PROJECT REPORT

Learning Track: Full Stack Developer (MERN)

Project Name:

FlightFinder - Real-Time Flight Booking System

College Name: *Sri Venkateswara College of Engineering (SVCE), Tirupati*

University Name: *Jawaharlal Nehru Technological University, Anantapur (JNTUA)*

TEAM DETAILS

Team ID: LTVIP2025TMID58576

Team Leader: Dishika Vaishkiyar (22BFA32020)

TEAM MEMBERS

Team Member 1: Gude Reddy Leela (22BFA32036)

Team Member 2: M Lasya (22BFA32062)

Team Member 3: Lavanya Sangaraju (22BFA32060)

Team Member 4: Gali Jeevana Sree (22BFA32030)

1. Introduction

FlightFinder transforms how people book flights by creating a live, responsive web application that mirrors real airline booking systems. Users can search, book, and manage flights while receiving instant updates about their travel plans. Administrators control the entire system, adding flights and pushing immediate notifications to passengers.

Built with React.js, secured by Supabase authentication, and powered by MongoDB for data management, the platform handles everything from user login to seat selection. Every interaction happens instantly no page refreshes, no delays, no confusion. When an admin cancels a flight, affected passengers know within seconds through real-time updates.

The application serves Indian travelers with localized currency (₹) and familiar city options, while maintaining international booking capabilities. From the animated airplane on the homepage to the one-click booking cancellation, every detail focuses on user experience.

2. Project Vision

We set out to solve three critical problems in online flight booking:

Delayed Information: Passengers often discover flight changes hours after they happen. FlightFinder eliminates this gap by pushing updates the moment administrators make changes.

Poor Mobile Experience: Most booking sites fail on smartphones. We designed FlightFinder mobile-first, ensuring smooth operation across all devices.

Complex Interfaces: Airline websites often overwhelm users with unnecessary features. Our interface prioritizes essential functions while maintaining professional aesthetics.

The result is a booking system that passengers trust and administrators control effortlessly. Every feature serves a specific purpose, from email validation that prevents fake accounts to toast notifications that confirm every action.

3. User Roles

Passenger Experience

Passengers start with simple email registration that validates real addresses and prevents spam accounts. The dashboard welcomes them with personalized greetings and immediate access to flight search.

Search functionality covers major Indian cities and international destinations. Results appear instantly with current pricing in rupees, seat availability, and flight status. Booking takes three clicks: select flight, choose seat, confirm booking.

The notification system keeps passengers informed without overwhelming them. Flight delays trigger immediate toast alerts.

Mobile users enjoy identical functionality to desktop users. Touch-optimized buttons, readable fonts, and logical navigation make booking from phones as simple as from computers.

Administrator Control

Administrators access a control panel that displays system-wide booking statistics and flight performance. Adding new flights requires filling dropdown menus pre-populated with city options, aircraft types, and pricing tiers.

Flight status changes happen in real-time. Marking a flight as delayed immediately notifies all booked passengers. Completed flights move to archive while maintaining booking records.

All administrative actions generate audit trails. Changes are logged with timestamps and administrator IDs, ensuring accountability and system transparency.

4. Technology Stack

We chose technologies that prioritize performance, scalability, and developer experience:

React.js powers the frontend with component-based architecture that keeps code organized and maintainable. Hooks manage state without complex class components.

Supabase handles user authentication and session management with built-in security features and JWT token validation.

MongoDB stores all application data with flexible schema design that adapts to changing requirements. Its document-based structure perfectly matches flight booking data patterns.

Real-time Updates are managed through MongoDB Change Streams and WebSocket connections, ensuring instant data synchronization across all connected users.

Tailwind CSS provides utility-first styling that creates consistent designs without custom CSS files. Responsive breakpoints adapt layouts automatically.

React Router DOM manages navigation between pages while maintaining fast load times through single-page application architecture.

5. Core Features

Real-Time Data Synchronization

The application maintains persistent connections between users and MongoDB through Change Streams. When administrators modify flight information, changes propagate to all connected users within milliseconds. This eliminates the frustration of discovering flight changes through email hours later.

Passengers see live seat availability during booking. If another user books the last window seat while someone browses options, the interface updates immediately through MongoDB's real-time data streaming. No double-bookings, no disappointed passengers.

Intelligent Notification System

Toast notifications appear for every significant action: successful logins, booking confirmations, flight updates, and cancellations. Messages are clear, actionable, and automatically dismiss after appropriate viewing time.

The notification center stores message history, allowing users to review past alerts. Unread notifications display with visual indicators until acknowledged.

Authentication Security

Email validation uses pattern matching to identify fake addresses during registration. Users receive immediate feedback for invalid formats with suggestions for correction.

Supabase handles password encryption, session management, and JWT security tokens. MongoDB stores user profiles and preferences with role-based access control that ensures passengers cannot access administrative functions.

Currency Localization

All prices display in Indian rupees with proper formatting. International flights show local currency with rupee conversions. This eliminates confusion and builds trust with Indian travelers.

Responsive Interface Design

Every screen adapts to device capabilities. Phone users get thumb-friendly buttons and simplified layouts. Tablet users see optimized two-column designs. Desktop users access full-featured interfaces with keyboard shortcuts.

6. System Architecture

Authentication Module

Registration validates email formats using regex patterns before creating accounts. Invalid emails trigger helpful error messages with format examples. Successful registration generates welcome emails and automatic login.

Login attempts validate credentials against encrypted database records. Failed attempts limit login speed to prevent brute force attacks. Successful logins redirect users based on assigned roles.

Booking Engine

Flight search queries MongoDB collections with source, destination, and date parameters. Results include real-time availability, current pricing, and aircraft information. MongoDB's indexing optimizes search performance even with thousands of flight records.

Seat selection displays aircraft layouts with available seats highlighted. Premium seats show additional charges. Selected seats are temporarily reserved in MongoDB during the booking process to prevent conflicts.

Booking confirmation generates unique reference numbers, sends confirmation emails, and updates MongoDB collections atomically. Payment processing integrates with secure gateways for transaction completion.

Administrative Panel

Flight management allows administrators to add routes, schedule departures, set pricing, and manage availability. Bulk operations handle multiple flights simultaneously.

Status updates push notifications to affected passengers automatically. Delay notifications include estimated new departure times. Cancellation notices provide rebooking options and refund information.

System monitoring displays booking trends, popular routes, and revenue analytics. Reports generate automatically for daily, weekly, and monthly periods.

Real-Time Communication

MongoDB Change Streams detect data modifications and trigger WebSocket notifications to connected clients. Database changes automatically push updates to affected users without polling or manual refresh.

Message queuing ensures delivery even when users are offline. Notifications appear when users return to the application, maintaining communication continuity through persistent MongoDB storage.

7. Interface Design

Landing Page Experience

The homepage greets visitors with animated airplane graphics that demonstrate the application's dynamic nature. Smooth transitions guide users through feature highlights without overwhelming them.

Call-to-action buttons stand out with contrasting colors and hover effects. Trust indicators display security badges and customer testimonials. The design builds confidence before users commit to registration.

User Dashboard

Personalized greetings create welcoming experiences for returning users. The dashboard balances information density with visual clarity, showing essential data without cluttering the interface.

Flight search occupies prominent screen position with large, clear input fields. Recent searches appear as quick-select options. Booking history displays with status indicators and action buttons.

Administrative Interface

The admin panel prioritizes functionality over decoration. Data tables display clearly with sorting and filtering options. Action buttons use consistent styling and clear labels.

Form inputs validate data entry with immediate feedback. Dropdown menus reduce typing errors and speed data entry. Confirmation dialogs prevent accidental changes to critical flight data.

8. Technical Challenges

Real-Time Performance

Managing multiple simultaneous database connections without degrading performance challenged our architecture. We implemented connection pooling and efficient query optimization to maintain responsiveness.

Memory leak prevention became critical with persistent connections. Proper cleanup of event listeners and database subscriptions prevents browser slowdowns during extended sessions.

Data Validation

Preventing invalid data entry while maintaining smooth user experience required careful balance. We implemented client-side validation for immediate feedback and server-side validation for security.

Email validation patterns needed. The solution accepts various legitimate formats while rejecting common spam patterns.

Security Implementation

Protecting user data and preventing unauthorized access required multiple security layers. Authentication tokens, encrypted communications, and input sanitization work together to maintain system integrity.

Role-based access control ensures users can only access appropriate functions. Administrative privileges are verified for every sensitive operation.

9. Impact and Results

FlightFinder demonstrates that modern web applications can match commercial airline booking systems in functionality while exceeding them in user experience. The real-time notification system solves a genuine problem that frustrates travelers worldwide.

The responsive design works seamlessly across devices, proving that mobile-first development creates better experiences for all users. Touch-optimized interfaces benefit desktop users while serving mobile users properly.

Administrative efficiency improves dramatically with real-time passenger communication. Flight changes that previously required hours of customer service calls now happen automatically with immediate passenger notification.

The technology stack proves that modern development tools can create enterprise-level applications without enterprise-level complexity. React and Supabase enable rapid development while maintaining professional standards.

User feedback highlights the intuitive interface design and reliable performance. Passengers appreciate immediate booking confirmations and proactive flight updates. Administrators value the streamlined management interface and comprehensive reporting.

10. Conclusion

FlightFinder succeeds as both a technical achievement and practical solution to real booking challenges. The application demonstrates that thoughtful design and modern technology can create superior user experiences in traditional industries.

Our team learned that successful applications require more than technical competence they demand understanding user needs, solving real problems, and maintaining high standards throughout development. Every feature decision considered actual user scenarios rather than technical possibilities.

The project validates our approach to modern web development: prioritize user experience, embrace real-time capabilities, design for mobile devices, and focus on essential functionality. These principles create applications that users actually want to use.

FlightFinder represents our commitment to building software that improves people's lives. Air travel involves stress, uncertainty, and significant financial commitment. Our application reduces that stress through clear communication, reliable performance, and respectful design.

The skills developed during this project: full-stack development, real-time systems, responsive design, and collaborative programming prepare us for professional software development careers. More importantly, we learned to create software that serves users rather than impressing developers.

FlightFinder stands ready for real-world deployment and user adoption. The foundation supports future enhancements while delivering immediate value to passengers and administrators. We built something that works, matters, and lasts.