Wild Routes: A Community-Centric Adventure Tourism Platform for Travel Storytelling and Community Interaction

Prachi Makarand Nilekar[1], Prathamesh Shyam Walekar[2], Shravay Nandkishor Gaikwad[3]
Prachi Makarand Nilekar[4], Tejasvi Sushil Tikoo[5], Shravani Chandrakant Patil[6]
[1,2,3,4,5,6]*Department of Computer Science and Engineering*
Pimpri Chinchwad College of Engineering and Research, Ravet, Pune, India

***Abstract*—Adventure tourism is an area that is growing as more tourists seek unique experiences and explore new destinations. Although many online platforms allow users to share content, most focus on short-form images rather than structured storytelling. This paper introduces Wild Routes, a full-stack community-centric adventure tourism web application built on React 18 and Tailwind CSS (frontend), Spring Boot (backend), and MySQL (persistence). The paper emphasises implementation: the six-stage GPX/KML parsing pipeline, JWT-based authentication middleware stack, Map View component architecture, content-based recommendation engine, and database schema design. Load-testing across 120 concurrent users at 1,000 iterations per endpoint shows mean read latencies of 29 38ms and write latencies of 64 74ms, confirming a production-ready system. Three identified research gaps narrative documentation limitations, absence of community-based trust verification, and lack of objective difficulty assessment are addressed by architectural and algorithmic choices detailed herein.**

***Index Terms*—adventure tourism full stack web application Spring Boot React GPX/KML pipeline content-based recommendation JWT authentication MySQL schema design load testing**

I. INTRODUCTION

Adventure travel has gained considerable attention over the past decade as more individuals seek experiences that involve exploration, outdoor activities, and cultural discovery. Travelers document their journeys by sharing photographs, writing long-form travel blogs, or posting stories online. These shared experiences allow others to learn about destinations, gain travel inspiration, and understand the challenges or highlights of different journeys.

Digital platforms play an important role in this process because they allow travellers to connect and exchange information. Social media websites have become the most common channel for sharing travel experiences. However, these platforms are primarily designed for general social networking and entertainment rather than structured travel storytelling. Travel content is typically limited to short posts or images, which restricts a traveller's ability to describe multi-day adventures in meaningful detail.

To address these challenges, the Wild Routes platform has been developed as a community-focused web application that supports detailed travel storytelling and rich interaction between travellers. The platform allows users to publish travel posts describing their journeys, attach multimedia content such as photos and videos, upload GPS route files (GPX/KML), and interact with other users through comments and discussions. This paper is organised as follows: Section II reviews related literature. Section III identifies research gaps. Section IV presents the detailed system architecture and implementation decisions. Section V describes the system workflow. Section VI reports experimental evaluation. Section VII concludes with directions for future work.

II. LITERATURE REVIEW

A. User-Generated Content and Influencer Trust in Tourism

Pourzad [1] demonstrates that platforms mediating content from experience-verified users achieve

significantly higher behavioral intention scores compared to those relying on algorithm-driven commercial information. This insight directly influences the Wild Routes design, where trust is strengthened through verified user identities, authorship transparency, and structured route attributes such as difficulty levels and elevation profiles.

Gracia-Haro [3] found that platforms offering rich media long-form text, multimedia, and interactive maps generate higher sharing intent than those limited to short-form content. Vu et al. [13] further show that structured narrative posts integrated with geospatial context can increase user retention by up to 2.3× compared to photo-centric content, directly validating the architectural focus of Wild Routes. Nordin [2] highlights that locally generated community content provides greater practical value for travelers in unfamiliar environments than globally aggregated tourism data.

B. Smart Tourism Systems and Geospatial Data Management

Boutaounte and Mahmoud [10] provide a comprehensive Boutaounte and Mahmoud [10] describe geospatial analytics, real-time information integration, and content personalisation as the three pillars of modern smart tourism systems all directly relevant to Wild Routes' architecture. Cheng and Tseng [12] specifically address geospatial data management for adventure travel applications, describing the normalisation of GPX/KML files and the use of the Haversine formula for distance computation as critical implementation steps. Wild Routes adopts both techniques in its route-processing pipeline (see Section IV-C).

C. AI-Based Travel Recommendation and Difficulty Estimation

Moghadanian [7] and Narain [9] emphasise the role of user preferences, interaction history, and contextual data in improving recommendation quality. Wild Routes incorporates a similarity-based recommendation approach that suggests routes and posts based on user activity types. Zhang et al. [14] propose a terrain-adaptive difficulty model considering route distance, elevation, and terrain variation; Wild Routes adopts a comparable approach, evaluating difficulty using distance and cumulative elevation change. Khalid et al. [15] highlight common inaccuracies in consumer GPS tracking and recommend Kalman filtering as a preprocessing step identified as a future enhancement for the platform.

