iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VII **Month of publication:** July 2026

DOI: https://doi.org/10.22214/ijraset.2026.84781

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII Jul 2026- Available at www.ijraset.com

WanderHub: A Community-Driven Full-Stack Web Platform for Discovering India's Undiscovered Tourism Destinations

Shubham Kumar[1], Manish Kumar[2], Akhilesh Kumar[3], Arvind Kumar[4], Md. Naiyer Hoda[5]
[1, 2, 3]*Diploma Student,* [4, 5]*Lecturers, Department of Computer Science & Engineering, Government Polytechnic Muzaffarpur, DSTTE, Government of Bihar, India*

Abstract: ***A large share of India's tourism-related digital traffic is concentrated on a small number of heavily commercialised platforms, leaving numerous culturally and historically significant local destinations largely undocumented online. This paper presents WanderHub, a community-driven, full-stack tourism web platform that allows registered users to discover, contribute, and review lesser-known Indian travel destinations, with all user-submitted listings subject to administrative approval before publication. The system follows the Model-View-Controller (MVC) architectural pattern, with a Node.js and Express.js backend, a MongoDB document store accessed through Mongoose, session-based authentication implemented with Passport.js, and an EJS-templated, glassmorphism-styled frontend. Key engineering contributions include complete CRUD operations for listings and reviews, role-based access control distinguishing guests, registered users, and administrators, server-side schema validation using Joi, and a centralized asynchronous error-handling pipeline. The prototype was evaluated through twelve functional and security test cases spanning registration, authentication, listing management, review management, and administrative access control, all of which produced the expected results. The findings demonstrate the functional feasibility of an open, moderated, community-powered model for local tourism discovery and establish a foundation for planned extensions such as geographic map integration, advanced search and filtering, and cloud-based image upload.***

Keywords: Tourism Platform, Node.js, Express.js, MongoDB, MVC Architecture, CRUD Operations, Passport.js, RESTful API, Community-Generated Content, Role-Based Access Control.

I. INTRODUCTION

Smartphones and mobile broadband have made travel-related information more accessible than at any point in the past, yet the online tourism discovery experience remains dominated by a small number of large, commercially driven platforms. India is home to thousands of culturally rich, historically significant, and naturally beautiful locations that remain largely invisible on mainstream travel websites because those platforms prioritise sponsored and high-margin listings over authentic, community-verified content [1], [3]. Prior research on web-based tourism platforms has explored recommender-driven discovery for marginal territories [1], user-centred, living-lab co-design of regional tourism portals [2], and multi-stakeholder, community-based tourism models [3], but comparatively little published work targets an openly moderated, full-stack platform purpose-built for undocumented destinations in the Indian context. WanderHub is proposed to address this gap. It is a full-stack web application that enables any registered user to add a new destination, attach a description and image, and have it reviewed by an administrator before it becomes publicly visible, combining the openness of user-generated content with a lightweight quality-control layer. The platform is built on a Node.js/Express.js backend connected to a MongoDB database, and its interface is designed around an editorial, glassmorphism-inspired aesthetic intended to feel premium rather than purely functional.

A. Problem Statement

India's tourism sector is estimated to contribute close to 7-8% of national GDP, yet the bulk of tourism-related web traffic and associated economic benefit is concentrated on a small number of highly commercialised destinations. Established digital platforms such as MakeMyTrip, TripAdvisor, and Booking.com tend to rank establishments that pay for visibility above genuinely noteworthy but unsponsored locations, leaving numerous authentic attractions under-visited. Local community members and small-scale tourism operators, in turn, have no low-cost, reliable digital channel through which to showcase destinations in their own regions, which contributes to lost cultural and economic opportunity in rural and semi-urban areas.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII Jul 2026- Available at www.ijraset.com

B. Objectives

- Design and develop a fully functional, responsive tourism web platform using a modern JavaScript-based stack.
- Implement a community-contribution model through which registered users can add new destinations with rich descriptions.
- Build secure, session-based user authentication using Passport.js.
- Enforce content quality through an administrative approval workflow prior to publication.
- Provide complete CRUD operations for listings and reviews.
- Implement server-side validation using Joi together with a centralized Express error-handling pipeline.
- Deliver a responsive, glassmorphism-styled user interface with dynamic flash notifications.

