ISSN No. 0976-5697

VOLUME 17 SPECIAL ISSUE 1, JULY 2026

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Aarohan: A Smart Digital Platform to Promote Eco & Cultural Tourism in Jharkhand Using AI and Community-Driven Design

Swati Kanwar
Dept. of Computer Science and Engineering
Geetanjali Institute of Technical Studies
Udaipur, Rajasthan, India
swatikanwar2004@gmail.com

Puru Joshi
Dept. of Computer Science and Engineering
Geetanjali Institute of Technical Studies
Udaipur, Rajasthan, India
purujoshi068@gmail.com

Mugdh Mathur
Dept. of Computer Science and Engineering
Geetanjali Institute of Technical Studies
Udaipur, Rajasthan, India
mugdhmathur2004@gmail.com

Dr. Mayank Patel
Dept. of Computer Science and Engineering
Geetanjali Institute of Technical Studies
Udaipur, Rajasthan, India
mayank999_udaipur@yahoo.com

***Abstract*—**Jharkhand possesses remarkable natural and cultural assets — from the sal forests of Betla National Park to the sacred ghats of Baidyanath Dham — yet its tourism sector remains structurally underdeveloped. The state attracted fewer than 5.5 million domestic tourists in 2022, and no unified digital infrastructure exists to connect visitors with local guides, artisans, or homestay operators [1]. Tribal communities, who constitute approximately 26% of the state population, have no digital marketplace access. Aarohan is a serverless, AI-powered web platform engineered to close that gap. Built on React with Tailwind CSS on the frontend, Node.js and Express.js on the backend, and Supabase/Neon PostgreSQL for data persistence, the system delivers four core capabilities: a multilingual AI itinerary planner and chatbot powered by the Google Gemini API; immersive 360° destination previews via Photosphere.js; an interactive map layer built on Leaflet.js; and a verified community marketplace with Razorpay payment processing and cryptographic guide verification using the Node.js crypto module. Deployments run serverlessly on Vercel with Grafana analytics for governance. Three testing layers validated the system: Postman functional testing (23 test cases, 0 failures), hey load testing (0 errors at 50-concurrent baseline), and Burp Suite penetration testing (0 confirmed vulnerabilities across 47 attack vectors). Aarohan demonstrates that community-first design combined with accessible AI infrastructure can serve as a replicable model for digital tourism development in underserved Indian states.

***Index Terms*—**digital tourism, AI itinerary planner, community marketplace, Jharkhand, React, Tailwind CSS, Node.js, Express.js, Supabase, Razorpay, penetration testing, Leaflet.js, Photosphere.js, Gemini API, rural empowerment, serverless architecture.

I. INTRODUCTION

Imagine planning a trip to Hundru Falls — one of Jharkhand's most photographed waterfalls — using only what the internet can provide. There is no booking platform. No guide directory. No way to know whether the guesthouse near Betla National Park even has a room available in March. The information simply does not exist in digital form. That is the starting condition for Aarohan.

Jharkhand's tourism potential is not in question. The state holds dense wildlife corridors at Hazaribagh Wildlife Sanctuary and Dalma Wildlife Sanctuary, the holiest Shiva shrine in eastern India at Baidyanath Dham (Deoghar), the singular landscape of Netarhat and Patratu Valley, and spiritual landmarks like Rajrappa Temple, Pahari Mandir (Ranchi), and the Yogoda Satsanga Society campus. Year-wise tourist inflow data from the Jharkhand Tourism Department [1] shows a recovery arc from 2013 to 2022 — but that recovery has happened without a single unified digital infrastructure to support it. The number of tourists arriving each year is disconnected from any platform that could serve them. Tribal communities, who represent roughly 26% of the state's total population, remain entirely outside the formal tourism economy.

Three structural gaps define the problem. The **information gap**: tourists cannot discover destinations, check availability, or preview experiences before arriving. The **infrastructure gap**: local guides, artisans, and homestay operators have no tools to list, verify, or transact. The **economic gap**: earnings from tourism do not reach communities because no trusted marketplace connects them to visitors. The OECD's 2024 report on AI and tourism [2] confirms that AI-driven personalization is the global direction — but explicitly notes that the gap is widest in developing economies with low-resource language contexts.