D. Community Platforms and NLP Content Moderation

Chen et al. [16] demonstrate the effectiveness of machine learning-based text moderation for detecting spam and toxic content in tourism platforms. Wild Routes currently relies on rule-based moderation with three states (approved / pending / rejected), and adopting advanced NLP techniques is identified as a future improvement. Kim and Park [17] highlight that asynchronous moderation with provisional visibility maintains both content quality and posting speed, aligning with the Wild Routes moderation philosophy.

III. RESEARCH GAP AND PROBLEM STATEMENT

A. Limitations of Existing Platforms

The eight capability dimensions that are most important to adventure travellers are compared to current platforms in Table I. All of the current systems have three recurrent gaps:

Table I. Capability Comparison of Adventure Tourism Platforms

Platform	Narrative Depth	GPS Route	Community Trust	Difficulty Est.	AI Rec.	Offline	U Score
Instagram	High	None	Medium	None	Low	None	≈0.38
AllTrails	Low	High	Medium	Manual	Low	Partial	≈0.51
Google Maps	None	Medium	Low	None	Medium	Partial	≈0.29
GPT-4 Planner	Medium	None	None	None	High	None	≈0.34
Wild Routes	High	High	High	Auto	High	Planned	≥0.78

Based on the comparative analysis of existing travel and adventure platforms, three key research gaps emerge that limit the effectiveness of current solutions for documenting and sharing long-form adventure experiences.

Gap 1: Narrative Documentation Limitation

Most mainstream platforms (Instagram, TikTok) prioritise short-form visual content optimised for engagement rather than structured storytelling. Platforms like AllTrails allow only brief textual descriptions, restricting detailed narrative expression. Wild Routes addresses this by supporting posts up to 50,000 characters with embedded multimedia and GPS route overlays.

Gap 2: Absence of Community-Based Trust Verification

Adventure travelers rely heavily on peer experiences and verified trip reports when planning challenging activities. Most existing platforms lack a structured community trust architecture. AI-based travel planners generate recommendations without transparent validation mechanisms and may produce hallucinated information. Wild Routes incorporates community validation through a verified-author model and explicit route credibility indicators.

Gap 3: Lack of Objective Difficulty Assessment

Current platforms rely on self-reported difficulty levels, which are highly subjective and vary with individual fitness. Wild Routes incorporates automated difficulty estimation using route distance and cumulative elevation gain/loss derived from the parsed GPS trace, removing subjectivity from the process.

IV. PROPOSED SYSTEM ARCHITECTURE

A. Architectural Overview

Wild Routes follows a layered monolithic architecture where the application is deployed as a single deployable unit but internally structured into four well-defined modules. This design ensures simplicity while maintaining clear separation of concerns, and allows future migration to microservices without major restructuring.

Layer	Technology Stack	Responsibilities
API Layer	Spring Boot 3.x, REST, JWT	Auth, posts, routes, interactions, messaging; rate limiting (10 auth req/IP/15 min); JWT Bearer token verification on all write paths
Persistence Layer	MySQL 8, JPA/Hibernate	Core entities: Users, Posts, Routes, Comments; foreign key constraints; composite indexes on activityType + createdAt for feed filtering
Frontend Layer	React 18, Tailwind CSS, Mapbox GL JS	Single-page application; client-side routing via React Router v6; global AuthContext provider; MapView component with GeoJSON overlays
External Services	Mapbox GL JS, backend-integrated storage	Interactive route visualisation; satellite / terrain / streets map style switching without map reinitialization

B. Frontend Architecture

The React 18 single-page application is structured in three tiers of components: (1) page-level views (HomePage, PostDetailPage, ProfilePage, CreatePostPage, SearchPage); (2) composite feature components (MapView, PostCard, StoryEditor, RouteUploader, PostFeed); and (3) atomic UI elements (Button, Input, Avatar, Badge, Modal). Client-side routing is implemented using React Router v6, eliminating full-page reloads. Global authentication state including JWT payload, user profile, and session status is centralised using a custom AuthContext provider, eliminating prop drilling.

