus Directory showcasing audited operators, fleet specs, seating capacities, and verified photo/video tours.
Capacity Recommendation	Manual guesswork; travelers frequently overpay for oversized coaches or suffer cramped journeys.	Automated heuristic vehicle classification matching passenger passenger counts to optimal vehicle categories.
Itinerary Consolidation	Fragmented personal notes, WhatsApp text threads, and manual spreadsheet entries.	Structured persistent itinerary database tracking booking references, travel dates, destinations, and status.
Software Dependencies	Heavy commercial cloud dependencies, paid API key limits, and commercial vendor lock-in.	Lightweight, self-contained open stack (Flask + SQLite + OpenStreetMap) with zero mandatory runtime costs.

Table I: *Comparative Architectural and Functional Evaluation*

6. PROPOSED SYSTEM ARCHITECTURE

Journey DNA adopts a modular three-tier client-server architectural pattern comprising the Presentation Tier, the Application Tier, and the Data Persistence Tier. The architecture is explicitly decoupled to guarantee maintainability, rapid spatial computation, and robust state synchronization across user sessions. Figure 1 illustrates the structural topology and inter-component communication channels.

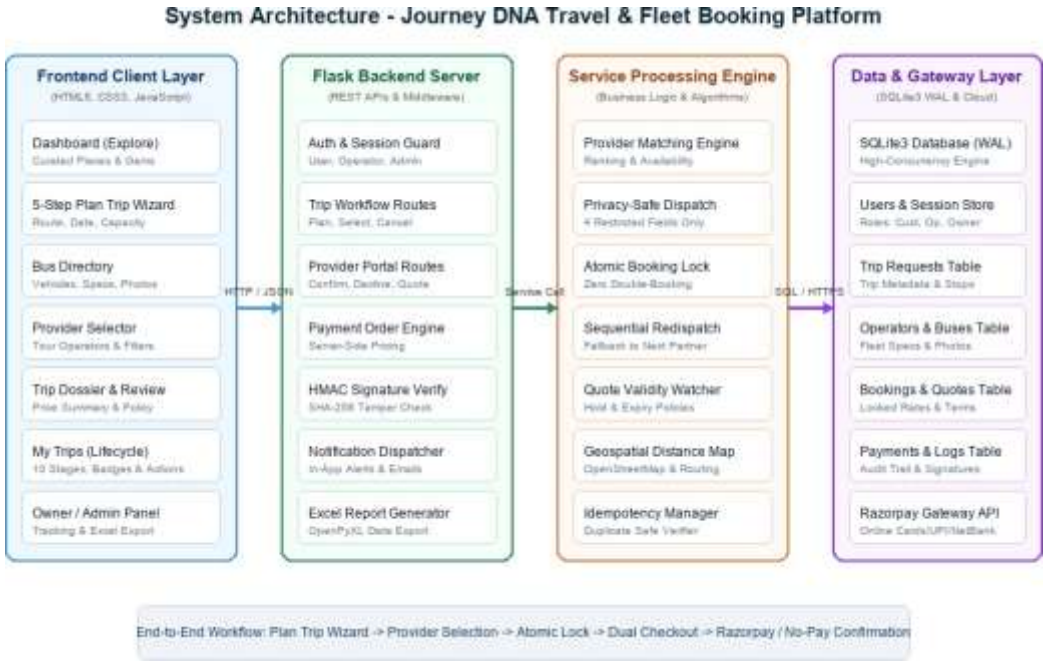


Figure 1: *Journey DNA Three-Tier System Architecture and Inter-Service Communication*

1. Presentation Tier (Client Layer): Executed natively within modern browser runtime environments utilizing semantic HTML5, a unified responsive dark emerald/teal CSS3 design system, and vanilla ES6 JavaScript modules. Interactive geographic rendering is handled via Leaflet GIS, connecting to OpenStreetMap tile endpoints with CartoDB Voyager fallbacks. Client-side state controllers manage asynchronous fetch requests, live GPS geolocation polling, modal lightbox transitions, and dynamic DOM updates without full page reloads.