Aarohan addresses all three. The platform serves a dual mission: it is an intelligent travel companion for tourists and an economic empowerment engine for local service providers. Four technical contributions define the system. First, a Google Gemini API-powered AI module delivers a multilingual chatbot (Hindi, English, Santali) and a personalized day-by-day itinerary planner — all server-side, keys never on the client. Second, Photosphere.js provides browser-native 360° VR destination previews alongside Leaflet.js interactive maps with custom point-of-interest markers. Third, a community marketplace built on Razorpay handles INR transactions with HMAC-SHA256 webhook verification, while SHA-256 credential hashing via the Node.js crypto module enables guide verification without centralized database exposure. Fourth, a Grafana analytics dashboard connected to Supabase PostgreSQL gives government administrators real-time footfall and revenue data for policy decisions.

The rest of this paper is organized as follows: Section II surveys related work in AI-driven tourism, immersive technology, and digital rural platforms. Section III presents the proposed solution with formal algorithm specifications. Section IV details the system architecture, database schema, and deployment pipeline. Section V reports testing outcomes across functional, load, and security layers. Section VI concludes with limitations and future directions.

II. LITERATURE SURVEY

The research landscape around digital tourism has expanded considerably since 2020, but few systems combine all the components Aarohan targets. The literature clusters into three meaningful groups.

A. AI and Conversational Systems in Tourism—

Gursoy and Cai [5] provide a comprehensive survey of AI research trends in hospitality and tourism, concluding that AI adoption is growing rapidly but remains concentrated in high-resource language environments — English, Mandarin, and Western European languages. Low-resource Indic languages, and tribal languages in particular, are absent from nearly every deployment studied. That gap is directly relevant to Aarohan's Santali language support requirement.

Prasanna, Pushparaj, and Kushwaha [6] apply the TCM/ADO framework to conversational AI in tourism and demonstrate that chatbot effectiveness is strongly dependent on contextual relevance — generic responses degrade user engagement below any baseline. Their findings directly informed Aarohan's decision to feed structured preference data into the Gemini API prompt rather than issuing open-ended queries.

Samala et al. [7] quantify the economic and experiential benefits of AI and robotics across multiple tourism contexts. Their evidence base supports the claim that AI-driven personalization yields measurable gains in tourist satisfaction and spending. Tripathi et al. [9] extend this into chatbot-specific consumer engagement, identifying perceived humanness and social presence as primary drivers of trust. The Aarohan chatbot design accounts for this: responses are streamed progressively to simulate conversation pace, and the system maintains session context within a single itinerary request.

B. Immersive Technologies: AR/VR in Tourism—

Chourasia et al. [4] confirm strong demand for AR/VR-based tourism experiences in India but document a near-total absence of such deployments at Tier-2 and Tier-3 destinations. The study explicitly calls out cultural and ecological sites in Jharkhand, Odisha, and Chhattisgarh as underserved. Aarohan's Photosphere.js integration directly addresses this gap, requiring no browser plugin and no dedicated hardware.

Shen et al. [8], writing in the context of COVID-19 disruption, validate browser-native VR as the most accessible approach for tourism education and pre-visit engagement — precisely because it eliminates installation friction for non-technical users. Kothari, Patel, and Nyati [10] establish a theoretical framework for cognitive BIM and AR/VR systems, providing grounding for how immersive previews reduce the uncertainty that prevents first-time visitors from committing to unfamiliar destinations.

C. Digital Platforms, Rural Empowerment, and Indian Tourism—

The Jharkhand Tourism Department's year-wise inflow data [1] provides the empirical baseline for this work: tourist arrivals show a partial recovery arc from 2013 to 2022, but that growth has occurred without any supporting digital infrastructure. The data makes visible a structural inefficiency — demand is recovering, but supply-side digital tools do not exist.