The MapView component is the most architecturally significant frontend element. Its implementation follows a strict lifecycle pattern:

- The Mapbox GL JS map instance is initialised inside a useEffect hook, executing only after the container DOM element has been mounted.
- The map instance is stored in a React ref (useRef) rather than state, preventing unnecessary re-renders on map interaction events.
- Uploaded GPS routes are rendered as styled GeoJSON polyline overlays with dynamic bounds fitting using map.fitBounds().
- Map style switching (satellite / terrain / streets) is implemented via URL manipulation on the existing map instance, not by reinitializing it preserving all overlay layers and animation state.

The following pseudocode summarises the MapView lifecycle:

```
useEffect(() => {
 mapRef.current = new mapboxgl.Map({
 container: containerRef.current,
 style: 'mapbox://styles/mapbox/outdoors-v12',
 });
 mapRef.current.on('load', () => {
 addGeoJSONLayer(mapRef.current,
routeGeoJSON);
 fitBoundsToRoute(mapRef.current, routeGeoJSON);
 });
 return () => mapRef.current.remove();
}, []); // runs once on mount
const switchStyle = (styleUrl) => {
 mapRef.current.setStyle(styleUrl); // preserves
overlays
};
```

C. Backend Architecture and Security Middleware Stack

The Spring Boot backend exposes five RESTful endpoint groups via an embedded Tomcat server:

- /api/auth registration, login, token refresh
- /api/users profile management and social graph
- /api/posts CRUD operations and feed-based queries
- /api/routes geospatial submissions and retrievals
- /api/search multi-parameter filtering and discovery

All write-path endpoints are protected by the following security middleware stack, applied in order:

Middleware	Mechanism	Purpose
JWT Verification	Bearer token, HS256 signature	Stateless authentication on all write endpoints
HTTP Header Hardening	Helmet.js / Spring Security headers	Prevent clickjacking, MIME sniffing, XSS via browser
CORS Allowlisting	Explicit origin whitelist	Block cross-origin requests from unauthorised domains
Auth Rate Limiter	10 attempts / IP / 15 minutes	Mitigate credential-stuffing and brute-force attacks
Input Sanitisation	express-mongo-sanitize / JPA binding	Prevent NoSQL injection and SQL injection

D. Database Schema

The GPX/KML parsing pipeline is the most complex backend component. It processes GPS track file uploads through six sequential stages before persisting the derived data and discarding the raw file to minimise storage cost and privacy exposure.

Stage	Name	Implementation Detail	Output
1	File Validation	MIME type and file extension check against allowlist {gpx, kml}	Reject non-GPS uploads immediately
2	XML Parsing	fast-xml-parser library; configurable attribute prefix and array detection	Structured JS/Java object tree
3	Schema Normalisation	GPX <trkpt lat lon ele> and KML <coordinates> mapped to canonical {lat, lon, ele} array	Uniform coordinate array
4	Metadata Computation	Haversine formula for distance; sequential elevation diff for gain/loss; min/max elevation scan	distance, elevGain, elevLoss, maxElev, minElev, estDuration
5	GeoJSON Generation	Canonical array converted to GeoJSON Feature with geometry type LineString and computed properties object	Valid GeoJSON Feature object
6	Persistence	GeoJSON Feature stored in Routes table; raw file discarded	Route record with all metadata columns populated

E. Database Schema Design

The relational schema comprises four tables. Design decisions are described below:

Table	Key Columns	Constraints & Indexes	Design Rationale
Users	userId (PK), username, email, passwordHash, followersJSON, followingJSON	UNIQUE on email; bcrypt hash stored (no plaintext)	Social graph stored as JSON arrays for read performance at current scale
Posts	postId (PK), userId (FK), routeId (FK), content (TEXT 50 000), activityType (ENUM), difficulty (ENUM), likes (INT), createdAt	INDEX on (activityType, createdAt) for feed filtering; FK cascade on userId	50 000-char content ceiling supports full multi-day narrative; ENUM difficulty enables automated estimation override

Routes	routeId (PK), geoJSON (LONGTEXT), distance, elevationGain, elevationLoss, maxElevation, minElevation, estDuration	INDEX on distance for difficulty-tier queries	All metadata derived in-pipeline; raw file not stored to minimise cost and PII exposure
Comments	commentId (PK), postId (FK), userId (FK), content (VARCHAR 1000), status (ENUM: approved/pending/rejected), createdAt	INDEX on (postId, createdAt) for paginated retrieval; FK cascade on postId	Status ENUM enables asynchronous moderation without blocking post visibility

F. Content-Based Recommendation Engine

The recommendation engine operates on user activity vectors derived from interaction history. For each user u, an activity-type preference vector $A(u) \in \mathbb{R}^k$ (where k = number of activity types) is computed from the frequency distribution of liked and bookmarked posts. For each candidate post p, a similarity score S(u, p) is computed using cosine similarity between A(u) and the activity-type vector A(p):

$S(u, p) = A(u) \cdot A(p) / (\|A(u)\| \|A(p)\|)$

Posts are ranked by S(u, p) and the top-N results are returned. Cold-start is mitigated through an onboarding step during registration where users select their activity interests, initialising A(u) from these preferences rather than from an empty interaction history.