C. Scope and Significance

The current implementation covers user registration, login, and session management; creation, editing, and deletion of listings; a star-rating and comment review system; role-based access separating guests, registered users, and administrators; and server-side validation with structured error handling. Advanced search and filtering, interactive map integration, a dedicated mobile application, and an AI-based recommendation engine are treated as planned extensions rather than features of the present prototype. Academically, the project demonstrates an end-to-end MVC implementation covering authentication, validation, CRUD design, and deployment; socially, it illustrates how a free, moderated, community-powered model can widen the visibility of tourism destinations that would otherwise remain absent from commercial travel platforms.

II. LITERATURE REVIEW

Digital tourism discovery has been studied from several complementary angles: recommender-driven platforms for under-visited regions, participatory or living-lab approaches to portal design, and multi-stakeholder models of community-based tourism. Generosi et al. proposed a recommender-based web platform intended to boost tourism in marginal Italian territories by connecting local guides, event organisers, and restaurateurs with travellers seeking distinctive experiences, emphasising co-creation between residents and visitors [1]. Pucihar et al. described the user-centred, living-lab design of the "GoGorenjska" regional tourism platform, in which tourism-service providers, technology providers, and end users iteratively co-designed the portal through successive workshops [2]. Working from a broader management perspective, George et al. examined community-based tourism as a multi-stakeholder business model, arguing that sustainable local tourism initiatives depend on structures that let community members directly participate in, and benefit from, promotion and delivery [3]. Maurizio and Pattanaro further proposed an end-user development architecture that let domain experts, rather than professional web developers, maintain a route-based cultural-tourism platform using Web 2.0 mash-up tools [4]. Collectively, this literature supports the premise underlying WanderHub: platforms that formally involve local contributors in content creation tend to surface destinations that centralised, commercially ranked platforms overlook.

A. Existing Commercial Tourism Platforms

The global online travel industry is dominated by a small number of large platforms. MakeMyTrip, Booking.com, Airbnb, and TripAdvisor collectively serve hundreds of millions of users annually, and in India specifically, MakeMyTrip and Goibibo control a large share of online hotel and package bookings. These platforms offer sophisticated search, payment, and support infrastructure, but their business models are built around commission-bearing listings. TripAdvisor's review-driven model is philosophically closest to WanderHub, yet it predominantly covers already-popular destinations and hotels, with limited penetration into rural or semi-urban tourism. Table I compares WanderHub against representative existing platforms across four dimensions relevant to community-driven discovery.

Platform	User Contributions	Local Focus	Free Listing	Admin Verification
MakeMyTrip	No	No	No	N/A
TripAdvisor	Partial	No	No	Partial
Airbnb	Yes	No	No	Yes
Google Maps	Yes	Partial	Yes	Partial
WanderHub	Yes	Yes	Yes	Yes

B. Limitations of Current Platforms

- High commission rates (typically 15-25%) charged from accommodation and experience providers, making commercial platforms inaccessible to small operators.
- Commercial bias in search rankings, where paid listings routinely outrank genuine, community-reviewed destinations.
- Negligible coverage of destinations outside major cities and established tourist circuits.
- No structured mechanism for local community members to document previously unlisted tourist spots.
- Overtourism at already-popular destinations, driven by algorithmic recommendations that lack diversity.
- Technical and financial barriers that prevent small tourism businesses from adopting digital tools.

These limitations, taken together, establish a case for a platform that removes commercial listing barriers, enables genuine community contribution, and still applies a fair verification process before publication -- the gap WanderHub is designed to occupy.

C. Related Technologies

Node.js is an asynchronous, event-driven JavaScript runtime well suited to I/O-intensive web applications; its non-blocking model allows a single process to handle many concurrent requests efficiently [5]. Express.js is a minimal, unopinionated web framework for Node.js that provides routing, middleware support, and templating-engine integration, and is the de facto standard for building RESTful APIs in the Node.js ecosystem [6].

MongoDB is a document-oriented NoSQL database whose flexible schema design suits the varied and evolving structure of tourism-destination data, with Mongoose providing an Object-Document Mapper (ODM) layer for schema enforcement and query construction [7], [8].