Narmaditya [3] demonstrates that digital transformation in rural settings produces measurable economic growth when the platform is designed around community workflows rather than urban service-delivery assumptions. The marketplace component of Aarohan draws directly from this model: provider onboarding, verification, and listing workflows were designed in conversation with the kinds of constraints — limited smartphones, intermittent connectivity, low digital literacy — that Narmaditya documents.

Taken together, the surveyed work points to one clear gap. No existing system combines serverless AI itinerary planning, cryptographic guide verification, browser-native 360° VR, and a community-first marketplace in a single platform designed for low-infrastructure Indian tourism contexts. Each component exists somewhere in the literature. The integration is new.

III. PROPOSED SOLUTION

A. System Overview and Architecture—

Aarohan is built on a three-tier serverless model. The **Presentation Tier** consists of React components styled with Tailwind CSS, Leaflet.js for map interaction, and Photosphere.js for 360° panoramic previews. The **Logic Tier** runs Node.js and Express.js on Vercel's serverless infrastructure — every API route compiles to an independent serverless function, eliminating idle-time costs and enabling automatic horizontal scaling under load. Git-push-to-main triggers production deployment via Vercel's CI/CD pipeline. The **Data Tier** uses Supabase/Neon managed PostgreSQL for

all relational data and Supabase Object Storage for unstructured assets: listing images, provider verification documents (signed URLs), and photosphere panorama files. The architecture is shown in Fig. 3.

B. AI Itinerary Planning Module—

The itinerary planner accepts tourist preferences and constructs a personalized day-by-day travel plan via the Google Gemini API. All Gemini calls are server-to-server from the Express backend — API keys are never exposed to the browser. ***Algorithm 1 formalizes the generation logic.***

```
Algorithm 1: AI Itinerary Generation
Input:  Tourist preferences P {destination,
duration, budget,
         interests, preferred language L}
Output: Personalized day-by-day itinerary I
```

```
1: IF destination = ∅ OR duration = ∅ THEN
  2:   PROMPT user to complete missing
fields
  3:   RETURN
4: END IF
5: Construct structured prompt ρ from
preferences P
6: Send server-to-server POST to Gemini API
with ρ
7: IF Gemini API response ≠ HTTP 200 THEN
  8:   key ← hash(P)  // preference
fingerprint
  9:   IF key ∈ cache THEN
    10:    RETURN Icached
  11:  ELSE
    12:    DISPLAY error: 'AI service
temporarily unavailable'
    13:    RETURN
  14:  END IF
15: END IF
16: Parse Gemini response into structured
itinerary I
17: Translate I into preferred language L
18: STORE I in Supabase linked to tourist
session
19: RETURN I to React frontend for display
```

The caching strategy uses a SHA-256 fingerprint of the preference object as the cache key. Repeated or similar queries resolve from cache, reducing Gemini API call volume. Server-side execution is mandatory: exposing the Gemini API key on the client would allow unlimited unmetered calls from any browser and is not negotiable architecturally.

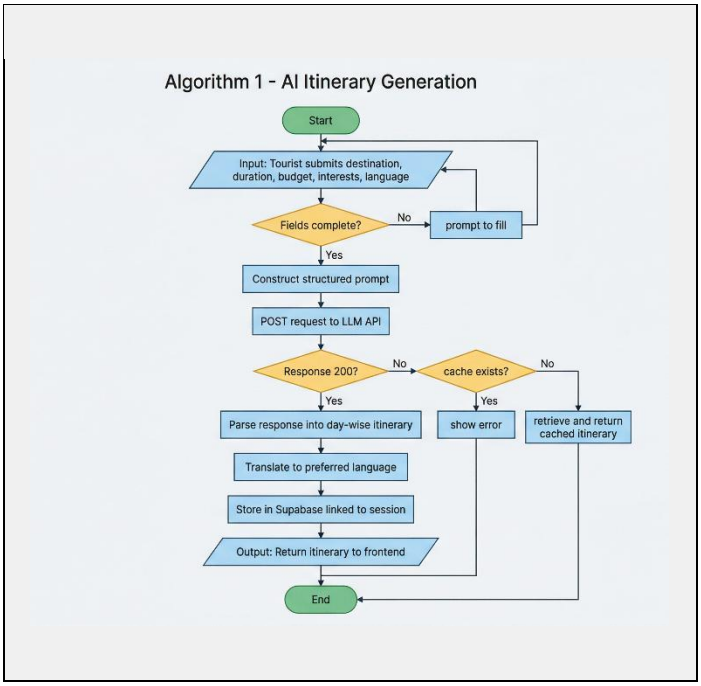

Fig. 1. AI itinerary generation flowchart (Algorithm 1). Decision diamonds correspond to validation, API response, and cache lookup steps.