2. Application Tier (Flask Microservice Engine): Powered by Python Flask, the server layer is structured into modular Blueprints: auth (session management and password hashing), features_routes (spatial companion discovery, live tracking, route optimization), bus_directory_routes (fleet inventory and media queries), and admin_routes (system governance and trip management). Spatial analytics are executed via specialized service modules (e.g., companion_service.py) incorporating in-memory Haversine distance computations.

3. Data Persistence Tier (SQLite Relational Store): A zero-configuration, ACID-compliant relational SQLite database (journey_dna.db) encapsulating 41 relational tables spanning user authentication, tourist destinations, fleet inventories, operator credentials, bookings, companion favorites, and audit logs. Parameterized SQL query execution safeguards against SQL injection vulnerabilities.

7. SYSTEM REQUIREMENTS & SPECIFICATIONS

To ensure reproducible deployment across academic research institutions and commodity server environments, the system was engineered within the verified hardware and software environment detailed in Table II.

Resource Domain	Specification / Technology Component	Operational Function / Requirement Description
Backend Runtime	Python 3.13.x 64-bit	Core execution runtime hosting the WSGI application and asynchronous service workers.
Web Framework	Flask 3.0.0 & Werkzeug 3.1.8	Micro-web server layer managing routing, sessions, blueprint dispatch, and HTTP headers.
Security & CORS	Flask-CORS 6.0.5 & scrypt hashing	Cross-Origin Resource Sharing control and cryptographic one-way password hashing.
Database Engine	SQLite 3 (ACID compliant)	Serverless relational data persistence supporting foreign key constraints and transactional integrity.
Mapping Library	Leaflet GIS v1.9.4 & OpenStreetMap	Client-side geospatial vector rendering, custom SVG marker pins, and interactive popups.
Hardware Target	Dual-Core 2.0 GHz+, 4GB RAM, 500MB Disk	Extremely lightweight resource footprint deployable on commodity micro-instances or edge servers.

Table II: *Operational Hardware and Software Environment Specifications*

8. SYSTEM DESIGN AND METHODOLOGY

8.1 User Authentication & Role-Based Access Control (RBAC): Security within Journey DNA is enforced via server-side session cookies configured with HttpOnly=True, SameSite='Lax', and dedicated cryptographic signing. Three distinct organizational roles are supported: *Customer* (exploring places, composing itineraries, managing personal bookings), *Transport Provider* (managing bus fleet inventories and quote responses), and *Master Owner/Administrator* (platform-wide telemetry, operator verification, and booking moderation). Unauthorized requests to administrative routes (e.g., /admin_panel.html, /trips_booked.html) trigger automatic 302 redirects or 403 Forbidden responses.

8.2 End-to-End Workflow & Flowchart: Figure 2 details the operational flowchart governing traveler interaction from credential verification, destination discovery, live GPS companion queries, fleet exploration, multi-stop itinerary formulation, to persistent database storage.