Passport.js is authentication middleware supporting several hundred strategies; its passport-local-mongoose extension simplifies user-model creation together with password salting and hashing [9]. EJS (Embedded JavaScript Templates) enables dynamic, server-rendered HTML without the added complexity of a full client-side framework, which is appropriate for a content-and-forms-oriented application such as WanderHub [10]. Server-side input validation is handled through Joi, a schema description and data-validation library for JavaScript [11].

III. SYSTEM DESIGN AND METHODOLOGY

A. Feasibility Study

A feasibility study was conducted across three dimensions before development began. Technically, every required technology -- Node.js, Express.js, MongoDB, and Passport.js -- is open-source, free to use, and extensively documented, confirming technical viability.

Operationally, the platform's navigation is intentionally simple, keeping the learning curve low for non-technical users, and its administrative tools are simple enough to be operated by a single moderator. Economically, the stack incurs no licensing cost, and the prototype can be hosted on free-tier services such as Render, Railway, or MongoDB Atlas, making the project financially viable in an academic setting.

B. System Architecture

WanderHub follows the Model-View-Controller (MVC) architectural pattern, illustrated in Fig. 1. Mongoose schemas in the Model layer define the structure and validation rules for the Listing, User, and Review collections and communicate directly with MongoDB.

EJS templates, supplemented by Vanilla CSS and JavaScript, form the View layer; EJS-Mate supplies a layout engine so that shared UI elements such as the navigation bar and footer are defined once and reused across pages. Express.js route handlers form the Controller layer: they receive HTTP requests, invoke the model layer, apply authentication and authorization middleware, and pass data to the view layer for rendering. The application runs as a single Node.js process listening on port 8080; incoming requests pass through a middleware pipeline that handles session parsing, flash-message injection, authentication-status checks, and body parsing before reaching the route handlers.

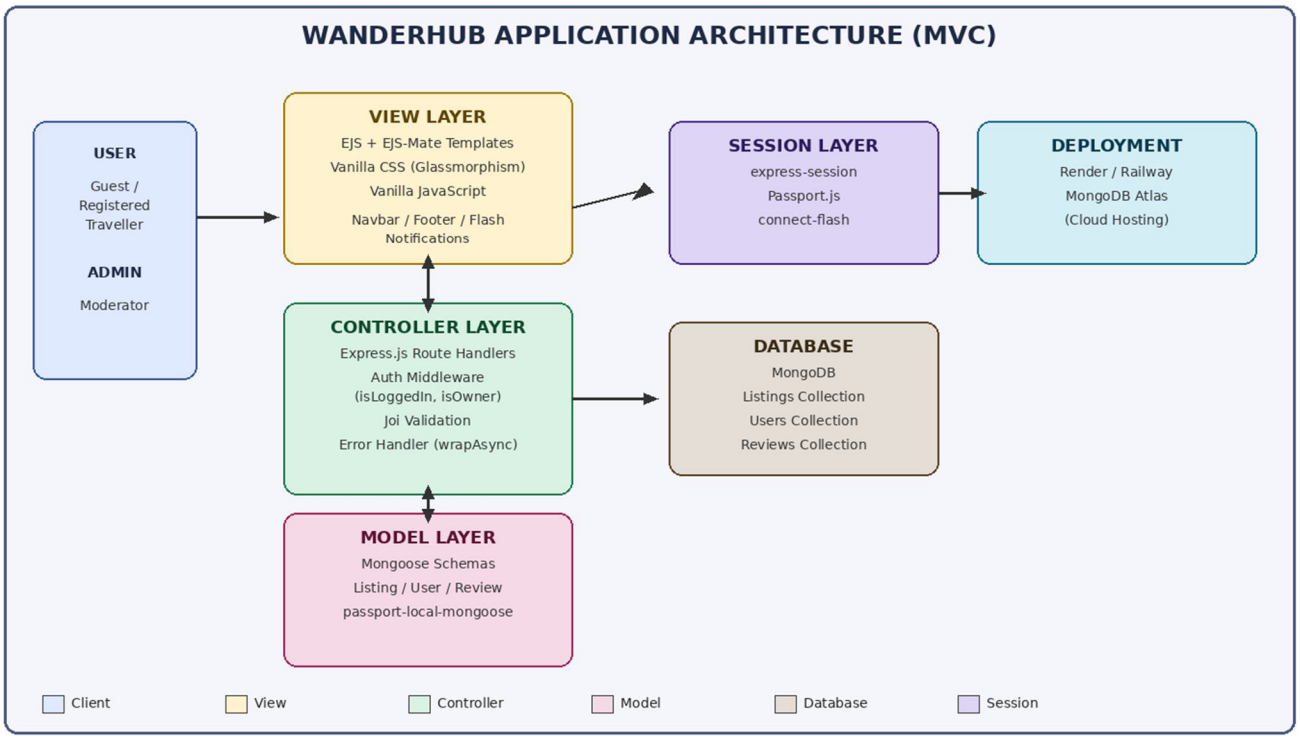


Fig. 1. WanderHub MVC system architecture.

