
 FlightFinder  Project Report -----

----- 1. Title FlightFinder – Flight Search and Booking System 2. Student Details Name: [Your Name] Department: ECE / CSE / IT College: [Your College Name] Year: Final Year / Semester X 3. Abstract FlightFinder is a web-based flight booking system that allows users to search for available flights based on source and destination, book flights, and simulate a payment process. It includes full-stack architecture using HTML, CSS, JavaScript, Node.js/Python/Java, and MySQL. 4. Objectives - Provide a user-friendly interface for flight search and booking - Build REST APIs to handle data operations - Implement a relational database schema - Integrate frontend, backend, and database - Perform testing and validation 5. System Architecture Frontend (HTML/CSS/JS) → Backend (Node/Flask/Spring Boot) → Database (MySQL) 6. Modules - Search Flights - Display Results - Booking Form - Payment Simulation - Confirmation Message 7. Tools & Technologies Frontend: HTML, CSS, JavaScript Backend: Node.js / Flask / Spring Boot Database: MySQL Testing: Postman Deployment (Optional): Render, Vercel, Netlify 8. Database Design Tables: flights, bookings, payments ER Diagram: [Flight] ← — [Booking] ← — [Payment] 9. Project Implementation Summary Step-by-step: 1. Setup database (SQL) 2. Create backend APIs 3. Build frontend UI 4. Connect API to frontend 5. Perform manual testing 10. Testing ☒ Valid search returns correct flights ☒ Booking stores user entry in database ☒ Payment confirmed after booking ☒ Invalid input handled gracefully 11. Final Output - Homepage with search box - Flight list - Booking confirmation alert - Payment data saved to DB 12. Advantages - Lightweight and responsive - Easy integration - Realistic booking simulation 13. Limitations - No user login system - No real payment integration - Minimal UI styling 14. Future Enhancements - Add login/signup - Integrate Razorpay/Stripe - Add PDF ticket export - Admin flight dashboard 15. Conclusion The FlightFinder project successfully simulates an airline booking system using full-stack technologies. It meets all requirements, works efficiently, and provides scope for future upgrades. 16. References - MDN Web Docs - W3Schools - Node.js Docs - Flask Docs - Spring Boot Docs 17. Development Model Incremental Waterfall Model – built module by module and tested at each stage. 18. Use Case Diagram (Text) [User] → Search → Book → Pay → Confirm 19. Functional Requirements - Search flights (FR1) - Book selected flight (FR2) - Save booking in DB (FR3) - Simulate payment (FR4) - Display confirmation (FR5) 20. Non-Functional Requirements - Responsive UI - Fast response (under 2 seconds) - Structured codebase - Input validation 21. Screenshots (to be added manually) - Homepage UI - Booking form popup - Console logs / API responses - MySQL DB screenshots 22. Version Info Node.js 18+ MySQL 8+ Flask 2+ / Spring Boot 3+ Postman (latest) 23. Data Flow Frontend → API Request → Backend Controller → MySQL DB → Response to Frontend 24. Output Analysis Flight list shows correctly. Booking and payment data saved. Error cases handled (empty inputs, bad routes). 25. Validation Results Test Case | Status -----|----- Flight Search Valid | ☒ Invalid Booking Input | ☒ handled Booking & Payment | ☒ API Timeout | ☒ error shown 26. Final Result FlightFinder works successfully with clean code, good user experience, and logical data structure. 27. Suggested Improvements - Admin panel for managing flights - Email notification on booking - Real-time price updates - Advanced filters (airline, time, price range) 28. Learning Outcomes - End-to-end web development - Database and API integration - JavaScript fetch and DOM - Error handling and testing - Full project documentation 29. Certificates / Links - [Your GitHub Repo URL] - [Live demo link if hosted] - [Course or certification links] 30. Acknowledgment Thanks to [Guide/Professor Name], classmates, and online communities for support and feedback. -----

---