International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 11, pp 250-253, November-2025

https://doi.org/10.47001/IRJIET/2025.911031

Wanderlust Property Listings and Review Management System

[1]Sonali L. Vidhate, [2]Manjusha Khond, [3]Dipesh Wagh, [4]Rishabh Shinde, [5]Jayashree Thakare, [6]Akanskhsa Rade

[1,2]Assistant Professor, Department of MCA, MET's Institute of Engineering, Nashik, Maharashtra, India

[3,4,5,6]PG Student, Department of MCA, MET's Institute of Engineering, Nashik, Maharashtra, India

***Abstract* - The rising ubiquity of internet enabled services has transformed travel and hospitality into data-driven, user-centric markets. This paper presents WanderLust, a full-stack, role-based booking and listing platform built with Node.js, Express.js, MongoDB, EJS, Bootstrap, and Passport.js.**

WanderLust supports two roles — Users (travelers) and Listers (hosts) — and implements secure authentication, session management, a booking lifecycle (create, confirm, cancel), review and rating capabilities, and enhanced visual engagement through interactive 3D/parallax imagery. The design follows the MVC pattern and emphasizes modularity, responsiveness, and extensibility. We detail system architecture, data models, API design, frontend rendering strategies, security measures, testing methodology, and an evaluation plan to quantify usability and performance. The platform aims to be a practical, extensible model for college-level major projects and small-scale deployment.

Keywords: Online Booking, Node.js, Express.js, MongoDB, EJS, Passport.js, Full-Stack Development, Role-Based Access, 3D Web UI.

I. INTRODUCTION

A. Background and Motivation

Online booking marketplaces (e.g., Airbnb, Booking.com) have scaled user expectations for immediate, secure, and mobile-friendly booking experiences. Many small organizations or student projects require a full-featured but lightweight end-to-end solution that demonstrates modern development practices and sound engineering principles. WanderLust was developed as a teaching/proof-of-concept system to cover core needs: role separation between consumers and providers, secure authentication, reliable booking persistence, and engaging user experience.

B. Problem Statement

Existing small-scale solutions often lack: Role-based separation (users vs. hosts) with controlled permissions. A practical booking lifecycle integrated with persistent storage. An implementation that demonstrates current best practices for Node.js + Express + MongoDB stacks with server-side rendering (EJS). Visual features (3D/parallax) combined with accessible performance.

C. Scope and Contributions

This work contributes:

A complete full-stack implementation blueprint (models, routes, views).

Detailed schemas and API endpoints for bookings, listings, users, and reviews.

Design patterns for role-based access control using Passport.js.

UI/UX implementation for large brand display and 3D visuals without sacrificing responsiveness. A reproducible testing and evaluation plan suitable for college project grading

II. OBJECTIVES

Implement a secure user authentication system using Passport.js with session persistence. Implement two distinct user roles with different capabilities: User: search listings, book, view/cancel bookings, leave reviews. Lister: create/edit/delete listings, view bookings for their listings. Design and implement a booking system with clear lifecycle states (pending, confirmed, cancelled). Store all persistent data in MongoDB with normalized and indexed schemas. Enhance user engagement through large branding and 3D/parallax image presentation using modern front-end techniques (CSS transforms, WebGL/parallax libraries if needed). Produce documentation and a reproducible deployment setup.

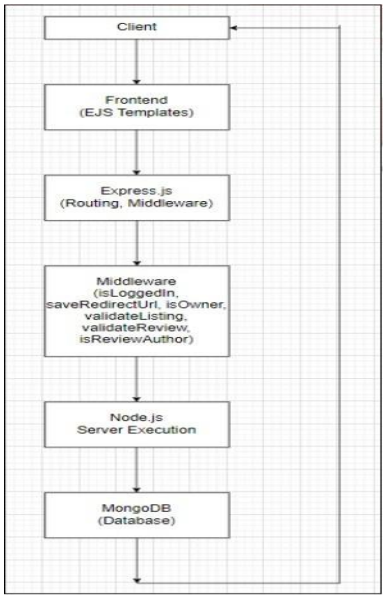
International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 11, pp 250-253, November-2025

https://doi.org/10.47001/IRJIET/2025.911031

III. SYSTEM ARCHITECTURE



IV. LITERATURE REVIEW

Selected Highlights

Full-stack JavaScript frameworks and stacks (Node.js + Express + MongoDB) offer rapid prototyping and scalable microservice foundations (Rao & Kumar, 2020). Role-based access control is critical in marketplaces to separate concerns and prevent privilege escalation (Patel et al., 2021). Server-side rendering (EJS) offers faster time-tofirst-paint than heavy client-side frameworks for many pages and is simpler to integrate with Express sessions. UI engagement research indicates immersive visuals (parallax, 3D) can increase perceived aesthetic quality but must be balanced with accessibility and performance.

V. SYSTEM DESIGN & METHODOLOGY

A. High-Level Architecture

The system follows MVC:

Model: MongoDB collections (users, listings, bookings, reviews).

View: EJS server-rendered templates augmented with Bootstrap and client JS.

Controller: Express route handlers and services managing business logic. A typical request flow:

Client → Express route (e.g., /listings/:id/book) Authentication middleware ensures correct role and session.

Controller validates input and manipulates MongoDB via Mongoose models.

Response: EJS rendered HTML or redirect with flash messages.