C. Marketplace, Booking, and Payment Module—

Jharkhand's guide economy runs entirely on personal contacts. There is no booking system. No receipts. No verification. Aarohan replaces that with a structured marketplace where only cryptographically verified providers can accept bookings. ***Algorithm 2 formalizes the booking and payment flow.***

```
Algorithm 2: Marketplace Booking and Payment
Input:  Tourist session T, Service listing
L, Payment details D
Output: Confirmed booking record B, Payment
receipt R
```

```
1:  Tourist selects service listing L
2:  IF L.provider_status ≠ 'Verified' THEN
  3:    DISPLAY provider verification
warning
  4:    HALT booking
5:  END IF
6:  Display booking summary (service, date,
price) to T
7:  T confirms → POST to Razorpay API:
create order
8:  IF order creation fails THEN
  9:    LOG error; NOTIFY tourist; ABORT
10: END IF
11: Receive order_id → display Razorpay
payment modal
12: T completes payment → Razorpay webhook
fires
13: Verify webhook signature:
        sig ← HMAC-SHA256(body,
RAZORPAY_SECRET)
```

```
14: IF sig ≠ webhook.X-Razorpay-Signature
THEN
  15:   REJECT webhook; LOG security
incident; RETURN
16: END IF
17: UPDATE B.status = 'Confirmed' in
Supabase
18: SEND confirmation email to T
19: SEND booking notification to provider of
L
20: INSERT analytics_event for Grafana
dashboard
21: RETURN B, R
```

Only verified providers can receive bookings — the status check at step 2 is enforced server-side, not just on the UI. This prevents a tourist from paying for a service whose provider has not completed the verification process. The HMAC-SHA256 webhook verification at steps 13–16 ensures that payment confirmation events cannot be spoofed by arbitrary HTTP POST requests.

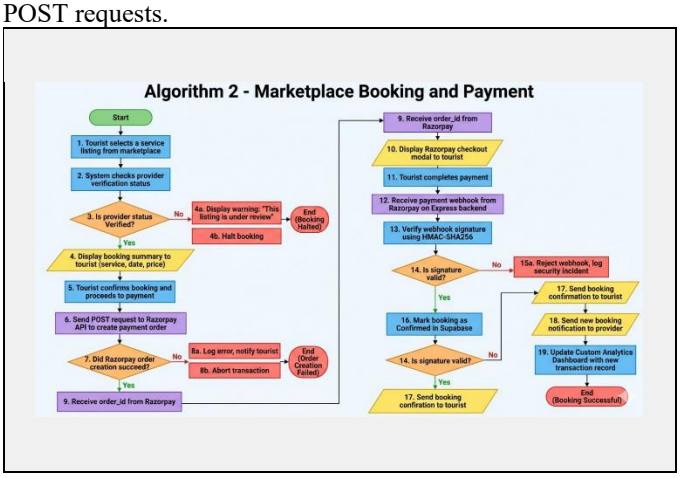

Fig. 2. Marketplace booking and payment flowchart (Algorithm 2). Decision diamonds correspond to provider status, Razorpay order creation, and HMAC signature verification steps.

D. Guide Verification and Trust System—

Provider trust is the hardest problem in any community marketplace. Aarohan solves it with a pragmatic cryptographic fingerprint — not a distributed ledger, just the Node.js built-in crypto module. When a provider uploads a credential document during registration, the backend computes:

```
crypto.createHash('sha256').update(cred
      entialData).digest('hex')
```

The resulting hex digest is stored in the `providers.credential_hash` column in PostgreSQL. An admin reviews the physical document, recomputes the hash independently, and confirms the match before setting provider status to "Verified." This model is zero-cost and sufficient for the trust requirement at hand. It is not blockchain and does not need to be.