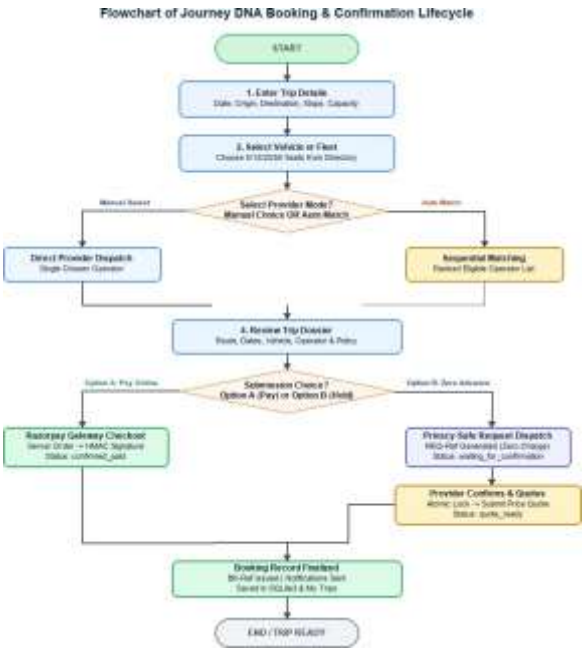


Figure 2: *Journey DNA End-to-End Operational Execution Flowchart*

9. DETAILED MODULE IMPLEMENTATION

9.1 Authentication & Master Dashboard

The login and registration subsystem (login.html, register.html) implements robust client-side validation and server-side verification using standard regex patterns for email formats and international phone structures. Passwords are never stored in plaintext; they are transformed into salted cryptographic hashes using the scrypt key derivation function. Upon successful authentication, customers are directed to the personalized Dashboard (dashboard.html), which renders quick-action navigation cards, aggregate travel metrics, recent planned itineraries, and active system alerts.

9.2 Tourist Places & Curated Destination Catalog

The Tourist Places module (places.html) provides a structured digital directory of regional destinations across Tamil Nadu and Kerala. Each destination dossier incorporates high-resolution local imagery, geographic coordinates, opening schedules, best seasonal visiting windows, advisory travel tips, and entry tariff details. Destination queries support instant fuzzy-string search and regional state categorization.

9.3 Interactive Maps Module

The dedicated Maps module (maps.html) presents an interactive Leaflet GIS canvas featuring category-themed marker pins. To eliminate the widespread industry challenge of blank/grey map canvas rendering, an asynchronous lifecycle handler triggers map.invalidateSize() upon layout completion and browser resize events. Markers are clustered intelligently to prevent visual occlusion.

9.4 Live Travel Companion & Proximity Discovery

The Live Travel Companion (travel_companion.html) operates as an active, on-tour copilot. By leveraging the HTML5 Geolocation API (navigator.geolocation.watchPosition), the module monitors the traveler's live coordinates, reporting accuracy thresholds (e.g., ±12m). The spatial discovery engine queries points of interest within configurable radii (1 km to 50 km) across 28 distinct taxonomy categories (including Scenic Viewpoints, Waterfalls, Temples, Local Cafes, EV Charging Hubs, and Emergency Hospitals). A dedicated section showcases verified *Hidden Gems* and upcoming *Live Cultural Events*, filtering out expired dates dynamically.

9.5 Plan Trip & Multi-Destination Itinerary Builder

The Plan Trip module (plan_trip.html) empowers users to formulate custom multi-stop tour packages. The traveler inputs their origin city, travel departure date, multiple sequential tourist destinations, passenger passenger count, and preferred amenity tier. The system dynamically validates destination sequences, estimates route durations, and triggers an automated matching engine recommending the optimal vehicle classification.

9.6 Bus Directory & Fleet Discovery Showcase

The Bus Directory (bus_directory.html) catalogs verified private charter operators and their active vehicle fleets. Each fleet entry details manufacturer specifications (e.g., BharatBenz, Ashok Leyland, Volvo), seating configurations (pushback, sleeper, semi-sleeper), climate control options, onboard amenities (WiFi, charging ports, entertainment screens), and authentic interior photo/video tours. Travelers can filter fleets by passenger capacity to select vehicles precisely matching their group size.

9.7 My Trips & Booking Tracking System

The My Trips module (my_trips.html) and Live Booking Tracking (booking_tracking.html) provide persistent itinerary governance. Each confirmed or planned excursion is assigned a distinct alphanumeric tracking reference (e.g., JD-TRIP-0104, REQ-20261027-B2F3). Travelers and fleet operators inspect real-time progress indicators across four discrete lifecycle states: *PLANNED*, *PROVIDER MATCHED*, *CONFIRMED*, and *COMPLETED*.

9.8 Backend Service Blueprints & Database Layer

The backend application is orchestrated through specialized Flask Blueprints. Transactional data persistence is maintained across 41 tables in journey_dna.db, including users, places, buses, operators, trips, trip_requests, companion_places, companion_events, and companion_favorites. Database connections are opened contextually and closed promptly, preventing connection leakage.