C. Technology Stack

Category	Technology	Purpose
Backend Runtime	Node.js v18	Server-side JavaScript execution
Web Framework	Express.js v4	Routing, middleware, RESTful API
Database	MongoDB + Mongoose v7	Persistent storage & ODM
Authentication	Passport.js + passport-local-mongoose	Session-based user authentication
Session Management	express-session	Server-side session storage
Templating Engine	EJS + EJS-Mate	Server-side HTML rendering
Validation	Joi v17	Server-side schema validation
Frontend	HTML5, Vanilla CSS, Vanilla JS	UI layout, styling, interactivity
Notifications	connect-flash	Flash messages between redirects
HTTP Override	method-override	PUT / DELETE via HTML forms

Table II. Technology stack used in WanderHub.

D. Database Design

WanderHub is backed by MongoDB and accessed through the Mongoose ODM. Three primary collections are defined, summarised in Tables III-V.

Field	Type	Required	Description
title	String	Yes	Name of the destination
description	String	Yes	Detailed description
image	String	No	URL of the destination image
price	Number	No	Approximate visit cost (INR)
location	String	Yes	City / region name
country	String	Yes	Country (default: India)
reviews	[ObjectId]	No	References to Review documents
owner	ObjectId	Yes	Reference to the creating User

Table III. Listing collection schema.

Field	Type	Required	Description
username	String	Yes	Unique login username
email	String	Yes	User email address
hash	String	Auto	Password hash (passport-local-mongoose)
salt	String	Auto	Password salt (passport-local-mongoose)

Table IV. User collection schema.

Field	Type	Required	Description
rating	Number	Yes	Star rating (1-5)
comment	String	Yes	Text body of the review
author	ObjectId	Yes	Reference to the reviewing User
createdAt	Date	Auto	Timestamp of creation

Table V. Review collection schema.

E. Use Case Summary

Use Case	Description
Register / Login	User creates an account or logs in; session initialised by Passport.js.
Browse / View Listings	Any visitor can view paginated, approved destination listings and their detail pages.
Create Listing	Authenticated users submit a new destination with title, description, price, and image URL.
Edit / Delete Listing	Only the listing owner or an admin can modify or remove it.

Use Case	Description
Add / Delete Review	Authenticated users can add a rating and comment; only the author or an admin can delete it.
Admin Approve Listing	Administrators review pending submissions and approve or reject them.
Logout	Session is destroyed and the user is redirected to the homepage.

Table VI. Principal use cases supported by WanderHub.

IV. SYSTEM IMPLEMENTATION

The implemented WanderHub prototype integrates destination discovery, community contribution, review management, and administrative moderation within a single Express.js application. As shown in Fig. 2, the homepage presents an editorial, glassmorphism-styled hero section with a destination search bar, inviting visitors to begin exploring without requiring an account.