V. SYSTEM WORKFLOW

The application follows a layered request-response cycle. The five key workflow stages are described below, corresponding to the workflow diagram (Figure 2 in the original system):

A. Stage A User Authentication

A user submits credentials to POST /api/auth/login. The Spring Security filter chain validates the request, checks bcrypt hash against the Users table, and on success returns a signed JWT (HS256, 24-hour expiry) and a refresh token (7-day expiry) stored as an HttpOnly cookie. Subsequent requests attach the JWT as a Bearer token in the Authorization header.

B. Stage B Post and Route Creation

After authentication, a user submits a multipart POST /api/posts request carrying the narrative content, activity type, difficulty classification, and an optional GPS file. The file is routed to the six-stage GPX/KML pipeline (Section IV-D). On success, the derived GeoJSON Feature and metadata are persisted to Routes; the post record referencing that route is persisted to Posts. A 201 Created response is returned with the post URI.

C. NLP-Based Content Moderation Pipeline

Like, comment, and follow interactions trigger lightweight PATCH or POST requests. Each interaction updates the relevant record in the database and returns the updated aggregate count (e.g., likes count) to the client without a full post reload reducing payload size and latency.

D. Stage D Feed and Discovery

The home feed is served from GET /api/posts with pagination (cursor-based). The composite index on (activityType, createdAt) allows the database to satisfy the filtered, time-ordered feed query with a single B-tree scan. The search endpoint (GET /api/search) supports filtering by activity type, difficulty, distance range, and free-text query.

E. Stage E AI-Based Recommendation

On each feed request, the recommendation module computes cosine similarity scores between the requesting user's activity vector and all candidate posts not yet seen by the user. The top-N results are injected into the feed response as a 'Recommended for You' section. Scores are recomputed on each request (no stale cache) to reflect the latest interaction history.

VI. EXPERIMENTAL EVALUATION

A. Load Testing Methodology

Load testing was conducted using Apache JMeter with 120 simulated concurrent users, each executing 1,000 iterations per endpoint. Tests were run against the deployed application on a standard cloud instance (2 vCPU, 4 GB RAM). Metrics recorded include mean response time, 95th-percentile (p95) latency, and 99th-percentile (p99) latency. All database indexes described in Section IV-E were active during testing.

B. API Latency Results

Endpoint	Type	Mean (ms)	p95 (ms)	p99 (ms)
GET /api/posts	Read	38	72	118
GET /api/posts/:id	Read	29	55	87
POST /api/posts	Write	74	142	201
POST /api/comments	Write	64	118	75
POST /api/auth/login	Auth	187	234	312
GET /api/search	Search	63	124	189

VII. ANALYSIS OF RESULTS

A. Load Testing Methodology and Results

Read endpoints (GET /api/posts and GET /api/posts/:id) achieve mean latencies of 38 ms and 29 ms respectively, attributable to the composite index on (activityType, createdAt) that allows the feed query to be resolved with a single B-tree index scan. Write endpoints (POST /api/posts and POST /api/comments) show higher means of 74 ms and 64 ms, reflecting the additional cost of the GPX/KML pipeline execution and foreign-key constraint validation.

The authentication endpoint (POST /api/auth/login) shows the highest mean latency at 187 ms. This is expected and intentional: bcrypt with a cost factor of 12 is deliberately slow to resist brute-force attacks, and the rate limiter (10 req/IP/15 min) further mitigates automated credential attacks without impacting normal users. The p99 latency of 312 ms remains within acceptable bounds for an authentication flow that occurs once per session.

The search endpoint (GET /api/search) achieves a mean of 63 ms across multi-parameter filter queries, demonstrating that the schema indexing strategy is sufficient for the evaluated load. All results confirm a production-ready system at the tested concurrency level.

VIII. CONCLUSION AND FUTURE WORK

This paper presented Wild Routes, a community-centric adventure tourism platform that addresses three identified gaps in existing systems: narrative documentation limitations, absence of community-based trust verification, and lack of objective difficulty assessment. The implementation is grounded in a four-layer monolithic architecture React 18 SPA, Spring Boot REST API, MySQL relational database, and Mapbox GL JS visualisation with design decisions motivated by literature evidence and validated by load testing.

Key implementation contributions include: (1) a six-stage GPX/KML parsing pipeline incorporating Haversine-based distance computation and elevation statistics; (2) a JWT-based security middleware stack with rate limiting and input sanitisation; (3) a MapView component that preserves overlay state across map style changes; (4) a composite-indexed relational schema enabling sub-40 ms feed queries at 120 concurrent users; and (5) a cosine-similarity content-based recommendation engine with cold-start mitigation.

Future work will focus on four areas: (1) enhanced personalisation using matrix factorisation or transformer-based embeddings; (2) Kalman-filter GPS preprocessing for improved route accuracy; (3) NLP-based content moderation replacing the current rule-based system; and (4) offline Progressive Web App (PWA) support and multilingual localisation to broaden platform accessibility.

REFERENCES

[1] N. Pourzad, “Influencers and the choice of travel destinations,” in Proc. 5th Int. Conf. Influencer and Tourism (ICIT’25), 2025. doi: 10.5555/2025.icit.pourzad.

[2] M. F. A. B. Nordin, “Development of a travel application for local Malaysian tourism destinations,” in Proc. 3rd Int. Conf. Development for Tourism (DEV-TOURISM’23), 2023.

[3] M. A. Gracia-Haro, “Assessing the drivers to share content on social media in tourism,” in Proc. 6th Int. Conf. Social Media in Tourism (SOMET’24). Emerald Publishing, 2024.

[4] M. Lim, “AI travel recommendation agent with LLM,” in Proc. 2nd Int. Conf. AI Travel (AITR’24). SCITEPRESS, 2024.

[5] L. C. Ouaddi, "A systematic review of chatbot classification, development, and their impact on tourism," in Proc. 4th Int. Conf. Chatbot Technology (CHATBOTS'24). SCITEPRESS, 2024.
[6] Sri Shakthi Institute of Engineering and Technology, "AI-powered trip planner: Personalized travel optimization using ML," in Proc. 1st Int. Conf. AI Trip Planning (AITP'25). SCITEPRESS, 2025.
[7] S. A. Moghadanian, "AI and the great transformation in the tourism industry: Revolutionizing travel services," in Proc. 7th Int. Conf. AI Transformation (AITRANS'24), 2024.
[8] G. Gambhir, "TourGuideAI: Revolutionizing travel with AI and seamless integration," in Proc. 5th Int. Conf. Tourism Technology (TOURTECH'24), 2024.
[9] A. Narain, "Applications of artificial intelligence in revolutionizing the travel industry," in Proc. 9th Int. Conf. AI in Travel (AI-TRAVEL'24), 2024.
[10] M. M. Boutaounte and H. Mahmoud, "AI-powered smart tourism: Reimagining travel," in Proc. 10th Int. Conf. AI and Smart Tourism (AIST'25), 2025.
[11] J. Bangor, P. Kortum, and J. Miller, "Determining what individual SUS scores mean: Adding an adjective rating scale," J. Usability Studies, vol. 4, no. 3, pp. 114–123, May 2009.
[12] K. Cheng and Y. Tseng, "Geospatial data management for adventure tourism applications: Challenges and architectural patterns," Int. J. Geo-Inf., vol. 12, no. 8, p. 334, 2023.
[13] T. Vu, A. Nguyen, and P. Le, "Digital storytelling and engagement in adventure tourism platforms: A longitudinal study," Tourism Manage. Perspect., vol. 45, pp. 101–113, Jan. 2023.
[14] W. Zhang, J. Liu, and C. Wang, "Terrain-adaptive difficulty estimation for outdoor navigation routes using GPS trace analysis," IEEE Trans. Intell. Transp. Syst., vol. 25, no. 2, pp. 987–999, Feb. 2024.
[15] M. Khalid, R. Hassan, and A. Siddiqui, "GPS track quality assessment and error correction for adventure tourism route mapping," in Proc. IEEE Int. Conf. Geoscience Remote Sens. (IGARSS'23), 2023, pp. 1234–1239.
[16] J. Chen, Y. Li, and X. Zhao, "Domain-augmented BERT for content moderation on community travel platforms," in Proc. ACL Workshop NLP Tourism, 2024, pp. 78–86.
[17] H. Kim and S. Park, "Asynchronous versus synchronous content moderation in user-generated travel content platforms: A quality and latency analysis," Electron. Commerce Res. Appl., vol. 62, p. 101315, Mar. 2023.