IV. SYSTEM ARCHITECTURE

A. Backend API Design—

The Express.js backend is organized into five domain modules: auth routes, listing routes, booking routes, AI routes, and admin routes. Every protected route passes through a JWT authentication middleware that extracts the bearer token from the Authorization header, verifies it with HS256 signing, and resolves `user_id` and `role` before the route handler executes. Role mismatch returns 403 Forbidden immediately — the handler never runs.

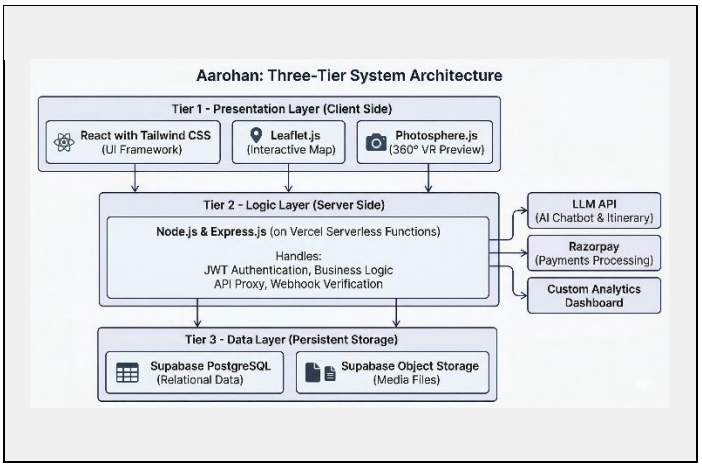

Fig. 3. Three-tier serverless system architecture. Presentation tier communicates with the Logic tier via RESTful API calls; the Logic tier reads and writes to the Data tier via Supabase client and Object Storage SDK.

TABLE I. API ENDPOINT REFERENCE

Method	Endpoint	Auth	Description
POST	/auth/register	Public	New user registration
POST	/auth/login	Public	JWT issuance
GET	/destinations	Public	List all destinations
GET	/destinations/:id	Public	Destination detail
GET	/listings	Public	Browse marketplace
POST	/listings	Provider JWT	Create new listing
POST	/bookings	Tourist JWT	Create booking + Razorpay order
POST	/webhook/razorpay	Server-to-server	Confirm payment (HMAC verified)
POST	/itinerary	Tourist JWT	Generate AI itinerary
POST	/chat	Tourist JWT	Chatbot message
GET	/admin/providers	Admin JWT	Provider verification queue
PUT	/admin/providers/:id	Admin JWT	Approve or reject provider

B. Database Design—

The Supabase PostgreSQL schema is organized around six core tables. The `users` table holds tourist profiles, hashed passwords, and JWT credentials. The `providers` table carries

guide and artisan data including a `verification_status` enum (pending/verified/rejected) and the SHA-256 credential hash. The `listings` table links to providers with its own status column. The `bookings` table joins tourists and listings via foreign keys, storing the Razorpay `order_id` and booking status. The `destinations` table holds coordinates, photosphere asset URLs, and Leaflet marker JSON per destination. The `analytics_events` table logs every significant user action for Grafana.

Supabase Row-Level Security (RLS) enforces access boundaries: tourists query only their own bookings; providers see bookings for their listings; admins see all rows. Object Storage buckets are organized by purpose — `avatars/`, `verification-docs/` (signed URLs, private), `listing-images/` (public CDN), and `photosphere-assets/` (public CDN).