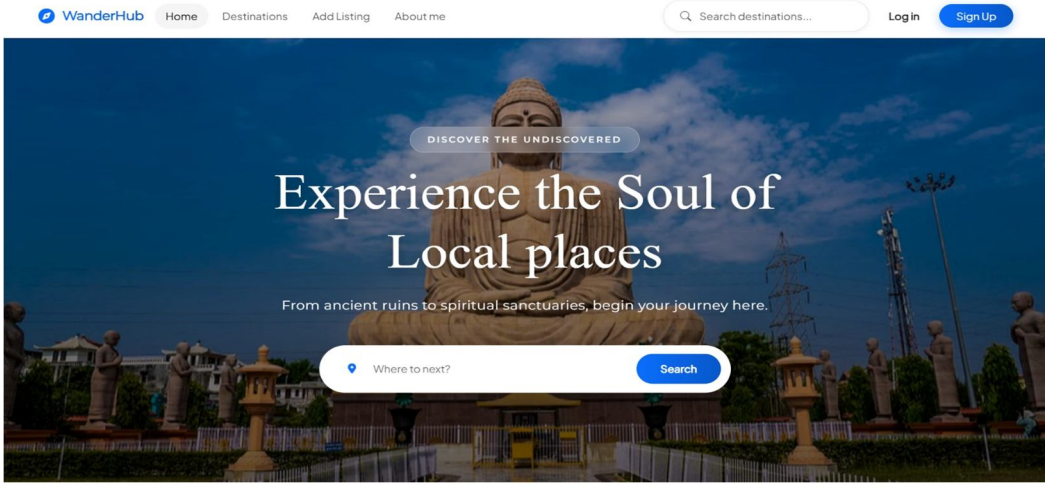


Fig. 2. WanderHub homepage with search-driven destination discovery.

Beneath the hero section, the listings page presents an "Our Recommendations" grid of destination cards (Fig. 3), each combining a representative photograph with a concise, locally grounded description. This layout is designed to give under-documented destinations, such as Varanasi's ghats and Jim Corbett National Park, the same visual prominence as heavily commercialised sites, directly supporting the platform's community-visibility objective.

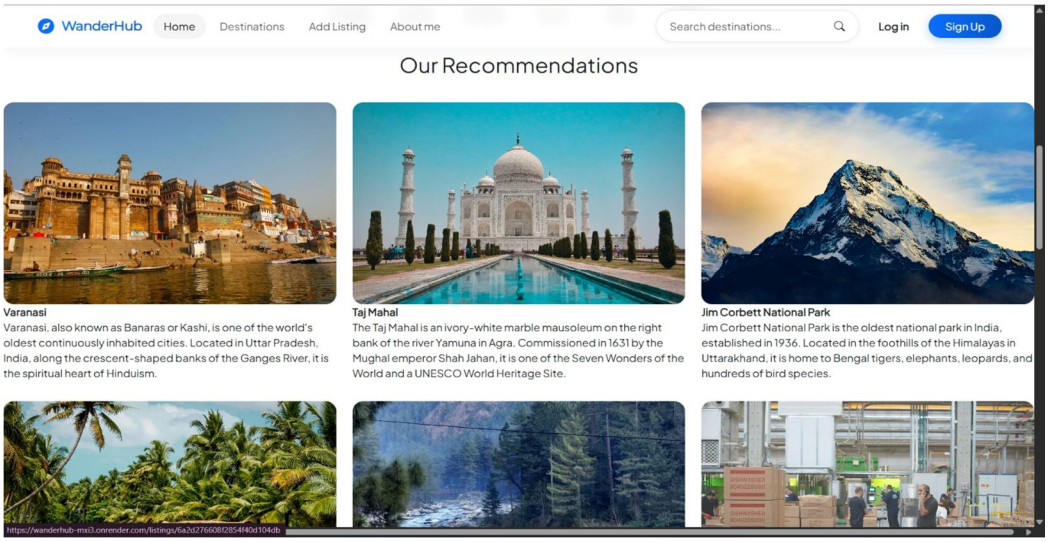


Fig. 3. Destination recommendation grid on the listings page.

Selecting a destination opens its detail page (Fig. 4), which presents the full description, an embedded location map under "Where you'll be," and, further down the page, the review and rating section where authenticated users can add or remove their own feedback. This page is rendered from a single EJS template that adapts its navigation and action buttons according to the visitor's authentication state -- guests see Log in / Sign Up controls, while owners additionally see Edit and Delete actions.