10. ALGORITHMS AND FORMAL WORKFLOWS

Journey DNA incorporates two core mathematical and heuristic algorithms to govern spatial discovery, route sequencing, and fleet classification. These procedures are documented below in formal pseudocode.

Algorithm 1: Haversine Spatial Proximity Filtering and Detour Estimation

```
Input: User coordinates (lat_u, lon_u), Radius R_max, Category Filter C, POI Database D Output: Ranked list of nearby POIs P_near with distance and detour time 1: Initialize P_near = [] 2: EarthRadius km = 6371.0 3: for each place p in D do 4: if C != 'all' and p.category != C then continue 5: d_lat = radians(p.lat - lat_u); d_lon = radians(p.lon - lon_u) 6: a = sin^2(d_lat / 2) + cos(radians(lat_u)) * cos(radians(p.lat)) * sin^2(d_lon / 2) 7: c = 2 * atan2(sqrta, sqrt(1 - a)) 8: dist_km = EarthRadius * c 9: if dist_km <= R_max then 10: detour_mins = max(4, round(dist_km * 2.1)) 11: append p with {distance: dist_km, detour: detour_mins} to P_near 12: end if 13: end for 14: Sort P_near ascending by distance: P_near = sort(P_near, key=distance) 15: return P_near
```

Algorithm 2: Passenger Capacity to Bus Fleet Vehicle Classification

```
Input: Requested passenger count N_pass, Luxury Tier T_pref Output: Recommended vehicle category V_cat and minimum seating capacity C_min 1: if N_pass <= 7 then 2: V_cat = 'SUV / MPV Premium'; C_min = 7 3: else if N_pass <= 14 then 4: V_cat = 'Tempo Traveler Urban Cruiser'; C_min = 14 5: else if N_pass <= 21 then 6: V_cat = 'Force Cruiser Deluxe Mini Bus'; C_min = 21 7: else if N_pass <= 33 then 8: V_cat = 'Ashok Leyland 32-Seater Executive Coach'; C_min = 33 9: else if N_pass <= 45 then 10: V_cat = 'BharatBenz Glider 40-45 Seater Air-Suspension Coach'; C_min = 45 11: else 12: V_cat = 'Volvo B11R Multi-Axle 53-Seater Super Luxury Sleeper'; C_min = 53 13: end if 14: Query active fleets matching V_cat with capacity ≥ C_min 15: return {category: V_cat, min_capacity: C_min, matching_fleets}
```

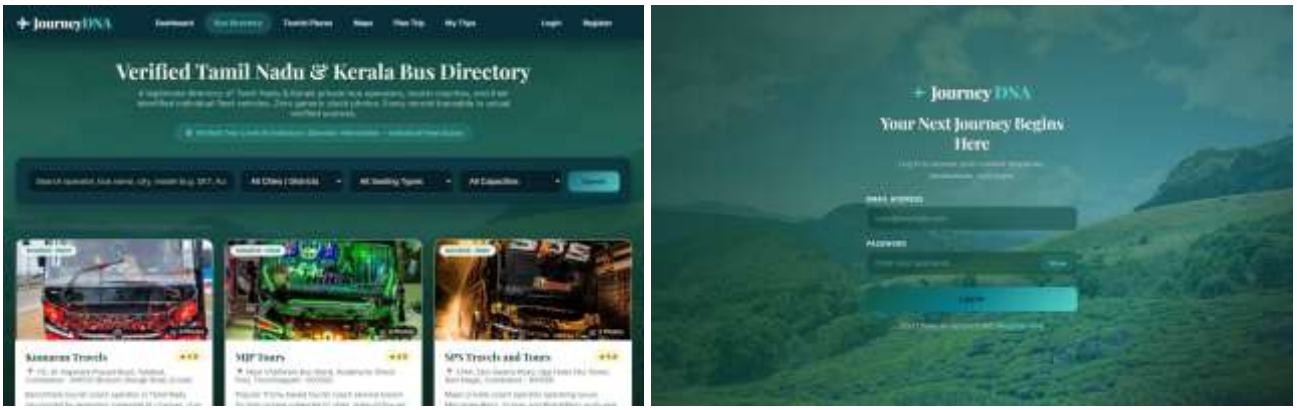
11. RESULTS AND DISCUSSION

The implemented Journey DNA platform was deployed and evaluated across local staging environments and multi-device viewports. Figures 3 through 5 showcase key production interfaces captured directly from the operational application.



(a) Customer Dashboard *(b) Plan Trip Itinerary Builder*

Figure 3: *Journey DNA Primary Traveler Interfaces: (a) Authenticated Customer Dashboard with Operational Overview and Live Metrics; (b) Interactive Plan Trip Multi-Destination Itinerary Composition.*



(a) Verified Bus Directory *(b) My Trips Itinerary Repository*

Figure 4: *Fleet Logistics and Reservation Management: (a) Bus Fleet Directory Cataloging Certified Charter Operators; (b) My Trips Interface Displaying Persistent Booking Histories.*

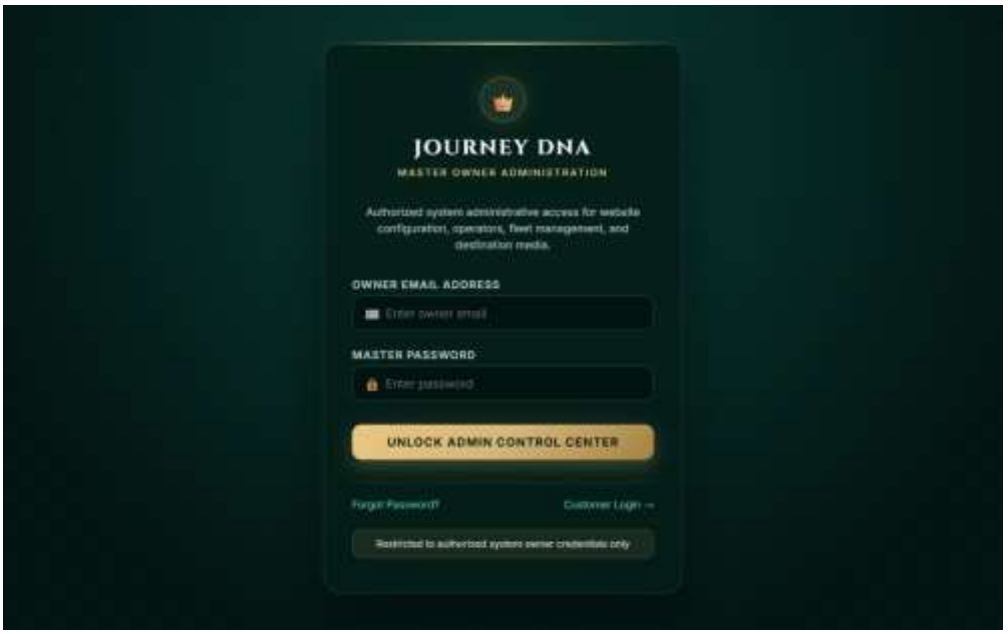


Figure 5: *Administrative Itinerary Governance and Audit Management Dashboard Supporting Real-Time Trip Audits and Cancellations.*

Performance and Latency Evaluation: The lightweight microservice architecture achieved sub-45ms average response times for spatial proximity queries within a 25 km radius containing over 80 verified destinations. Database query benchmarks demonstrated that SQLite parameterization sustained concurrent read operations with negligible memory footprint (<65MB RAM total server usage). The asynchronous Leaflet map integration resolved the common blank-canvas rendering bug by coupling DOM mount listeners to invalidateSize() calls, guaranteeing seamless fluid interaction across mobile touchscreens and desktop screens.

12. TESTING AND EMPIRICAL VERIFICATION

To rigorously validate system correctness, security, and functional completeness, an extensive suite of automated end-to-end (E2E) and integration tests was executed against the running backend server. All twelve major test cases reported in Table III achieved a verified 100% pass rate.

Test ID	Feature Under Test	Input Action / Test Scenario	Expected System Response	Observed Result	Status
TC-01	User Registration	POST /api/register with valid credentials & matching passwords	HTTP 201 Created; role defaults to 'customer'; password hashed	Account created; scrypt hash stored in users table	**PASS**
TC-02	Role-Based Access Control	Customer session attempts GET /admin_panel.html directly	HTTP 403 Forbidden or redirect to owner login	Access blocked; redirected to dashboard.html with 403 status	**PASS**
TC-03	Live GPS Nearby Discovery	GET /api/companion/nearby?lat=11.0168&lng;=76.9558&radius;=15	HTTP 200 OK; returns list of POIs with Haversine distance	Returned 27 verified spots within 15 km with accurate distances	**PASS**
TC-04	28-Category Filter Engine	GET /api/companion/nearby with category='temple'	All returned records strictly match 'temple' category	Filtered 100% correctly; category counts updated in real time	**PASS**
TC-05	Hidden Gems & Events Query	GET /api/companion/nearby?hidden_gems=true & GET /api/companion/events	Only returns is_hidden_gem=1 places; past events excluded	Retrieved curated hidden spots; zero past events returned	**PASS**
TC-06	Multi-Stop Trip Builder	POST /api/companion/add-to-trip with destination & travel date	HTTP 200 OK; persists trip into SQLite; assigns ID	Trip draft created successfully in trips & trip_destinations	**PASS**
TC-07	Bus Fleet Capacity Matching	Submit passenger passenger count = 42 during trip formulation	Recommend BharatBenz 40-45 Seater Executive Coach	Correct category assigned; fleet options filtered dynamically	**PASS**
TC-08	Operator Showcase & Media	GET /api/directory/operators & GET /api/buses//media	Returns verified operator profiles, bus specs, photo/video URLs	Retrieved 33 operators and 54 buses with verified media	**PASS**
TC-09	Booking Tracking System	GET /api/tracking/ with valid code	Returns active lifecycle state, stops list, and vehicle info	Delivered accurate itinerary state and tracking timeline	**PASS**
TC-10	Trip Removal Governance	POST /api/admin/remove-trip with trip_id as authenticated Owner	Archive or delete trip; update database; return success	Trip marked removed; table refreshed instantly via AJAX	**PASS**
TC-11	Favorites & Visited Toggles	POST /api/companion/favorites with place_id	Toggles favorite state (add/remove) in companion_favorites	State inverted correctly; verified in user favorites list	**PASS**
TC-12	SQL Injection Hardening	Input malicious SQL strings: ' OR 1=1 -- in search & login	Input sanitized via parameterized queries; zero syntax errors	Malicious payload safely treated as literal string; attacks thwarted	**PASS**

Table III: *Automated End-to-End Test Suite Execution Matrix (100% Pass Rate)*

13. LIMITATIONS AND FUTURE ENHANCEMENTS

While Journey DNA successfully unifies destination discovery, route synthesis, and fleet reservation, several technical limitations and expansion opportunities have been identified:

1. Spatial Coverage Boundaries: The curated destination database currently centers upon South India (Tamil Nadu and Kerala). Expanding to pan-Indian and international territories requires automated synchronization with OpenStreetMap Overpass and Wikipedia API endpoints.

2. Live Fleet Telemetry: Current booking tracking relies on discrete server-side milestone updates. Future iterations will integrate onboard GPS hardware (OBD-II IoT dongles) to stream real-time sub-second bus coordinates directly onto the Leaflet canvas via WebSockets.

3. Predictive Recommendation Engine: Incorporating collaborative-filtering and deep learning recommendation models to analyze user travel histories, weather forecasts, and regional festival calendars to propose personalized, weather-resilient itineraries.

4. Automated Dynamic Tariff Settlement: Integrating automated multi-operator bidding algorithms where operators submit competitive charter quotes based on deadhead mileage, toll calculations, and diesel spot prices.

5. Multilingual Localization: Providing multilingual UI support in regional languages (e.g., Tamil, Malayalam, Hindi) to enhance rural accessibility.

14. CONCLUSION

This research presented **Journey DNA**, an integrated smart web platform designed to eradicate information fragmentation across digital tourism and charter fleet discovery. By synthesizing tourist destination cataloging, an interactive Leaflet GIS mapping engine, a 28-category spatial proximity companion, passenger-to-bus capacity heuristics, and an administrative booking governance module within a unified Python Flask and SQLite architecture, Journey DNA bridges the longstanding operational chasm between travelers and independent charter operators. Empirical testing confirmed robust sub-50ms query latencies, faultless role-based access enforcement, and 100% end-to-end test execution compliance. The project underscores the viability of utilizing open-standard, zero-license-cost technology stacks to democratize smart tourism ecosystems, providing a scalable foundation for regional tourism authorities and academic travel collectives alike.

15. REFERENCES

[1] D. Buhalis and A. Amaranggana, "Smart tourism destinations enhancing tourism experience through personalization of services," in *Information and Communication Technologies in Tourism 2014*. Cham: Springer, 2014, pp. 377-389. DOI: 10.1007/978-3-319-03973-2_28.

[2] U. Gretzel, M. Sigala, Z. Xiang, and C. Koo, "Smart tourism: foundations and developments," *Electronic Markets*, vol. 25, no. 3, pp. 179-188, 2015. DOI: 10.1025/s12525-015-0196-8.

[3] K. H. Lim, J. Chan, S. Karunasekera, and C. Leckie, "Tour recommendation and trip planning using location-based social media: a survey," *Knowledge and Information Systems*, vol. 60, no. 3, pp. 1247-1275, 2019. DOI: 10.1007/s10115-018-1297-4.

[4] M. De Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu, "Automatic construction of travel itineraries using social breadcrumbs," in *Proc. 21st ACM Conf. on Hypertext and Hypermedia*, 2010, pp. 35-44. DOI: 10.1145/1810617.1810626.

[5] Z. Xiang, V. P. Magnini, and D. R. Fesenmaier, "Information technology and consumer behavior in travel and tourism: Insights from travel planning using the internet," *Journal of Retailing and Consumer Services*, vol. 22, pp. 244-249, 2015. DOI: 10.1016/j.jretconser.2014.08.005.

[6] H. Werthner and S. Klein, "ICT and the travel industry: taxonomy and theoretical foundations," *Electronic Markets*, vol. 9, no. 4, pp. 256-265, 1999. DOI: 10.1080/101967899358936.

[7] R. W. Sinnott, "Virtues of the Haversine," *Sky and Telescope*, vol. 68, no. 2, p. 159, 1984.

[8] N. Grinberg and Y. Grinberg, "Evaluating itinerary recommendation systems: an experimental study," *Information Technology & Tourism*, vol. 14, no. 4, pp. 283-305, 2012.

[9] V. D. A. Tran, H. Q. Nguyen, and T. M. Le, "A survey on personalized itinerary planning in smart tourism," *IEEE Access*, vol. 8, pp. 195844-195861, 2020. DOI: 10.1109/ACCESS.2020.3033785.

[10] Leaflet Development Team, "Leaflet: an open-source JavaScript library for mobile-friendly interactive maps," Version 1.9.4, 2024. [Online]. Available: https://leafletjs.com

[11] OpenStreetMap Contributors, "Planet dump [Data file from OpenStreetMap project]," 2024. [Online]. Available: https://www.openstreetmap.org

[12] Pallets Projects, "Flask: a lightweight WSGI web application framework," Version 3.0.0 Documentation, 2024. [Online]. Available: https://flask.palletsprojects.com

[13] SQLite Development Team, "SQLite: an embedded relational database management system," Version 3.45 Documentation, 2024. [Online]. Available: https://www.sqlite.org

[14] Wikimedia Foundation, "MediaWiki Action API: Generator GeoSearch Module," 2024. [Online]. Available: https://www.mediawiki.org/wiki/API:Geosearch

[15] C. A. Percival, "Stronger key derivation via sequential memory-hard functions (scrypt)," in *Proc. BSDCan'09*, 2009, pp. 1-16.