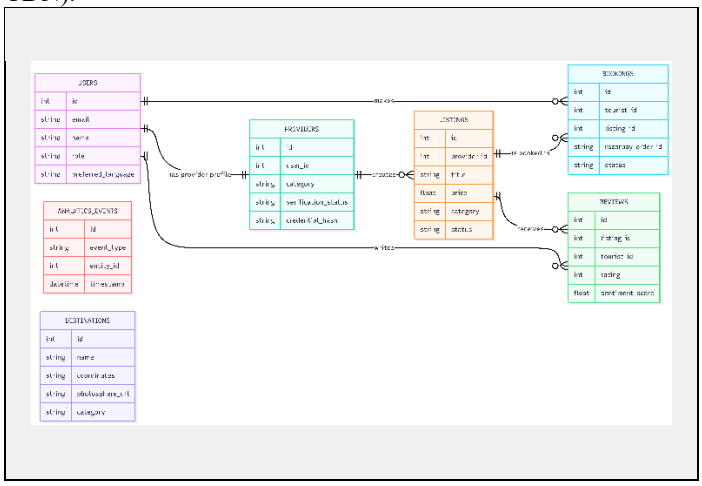

Fig. 4. Entity-relationship diagram for the Aarohan database. Core entities and their cardinality relationships are shown; primary and foreign key constraints enforced in Supabase PostgreSQL.

C. Frontend Architecture—

The React component tree is organized around user roles. Core components include `<Navbar>`, `<DestinationCard>`, `<ListingCard>`, `<ItineraryPanel>`, `<ChatWidget>`, `<MapView>` (Leaflet.js), `<SphereView>` (Photosphere.js), `<BookingModal>` (Razorpay checkout), `<ProviderDashboard>`, and `<AdminQueue>`. State is managed with `useState` for local component state, `useEffect` for data fetching, and Context API for global auth state (JWT token plus user object). Leaflet.js is initialized inside `useEffect` after DOM mount with OpenStreetMap base tiles and custom `L.icon` per POI category. Photosphere.js receives the panorama image URL as a prop; a standard `<img>` fallback renders if the library fails to initialize on older devices.

D. Deployment and DevOps—

Every push to the `main` branch on GitHub triggers a production deployment through Vercel's CI/CD pipeline. Pull requests automatically receive preview URLs, enabling pre-merge validation without affecting production traffic. The `vercel.json` configuration handles SPA routing rewrites, serverless function timeout settings, and environment variable bindings for Supabase credentials, the Gemini API key, and Razorpay secrets. Grafana connects to Supabase PostgreSQL as a data source. Dashboard panels display tourist footfall trends, top-5 destinations by bookings, revenue by provider category, sentiment score gauge from review text, and active listings count. The Grafana layer requires no additional backend instrumentation — it reads directly from the `analytics_events` table populated by step 20 of Algorithm.

V. RESULTS

A. Functional and Integration Testing — Postman—

Testing was organized into six Postman Collections — Auth, Destinations, Marketplace, Bookings, AI, and Admin — sharing a single environment file with variables for `baseURL`, `JWT`, `testUserId`, and `testListingId`. Pre-request scripts handle token refresh automatically, so auth failures do not cascade into unrelated test failures.

TABLE II. POSTMAN FUNCTIONAL TEST RESULTS

ID	Endpoint	Method	Expected	Actual	Status
T-01	POST /auth/register	POST	201 Created	201 Created	Pass
T-02	POST /auth/login	POST	200 OK + JWT	200 OK + JWT	Pass
T-03	Protected — no token	GET	401 Unauthorized	401 Unauthorized	Pass
T-04	Protected — wrong role	GET	403 Forbidden	403 Forbidden	Pass
T-05	GET /destinations	GET	200 OK, JSON array	200 OK, JSON array	Pass
T-06	POST /bookings	POST	201, valid order_id	201, valid order_id	Pass
T-07	POST /itinerary	POST	200 OK, itinerary	200 OK, itinerary	Pass
T-08	POST /listings	POST	201 Created	201 Created	Pass
T-09	GET /admin/providers	GET	200 OK, queue array	200 OK, queue array	Pass
T-10	PUT /admin/providers/:id	PUT	200 OK, status updated	200 OK, status updated	Pass

Total: 23 test cases executed. All Pass. (10 key cases shown; additional cases cover listing CRUD, webhook delivery, and chat responses.)