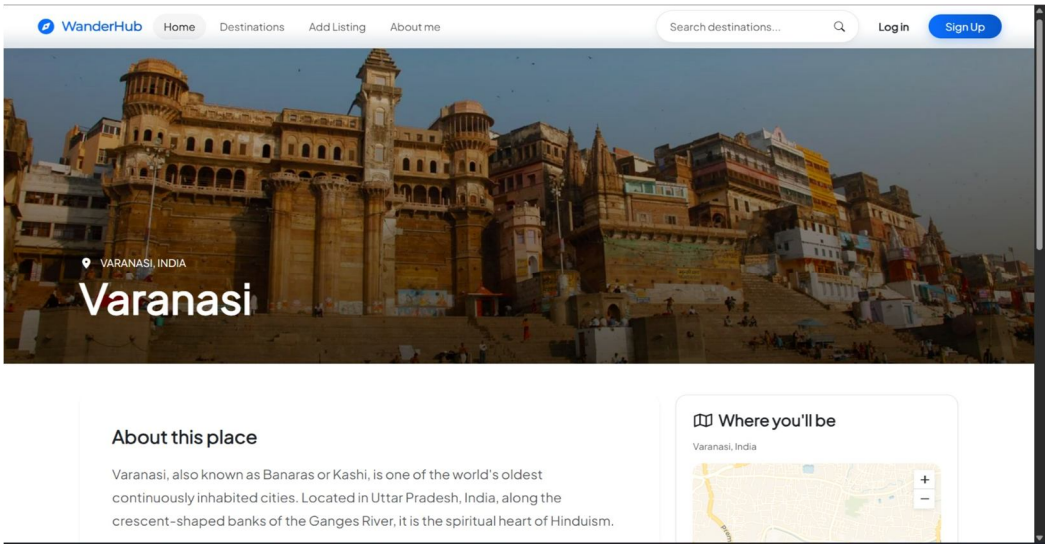


Fig. 4. Destination detail page showing description and location map.

A. User Module

The User Module governs functionality available to guests and registered travellers: browsing and viewing all approved listings; creating, editing, and deleting one's own listings; adding and deleting one's own reviews; and registering, logging in, and logging out with flash-message feedback at every step. Ownership checks are enforced through dedicated Express middleware (isLoggedIn, isOwner, isReviewAuthor) applied to every mutating route.

B. Admin Module

The Admin Module grants elevated privileges to trusted moderators. Administrators can review pending listing submissions and approve or reject them before publication; edit or delete any listing or review regardless of ownership, to correct errors or remove inappropriate content; and, through role-checking middleware, are the only users permitted on sensitive routes, with unauthorised visitors redirected to the homepage with an explanatory flash message.

C. Authentication and Security

- Passport.js with a local strategy verifies credentials against hashed passwords; passport-local-mongoose automatically salts and hashes passwords using PBKDF2.
- express-session maintains authenticated state across requests through a server-side session store identified by an encrypted cookie.
- Custom authorization middleware (isLoggedIn, isOwner, isReviewAuthor) is applied to every route that mutates data.
- method-override lets HTML forms send PUT and DELETE requests, preserving REST conventions without exposing these verbs directly.
- Joi schemas validate every listing and review submission on the server before any database write is attempted.

D. Error Handling and Validation

A custom ExpressError class extends the native JavaScript Error class with an HTTP status code, and a wrapAsync higher-order function wraps every asynchronous route handler so unhandled promise rejections are automatically forwarded to Express's error-handling middleware, eliminating repetitive try/catch blocks. A terminal, global error handler catches all errors, sets the appropriate status code, and renders a user-facing error page. Bootstrap's client-side validation classes provide immediate, in-browser feedback for common input errors ahead of submission.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII Jul 2026- Available at www.ijraset.com

V. TESTING AND RESULTS

The prototype was evaluated through manual black-box functional and security testing covering authentication, listing management, review management, validation, and access control. Twelve test cases were executed; the scenarios and outcomes are summarised in Table VII.

TC#	Test Case	Expected Result	Status
TC01	User registration with valid data	Redirect to listings with success flash	Pass
TC02	Registration with duplicate username	Error flash: username taken	Pass
TC03	Login with correct credentials	Session created; redirect to listings	Pass
TC04	Login with incorrect password	Error flash: incorrect password	Pass
TC05	Create listing (authenticated)	Listing saved; redirect to listing page	Pass
TC06	Create listing (unauthenticated)	Redirect to login with flash message	Pass
TC07	Edit listing by non-owner	403 error: not authorised	Pass
TC08	Delete listing by owner	Listing deleted; redirect to index	Pass
TC09	Add review to listing	Review saved; redirect to listing	Pass
TC10	Submit listing with missing title	Joi validation error response	Pass
TC11	Submit review with invalid rating (0 or 6)	Joi validation error response	Pass
TC12	Access admin route as regular user	Redirect with unauthorised flash	Pass

Table VII. Functional and security test cases and results.