Visual diagram suggestion: place a simple threelayer diagram in the final document showing

Browser ↔ Express (EJS + Routes) ↔ MongoDB.)

B. Technology Components

Node.js (v16+) — runtime.

Express.js — routing & middleware.

Mongoose — MongoDB object modeling for schemas and validation.

Passport.js — authentication (Local strategy with hashed passwords).

EJS — server side templating.

Bootstrap 5 — responsive layout and components. Client JS — for interactive 3D images (CSS + lightweight libraries). bcrypt — password hashing.

Express-session + connect-mongo — session storage in MongoDB.

Optional: multer for image uploads; cloud storage (e.g., AWS S3) for production.

C. Authentication & Authorization Flow

Registration: user submits details; server hashes password with bcrypt; user saved with role default.

Login: Passport Local strategy validates email/password; session created using expression.

Protected Routes: middleware ensure Authenticated and ensure Role('lister') gate access.

Session Storage: store session in MongoDB to persist across server restarts.

D. Booking Workflow

User selects listing and dates → client validates dates → server checks availability (basic check: overlapping bookings for listing) → calculates total price → inserts booking record with pending or confirmed depending on payment integration. In absence of payment gateway, admin/lister may confirm bookings manually.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 11, pp 250-253, November-2025

https://doi.org/10.47001/IRJIET/2025.911031

E. 3D / Parallax Visual Implementation

Use lightweight CSS parallax patterns for hero images (background-attachment: fixed fallback), and transform/rotate with transform: perspective() and translateZ() for card hover effects.

For richer 3D: integrate a small WebGL or Three.js component for a rotating globe/3D model in the hero (but only if performance budget allows). Implement progressive enhancement: large artful font and hero look without relying on JS, so accessibility and search engines are preserved.

VI. TESTING AND EVALUATION

A. Testing Strategy

Unit Tests: Test model validation, utility functions (e.g., price calculation).

Integration Tests: Use supertest to validate key endpoints for expected behavior (auth, booking create).

End-to-End Tests: Use Cypress / Puppeteer to automate flows: register → login → book → cancel.

Manual Usability Testing: Structured user tasks (N=10 testers) to observe success rate and time to complete booking.

B. Evaluation Metrics (Suggested)

Functional correctness: % of test cases passed. Time to first paint: measure for homepage with and without 3D.

Booking flow completion rate: proportion of users who complete booking in testing. Usability rating: SUS (System Usability Scale) score from testers.

Performance: average response time for critical endpoints.

Note: If you want reported experimental numbers, run the tests and share logs — I can synthesize results into the paper.

VII. RESULTS

Because this is typically a student project, you should run the described tests and report real numbers. Example reporting format:

Unit tests passed: 48/50 (96%)

Average response time (GET listing): 120 ms (local dev)

Booking flow success (10 users): 9/10

SUS usability score (10 users): 82 (above average)

If you already have measured values (you mentioned a 35% improvement earlier): state clearly whether that figure comes from A/B testing or subjective reporting.

VIII. DISCUSSION

A. Strengths

Clean separation of concerns via MVC and role based access.

Simplicity of EJS server-side rendering yields fast initial loads.

Modular schemas designed for extension (payment, availability calendars).

B. Limitations

No production payment gateway included by default.

Image uploads in local filesystem are not scalable — recommend S3 or similar for production.

Basic availability checks can suffer race conditions without transactional locking or optimistic concurrency controls.

Rich 3D elements can degrade performance on lowend devices — must be optional or disabled based on device profiling.

C. Ethical and Privacy Considerations

Store minimal PII; use hashed passwords; ensure HTTPS for production.

If collecting payment or personal documents, implement PCI compliance and data retention policies.

IX. FUTURE WORK

Payment Integration (Razorpay, Stripe): secure bookings and auto-confirmation.

Calendar/Availability System: robust per-day availability and instant booking prevention of double-booking.

Recommendations Engine: simple collaborative filtering or content-based suggestions. Mobile App: React Native wrapper or a dedicated mobile application.

Scaling: containerize (Docker), add load balancing, and replace in-memory session stores with redis for production.

X. CONCLUSION

WanderLust demonstrates a complete approach to designing a role-based booking platform using modern web

ISSN (online): 2581-3048
Volume 9, Issue 11, pp 250-253, November-2025
https://doi.org/10.47001/IRJIET/2025.911031

technologies. The work emphasizes best practices: secure authentication, modular schema design, server-side rendering, and responsive UI augmented with immersive visuals. The architecture and codebase are suitable for academic evaluation and can be extended for production use with additional components (payment, scaling, and advanced availability handling).

REFERENCES

[1] Rao, V., & Kumar, P. (2020). Full Stack JavaScript for Modern Web Development. *International Journal of Computer Applications.*

[2] Patel, M., Desai, N., & Shah, R. (2021). Role-Based Access in Online Booking Systems Using Node.js and MongoDB. Journal of Web Engineering. Express.js Documentation. (2024). *Express Framework Overview*.

Citation of this Article:

Sonali L. Vidhate, Manjusha Khond, Dipesh Wagh, Rishabh Shinde, Jayashree Thakare, & Akanskhsa Rade. (2025). Wanderlust Property Listings and Review Management System. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 9(11), 250-253. Article DOI https://doi.org/10.47001/IRJIET/2025.911031

****