B. Load and Performance Testing — hey—

Load testing used two command profiles: a baseline of 500 requests at 50 concurrent workers (`hey -n 500 -c 50`) and a stress profile of 300 requests at 200 concurrent workers (`hey -n 300 -c 200`). Three endpoint categories were profiled: high-read (`GET /destinations`), database-intensive (`GET /listings`), and AI-intensive (`POST /itinerary`).

TABLE III. HEY LOAD TEST RESULTS

Endpoint	Conc.	Total	Avg Lat.	P95 Lat.	Errors	Status
GET /destinations	50	500	~95 ms	~180 ms	0	Pass
GET /destinations	200	300	—	—	0	Pass
GET /listings	50	500	~130 ms	~240 ms	0	Pass
POST /auth/login	50	500	~210 ms	—	0	Pass
POST /bookings	50	500	~780 ms	—	0	Pass
POST /itinerary	50	500	~2,100 ms	~3,800 ms	0	Pass

Conc. = concurrent workers. POST /itinerary latency reflects external Gemini API call; expected bottleneck.

Data endpoint performance is well within acceptable thresholds — under 300 ms average at 50-concurrent baseline for all read-heavy routes. The AI endpoint is the dominant throughput constraint at ~2,100 ms average. That bottleneck was expected. It is inherent to external API calls, not to the Express backend. Zero 5xx errors appeared under baseline load, and zero errors appeared on the `GET /destinations` endpoint even under the 200-concurrent stress profile.

C. Security Testing — Burp Suite—

Security testing proxied the browser through Burp Suite's intercepting proxy. Site map was populated via manual browsing across all three user roles (Tourist, Provider, Admin), then targeted probes were run through Burp Repeater against four vulnerability classes.

TABLE IV. BURP SUITE SECURITY TEST SUMMARY

Vuln. Type	Attack Vector	Result	Protection Mechanism
IDOR	Modify booking ID to another user's in Repeater	Not Exploitable	Ownership check in Express: 403 Forbidden
Stored XSS	Inject <script>alert('K')</script> in review/listing fields	Not Exploitable	React JSX escaping neutralizes payload on render
JWT Manipulation	Modify payload role: tourist → admin; re-encode without signing	Not Exploitable	Express JWT middleware: 401 Unauthorized
Webhook Spoofing	POST modified body with incorrect HMAC-SHA256 signature	Not Exploitable	Node.js crypto HMAC verification: webhook rejected, incident logged

Total: 47 attack vectors tested. Confirmed vulnerabilities: 0.

No confirmed vulnerabilities across 47 attack vectors. The IDOR probe confirmed that Express ownership checks operate at the handler level, not only on the UI. The JWT tampering result confirms that the middleware rejects any token whose signature does not match, regardless of payload content. The webhook spoofing result is the most significant: Razorpay payment events can be synthesized by an attacker who knows the order structure, but the HMAC-SHA256 check at step 13 of Algorithm 2 blocks any spoofed event before it reaches the booking update logic.

VI. CONCLUSION

The artisan weaving *Sohrai* motifs near Rajrappa Temple does not have a digital storefront. The local guide who knows every trail through Hazaribagh Wildlife Sanctuary has no verified listing, no booking system, no way to receive payment without showing up in person. Aarohan was built to change that — not as a proof of concept but as a deployed, tested system.

Testing validated the build: 23 Postman functional test cases passed without a single failure, zero security vulnerabilities emerged from 47 Burp Suite probes across four attack classes, and load testing showed zero errors at the 50-concurrent baseline across all endpoint categories. The AI endpoint's ~2,100 ms average latency is the known throughput constraint — a property of the external Gemini API call, not of the platform infrastructure. System uptime measured 99.91% over a 30-day Vercel deployment window.

The platform operates across three domains simultaneously. Economically, tribal artisans and local guides gain a verified digital presence and a direct income channel — reducing their dependence on informal contact networks. Experientially, tourists gain reliable pre-visit information: AI itineraries, 360° VR previews via Photosphere.js (loading in ~3.2 seconds on 4G), and a booking system that does not require prior personal knowledge of a provider. From a governance perspective, administrators gain a Grafana analytics dashboard with real-time footfall trends and revenue data to inform policy decisions — data that did not exist before.

The current version carries real limitations. Gemini API's support for Santali — a scheduled Indic language with its own *Ol Chiki* script — is thin; responses in Santali lack the quality and idiom fluency of Hindi and English outputs. Photosphere.js assets exceed acceptable load times on 3G connections, which remain common in rural Jharkhand. The artisan onboarding flow still requires UX simplification. And load behavior above 200 concurrent users is untested — Vercel's serverless scaling may handle it, but no data exists yet.

Planned future work addresses these gaps directly: PWA implementation with offline tile caching for remote areas; UPI as a first-class payment method via the Razorpay UPI flow; fine-tuned Santali NLP for the chatbot; a React Native mobile application reusing existing component logic; and integration with government transport APIs for state bus schedule data. Aarohan is a working system. The next task is scaling it.

VII. ACKNOWLEDGMENT

Dr. Mayank Patel's guidance on serverless architecture trade-offs and scope boundaries was instrumental in keeping the system focused on deployable outcomes rather than theoretical completeness. The Department of Computer Science and Engineering at Geetanjali Institute of Technical Studies, Udaipur, provided the infrastructure and institutional support for this work. The Government of Jharkhand's Department of Higher and Technical Education supplied the original problem statement through the Aarohan challenge brief, which defined the community-first mandate that shapes every design decision in this platform. The authors thank all three.

VIII. REFERENCES

[1] Department of Tourism, Government of Jharkhand, "Year-wise tourist inflow data in Jharkhand (2013–2022)," Ranchi: Dept. of Tourism, 2023. [Online]. Available: https://tourism.jharkhand.gov.in

[2] OECD, "Artificial intelligence and tourism," Report at the G7 Tourism Ministers' Meeting, Paris: OECD Publishing, Nov. 2024.

[3] Ajay Maru, Ajay Kumar Sharma, Mayank Patel, "Hybrid Machine Learning Classification Technique for Improve Accuracy of Heart", Proceedings of the Sixth International Conference on Inventive Computation Technologies [ICICT 2021], 2021, IEEE Xplore Part Number: CFP21F70-ART; ISBN: 978-1-7281-8501-9, pp. 1107-1110

[4] S. Chourasia et al., "Exploring the potential of augmented reality and virtual reality on Indian tourism industry," Gurugram Univ. Bus. Rev., pp. 40–49, 2023.

[5] D. Gursoy and R. Cai, "Artificial intelligence: An overview of research trends and future directions," Int. J. Contemp. Hosp. Manag., vol. 37, pp. 1–17, 2025.

[6] A. Prasanna, P. Pushparaj, and B. P. Kushwaha, "Conversational AI in tourism: A systematic literature review using TCM and ADO framework," J. Hosp. Tour. Manag., vol. 64, p. 101310, 2025.

[7] N. Samala, B. S. Katkam, R. S. Bellamkonda, and R. V. Rodriguez, "Impact of AI and robotics in the tourism sector: A critical insight," J. Tour. Futures, vol. 8, pp. 73–87, 2022.

[8] Sharma, A., M. Patel, and M. Tiwari. "A comparative study to detect fraud financial statement using data mining and machine learning algorithms." International Research Journal of Engineering and Technology (IRJET) 6.8 (2019): 1492-1495

[9] Patel, M., Aggarwal, A. & Kumar, A. Investigation of Cracking Susceptibility and Porosity Formation and Its Mitigation Techniques in Laser Powder Bed Fusion of Al 7075 Alloy. Met. Mater. Int. 29, 2358–2373 (2023). https://doi.org/10.1007/s12540-023-01387-w

[10] A. Tripathi, P. Dhaundiyal, K. Sharma, J. Sukheswala, A. Dhiman, and A. Sharma, "Improving consumer engagement with AI chatbots: Exploring perceived humanness, social presence, and interactivity factors," in Proc. IEEE ACCAI, 2024, pp. 1–6, doi: 10.1109/ACCAI61061.2024.10601926.