All twelve test cases produced the expected outputs, and no critical defects were observed during testing. Minor UI issues noticed during peer user-acceptance testing, primarily related to form-feedback clarity on mobile screens, were resolved before final submission. The application also behaved stably under concurrent manual testing across multiple browser sessions. As with the underlying prototype, these results demonstrate functional correctness and stability rather than measured improvement in visitor traffic or destination discoverability, which would require a controlled field deployment.

VI. LIMITATIONS AND FUTURE WORK

The current prototype relies on user-supplied image URLs rather than direct image upload, which can reduce the visual consistency of listings; integrating a cloud storage provider such as Cloudinary is planned to resolve this in the next iteration. The platform does not yet include an interactive map for browsing destinations geographically, advanced search and filtering by region, price, or category, or a mobile application, all of which were placed outside the scope of the present prototype. Planned enhancements include:

- Advanced search and filtering by region, price range, category, and rating.
- Interactive geographic maps using the Google Maps JavaScript API or Mapbox, including district-level highlights for regions such as Bihar.
- Cloud-based image upload via Cloudinary in place of manually entered image URLs.
- A React Native or Flutter mobile application with offline access and push notifications.
- An AI-powered recommendation engine that suggests personalised itineraries from browsing history and preferences.
- Support for OAuth-based login (Google, Facebook) alongside the existing local authentication strategy.

An additional direction for future work is a controlled field evaluation of WanderHub with real travellers and local contributors, measuring outcomes such as listing growth, review volume, and visitor traffic to previously undocumented destinations, which would extend the present functional validation toward evidence of real-world tourism impact.

VII. CONCLUSION

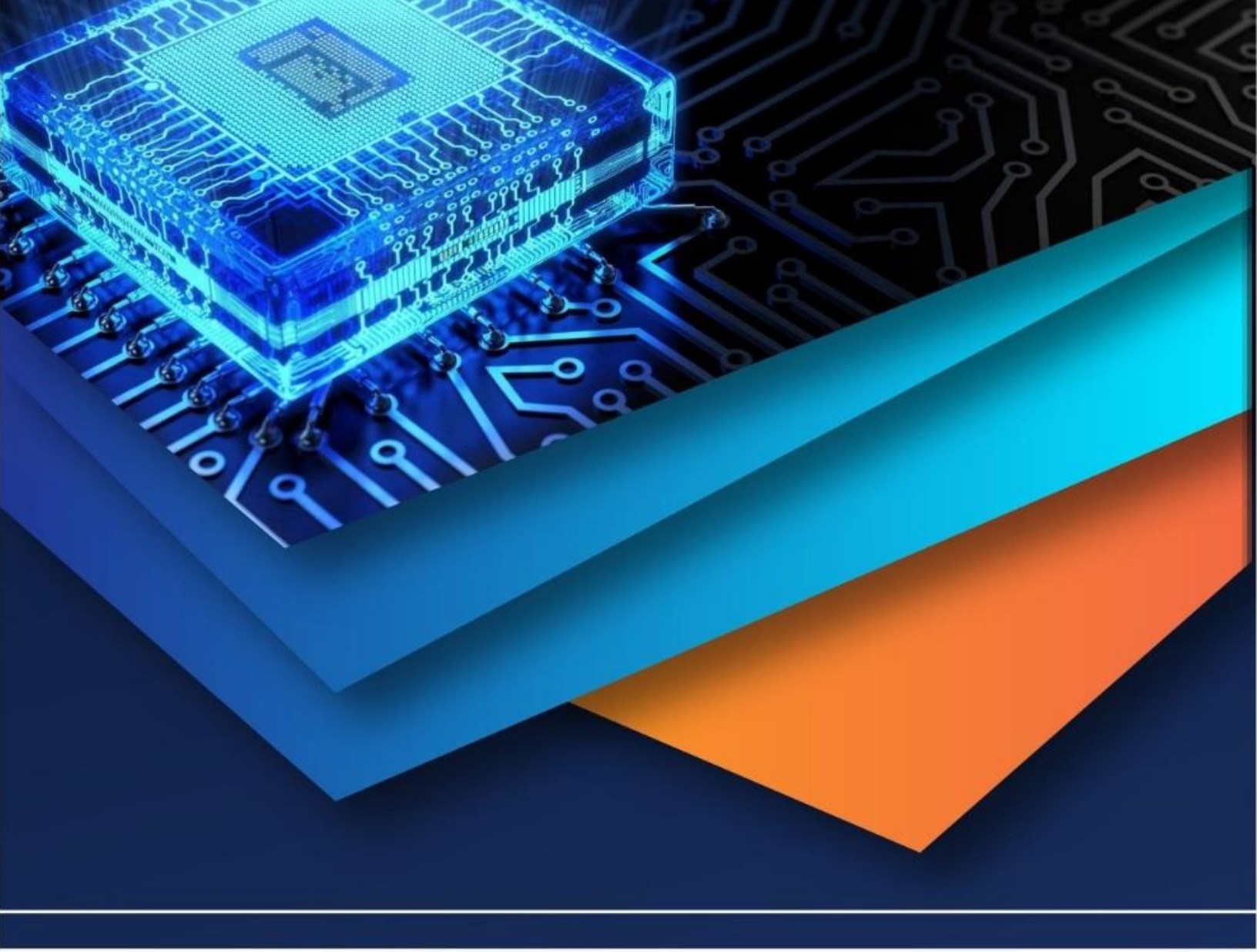
This paper presented the design, implementation, and functional evaluation of WanderHub, a community-driven, full-stack tourism web platform built on the MVC pattern with Node.js, Express.js, MongoDB, and Passport.js. The platform allows registered users to contribute new tourist destinations, subject to administrative approval, and to review existing ones, addressing the visibility gap left by commercially ranked mainstream travel platforms. Twelve functional and security test cases covering authentication, listing management, review management, and role-based access control all produced the expected results, confirming the functional feasibility of the implemented architecture. The contribution of this work is an integrated system design and working prototype rather than measured evidence of tourism impact; the planned extensions -- map integration, advanced search, cloud image upload, and a mobile application -- together with a controlled field evaluation, form a concrete path toward that evidence.

VIII. ACKNOWLEDGMENT

The authors gratefully acknowledge the academic and institutional support provided by the Department of Computer Science and Engineering, Government Polytechnic Muzaffarpur, during the development and evaluation of the WanderHub platform.

REFERENCES

[1] A. Generosi, J. Y. Villafan, M. Ferretti, and M. Mengoni, "A recommender-based web platform to boost tourism in marginal territories," Information Technology & Tourism, vol. 27, no. 3, 2025, doi: 10.1007/s40558-025-00327-1.
[2] A. Pucihar, A. Malesic, G. Lenart, and M. Kljajic Borstnar, "User-Centered Design of a Web-Based Platform for the Sustainable Development of Tourism Services in a Living Lab Context," in Smart Organizations and Smart Artifacts, Lecture Notes in Information Systems and Organization, Springer, 2014, doi: 10.1007/978-3-319-07040-7_24.
[3] B. P. George, A. Nedelea, and M. Antony, "The Business of Community Based Tourism: A Multi-Stakeholder Approach," Journal of Tourism Research, vol. 1, no. 1, pp. 118-139, 2010.
[4] M. Maurizio and G. Pattanaro, "An End-User Development Architecture for Route-Based Tourism in a Web 2.0 Environment," e-Review of Tourism Research, vol. 10, issue 3, 2012.
[5] Node.js Foundation, "Node.js Official Documentation (v18 LTS)." [Online]. Available: https://nodejs.org/en/docs/
[6] OpenJS Foundation, "Express.js Official Documentation (v4.x)." [Online]. Available: https://expressjs.com/
[7] MongoDB, Inc., "MongoDB Manual (v6.x)." [Online]. Available: https://www.mongodb.com/docs/
[8] Mongoose ODM, "Mongoose Official Documentation (v7.x)." [Online]. Available: https://mongoosejs.com/docs/
[9] Passport.js, "Passport.js Official Documentation." [Online]. Available: https://www.passportjs.org/
[10] EJS, "Embedded JavaScript Templating Documentation." [Online]. Available: https://ejs.co/
[11] Joi, "Joi Schema Validation Documentation (v17.x)." [Online]. Available: https://joi.dev/api/
[12] Ministry of Tourism, Government of India, India Tourism Statistics 2023. New Delhi: Government of India Press, 2023.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY