Project Report On

“AIR CARGO BOOKING & TRACKING”

Submitted in complete fulfillment of the requirement for the award of degree of

Bachelor of Technology

In

Computer Science and Engineering

Batch (2022-2026)

Submitted to:

Dr. Ridhi Kapoor

Assistant Professor

(Department of CSE)

Submitted by:

Saksham Chadha

12200891

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DAV UNIVERSITY

JALANDHAR-PATHANKOT NATIONAL HIGHWAY NH 44,

SARMASTPUR PUNJAB

144012

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my project guide,Dr. Ridhi Kapoor, for their valuable guidance, constructive feedback, and continuous support throughout the duration of this project. I am also thankful to Dr. Rahul Hans (Head of Department) and the Department of Computer Science and Engineering for providing the necessary resources and an encouraging academic environment.

My appreciation extends to all faculty members and classmates whose suggestions and cooperation helped strengthen the quality of this work. Lastly, I am deeply grateful to my family for their constant encouragement and support, which motivated me to complete this project successfully.

SAKSHAM

DECLARATION

I, Saksham Chadha, hereby declare that the work presented in this project titled "Air Cargo Booking and Tracking" is an authentic record of my own efforts. This project has been completed in partial fulfilment of the requirements for the award of the Bachelor of Technology (B.Tech) Degree in Computer Science and Engineering under the guidance of Dr.Ridhi Kapoor (Assistant Professor).

To the best of my knowledge, the content of this report has not been submitted to any other University or Institute for the award of any degree or diploma.

SAKSHAM CHADHA
12200891

Dr. Ridhi Kapoor
Assistant Professor
(Department of CSE)

ABSTRACT

The Air Cargo Booking and Tracking System is a full-stack web application designed to simplify and digitize the process of booking and monitoring air cargo shipments. The system enables users to create bookings by selecting origin and destination airports, checking available flights, and entering cargo details through a streamlined interface. It provides an efficient method for retrieving booking information using a unique reference ID and displays all bookings in a centralized view for easy management. A key highlight of the system is its route visualization feature, which shows all intermediate airports between the origin and destination, allowing users to request landing at any point along the route for flexible cargo delivery. By integrating booking, tracking, route management, and data storage into one platform, the project enhances operational transparency, reduces manual workload, and improves user convenience. This system demonstrates the practical application of modern full-stack development technologies in solving real-world logistics challenges.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The Air Cargo Booking and Tracking System is a full-stack web application designed to simplify and digitalize the process of booking, managing, and tracking air cargo shipments. The project focuses on providing users with an efficient and user-friendly platform where they can create cargo bookings, check flight availability, and track their shipment across multiple connected airports. By integrating booking management with real-time route visibility, the system enhances transparency and improves user experience in cargo transportation.

1.1 Purpose and significance

The purpose of the Air Cargo Booking and Tracking System is to create a centralized digital platform that streamlines the complete cargo booking and monitoring process. It aims to make cargo transportation more organized by allowing users to book shipments, check flight availability, track routes, and manage booking details in a simple and efficient manner. The system also provides an advanced feature where users can request landing or cargo drop-off at any airport that lies along the booked flight route, enabling flexible delivery options. Overall, the project is designed to simplify logistics operations and enhance transparency between the user and the service provider.

The significance the project lies on:

- Improves efficiency by providing a digital platform for booking and tracking cargo, reducing manual work and errors.
- Enhances transparency as users can view all booking details and track the shipment route anytime.
- Offers flexible delivery options through the unique feature of requesting landing at any airport along the selected flight route.
- Saves time for both users and service providers by automating flight search, booking, and tracking processes.
- Increases user convenience with simple pages like Create Booking, Search Booking, All Bookings, and Booking Details.
- Supports better logistics planning by allowing users to see the complete route of the flight and decide suitable delivery points.

1.2 Objectives

- To develop a user-friendly platform for booking air cargo by providing simple forms for flight selection and cargo details.
- To allow users to track their shipment through a dedicated booking reference ID for easy and quick access.
- To display all user bookings in one place for better management and record-keeping.
- To provide route visibility so users can view all intermediate airports between origin and destination.
- To enable landing requests at any intermediate airport along the route, offering flexible delivery options.

1.3 Problem Definition

- Traditional cargo booking processes are manual, time-consuming, and prone to errors, making it difficult for users to manage shipments efficiently.
- Users have limited visibility into available flights, routes, and intermediate airports, leading to confusion while planning cargo deliveries.
- Tracking cargo status is difficult, as users often cannot instantly view booking details or shipment progress.
- No unified platform exists where users can book cargo, view all bookings, and check route information in one place.
- Lack of flexible delivery options, especially the ability to request landing at airports along the flight path, affects timely delivery.
- Data retrieval becomes challenging without a proper digital system, making it hard for users to access booking records or reference information.
- Inefficient communication between users and service providers causes delays and reduces overall operational transparency.

CHAPTER 2

EXISTING SYSTEM

In the existing cargo booking system, most operations are carried out manually or through separate, unconnected tools. Users often have to visit cargo offices, communicate through phone calls, or use basic online forms that do not provide complete information about flights, routes, or booking history. The lack of an integrated digital platform makes the process slow and inconvenient for both customers and service providers.

Cargo tracking is also limited in traditional systems, as users cannot view real-time route information or track shipments using a simple reference ID. Moreover, there is no option for users to request landing or cargo drop-off at intermediate airports, which restricts delivery flexibility. Overall, the existing system fails to provide transparency, ease of use, and efficient cargo management features that modern logistics operations require.

2.1 Current Method

2.1.1 Manual Cargo Booking

In the existing process, cargo booking is mostly done through physical forms or by visiting the cargo office. Users need to manually provide shipment details, choose flights, and confirm availability. This slows down the workflow and increases dependency on staff. It also raises the chances of miscommunication and delayed processing.

2.1.2 Flight Availability Checked Manually

Flight options are generally verified through internal staff or outdated systems. Users cannot check flight schedules or availability on their own. Since this depends on human assistance, it leads to delays during busy hours. The lack of automation makes the process inefficient.

2.1.3 Route Information Not Provided to Users

In the current method, customers do not receive detailed information about flight routes or intermediate airports. They are only given departure and destination details. Without route visibility, users cannot plan flexible delivery options. This creates uncertainty in cargo movement.

2.1.4 Tracking Through Phone Calls or Office Visits

Users must contact customer support or visit offices to know the status of their cargo. There is no instant access to booking details or tracking information. This makes monitoring difficult and time-consuming. It also reduces transparency in the delivery process.

2.1.5 Booking Records Maintained Separately

The existing system stores booking information in separate registers or disconnected digital files. Retrieving past bookings or verifying shipment details becomes slow and complicated. There is no centralized platform to handle all records. This leads to confusion and inefficiencies in data management.

2.2 Limitations of the Existing System

2.2.1 No Automation of the Booking Process

Since the entire process is manual, tasks like checking flight availability, storing details, and retrieving bookings take more time. Manual operations lead to frequent errors and miscommunication. Automation is required to speed up and standardize the workflow.

2.2.2 Lack of Real-Time Tracking

The system does not offer instant tracking or booking visibility to users. They rely solely on staff for updates, which is inconvenient. Without real-time access, users cannot monitor their cargo’s movement. This reduces customer satisfaction and trust.

2.2.3 No Option for Landing at Intermediate Airports

The traditional system does not allow flexible delivery options like landing requests at intermediate airports. Users must stick to the final destination without any customization. This limits operational flexibility and sometimes delays delivery to nearby locations.

2.2.4 Poor Data Management and Record Retrieval

With data stored in scattered files or registers, finding previous bookings becomes tedious. The chances of losing or misplacing records are high. This slows down support services and affects overall system reliability.

2.2.5 Limited Transparency and User Convenience

Users do not get complete information about flights, routes, or cargo status. The absence of a centralized digital system makes the experience less user-friendly. This results in delays, confusion, and lower efficiency for both customers and staff.

CHAPTER 3

PROPOSED SYSTEM

The proposed Air Cargo Booking and Tracking System provides a modern, automated, and user-friendly platform that overcomes the challenges of traditional cargo management. It allows users to create bookings by selecting origin, destination, and available flights, followed by cargo details. The system also offers a dedicated tracking feature where users can instantly check booking status using a reference ID. Additionally, it displays the complete flight route so users can request landing at any intermediate airport for flexible cargo delivery. By integrating booking, tracking, route visualization, and data management into a single system, the proposed solution ensures faster processing, improved accuracy, and greater transparency in air cargo operations.

3.1 System Overview

3.1.1 Digital Cargo Booking Interface

This module allows users to create new bookings by selecting origin, destination, and departure dates. It automatically displays available flights based on user inputs, reducing manual search time. The process is simple and user-friendly, with clear fields for cargo details like weight and number of pieces. By integrating automation, it ensures data accuracy and reduces errors. It forms the foundation for smooth cargo processing.

3.1.2 Booking Reference Search System

This component enables users to retrieve booking details using a unique reference ID. Instead of contacting support, users can instantly view their cargo status. It enhances convenience and provides transparency regarding shipment movement. The system ensures secure fetching of data, preventing unauthorized access. It also speeds up customer service by reducing manual inquiries.

3.1.3 Centralized All Bookings Dashboard

The system offers a page where all created bookings are displayed together. This centralized view helps users and administrators monitor all shipments easily. It ensures efficient data management by organizing bookings in one place. Users can revisit past bookings without searching multiple files. This improves record keeping and operational clarity.

3.1.4 Route Visualization and Airport Mapping

This feature displays the entire flight route including all intermediate airports. It helps users understand how their cargo will travel from origin to destination. The visual route information enables smart delivery decisions. It allows users to plan drop-off or landing requests efficiently. This improves transparency and overall shipment safety.

3.1.5 Landing Request Feature at Intermediate Airports

This unique option lets users request landing at any airport shown on the flight route. It enables flexible delivery of cargo at locations before the final destination. This improves convenience, especially for users closer to intermediate airports. It strengthens the system's usefulness in real-world logistics. It also saves time for both customers and service providers.

3.2 Features of Proposed System

3.2.1 Simple and Interactive Booking Form

The system provides a clean interface for entering airport details, selecting flights, and adding cargo information. The layout is designed to reduce confusion and guide the user step-by-step. Clear input fields ensure accurate booking without mistakes. It enhances the user experience by simplifying complex logistics steps. Overall, it makes cargo booking quick and reliable.

3.2.2 Automated Flight Availability Search

The system automatically fetches flights based on origin, destination, and date. This removes the need for manual flight verification by staff. Automated results allow users to compare options quickly. It saves time and reduces delays during the booking process. It ensures users always receive updated and accurate flight information.

3.2.3 Instant Booking Retrieval Using Reference ID

Users can instantly fetch booking details using their unique reference number. This eliminates the need to contact support staff to check cargo status. The feature improves transparency, as users get real-time access to their shipment details. It also helps in quick corrections or updates if needed. This builds trust and reliability within the system.

3.2.4 Complete Flight Route Display

The system visually displays all airports between origin and destination. This gives users a clear understanding of their cargo's path. It helps in planning delivery strategies and choosing suitable landing points. The route visibility ensures transparency and user control. It also improves safety by keeping users informed throughout the shipment journey.

3.2.5 Digital Record Management

All bookings and cargo details are stored securely in the database. This avoids the risk of losing physical records or spreadsheets. It allows instant access to past and current bookings. The system supports smooth backup and data recovery options. It strengthens long-term reliability of the platform.

3.3 Benefits of Proposed System

3.3.1 Saves Time and Reduces Manual Effort

By automating flight search, booking, and tracking, the system dramatically reduces dependency on manual work. Users complete bookings faster, and staff spend less time managing queries. This leads to quicker operations and improved workflow. It helps in handling more cargo efficiently. The system ultimately increases productivity across the entire process.

3.3.2 Enhances Transparency and Tracking

Users can check booking details and routes anytime without assistance. This ensures full transparency regarding cargo movement. It builds confidence as users are kept informed throughout the shipment process. The system eliminates confusion caused by manual communication gaps. It improves the user’s trust and overall experience.

3.3.3 Offers Flexible Delivery Options

With the landing request feature, users can choose intermediate airports for delivery. This allows faster and more convenient cargo handling. It adds a level of customization not present in traditional systems. Users can save time and transportation costs. It greatly improves operational flexibility.

3.3.4 Reduces Errors Through Digitization

Digitally collecting and storing information eliminates mistakes caused by handwritten or manually processed data. The system validates inputs to ensure accuracy. It helps avoid

duplicate bookings or incorrect cargo entries. This enhances data integrity and ensures smooth workflow. It also reduces administrative workload.

3.3.5 Centralized Data for Easy Access

All booking information is kept in a unified system rather than scattered files. This makes data retrieval faster and more organized. It helps staff and users find information without delays. Centralization improves decision-making by providing complete data visibility. It increases overall operational efficiency.

3.4 Outcome Expectation

3.4.1 Streamlined and Efficient Booking Process

The system automates key booking functions, making the entire process much faster. Users can complete bookings without visiting offices or waiting for staff updates. This speeds up cargo handling and reduces turnaround time. It also improves accuracy by eliminating manual errors. Overall, the booking experience becomes smoother and more efficient.

3.4.2 Improved Cargo Tracking and Transparency

With easy access to booking details, users can track cargo progress anytime. This ensures clarity and builds trust between users and the service provider. It reduces the need for communication with staff. Real-time information helps users plan delivery timings better. This leads to a more satisfying user experience.

3.4.3 Better Route Planning and Delivery Flexibility

Route visualization helps users understand the path of their cargo. The ability to request landing at intermediate airports enhances delivery convenience. This supports better logistics planning for businesses and individuals. It reduces the overall delivery time and increases operational efficiency. It makes the system more practical for real-world logistics.

3.4.4 Efficient Data Storage and Management

All data is stored digitally, making record management faster and more reliable. It improves long-term accessibility and avoids data loss. Staff can easily access past bookings for audits or verification. It supports scalability as the system grows. It ensures that the system remains stable and well-organized.

3.4.5 High User Satisfaction and System Reliability

The combination of flexibility, transparency, and automation increases user satisfaction. Users feel empowered as they have control over booking and tracking. System reliability improves due to fewer errors and faster operations. This contributes to a smooth and dependable experience for all users. It ultimately makes the system suitable for real-time logistics operations.

CHAPTER 4

FEASIBILITY STUDY

A feasibility study is an assessment conducted before developing a system to determine whether the proposed project is practical, achievable, and worth implementing. It evaluates different factors such as cost, time, technology, resources, and operational requirements to ensure that the system can be successfully developed and deployed.

Figure:4.1

4.1 Technical Feasibility

- Checks whether the required technologies (frontend, backend, database, API integration) can support the system.
- Ensures that the platform can handle features like flight search, route mapping, and landing request functionality.
- Confirms that developers have the technical skills for full-stack development.
- Verifies that the server, hosting, and database can handle user bookings securely.

4.2 Economic Feasibility

- Evaluates whether the cost of developing and maintaining the system is justified by the benefits.
- Shows that the system reduces manual work, increases efficiency, and saves operational costs in the long run.

- Confirms that using modern web technologies lowers infrastructure and development expenses.

4.3 Operational Feasibility

- Examines whether the system will work smoothly in a real environment and be accepted by users.
- Ensures the interface is simple for users to create bookings, search bookings, and view routes.
- Checks if the landing request feature meets practical delivery needs.
- Confirms that the system enhances transparency and user satisfaction.

4. Schedule Feasibility

- Determines whether the project can be completed within the planned time frame.
- Ensures that all five modules—Create Booking, Search Booking, All Bookings, Find Routes, and Booking Details—can be developed on schedule.
- Confirms that the timeline for testing, deployment, and documentation is realistic.

5. Legal Feasibility

- Ensures that the system follows guidelines related to data privacy, cargo handling, and digital records.
- Checks that user data and booking information are stored securely and ethically.
- Make sure no rules related to air cargo regulations are violated.

CHAPTER 5

SYSTEM REQUIREMENT SPECIFICATION

A System Requirement Specification (SRS) is a detailed document that describes what a system should do and how it should perform.

It serves as a blueprint that guides developers, designers, and testers throughout the entire project lifecycle.

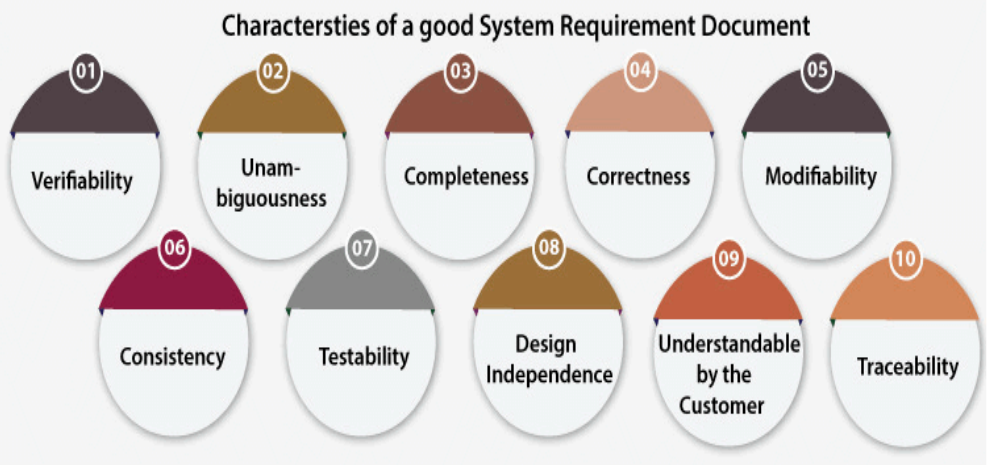


Figure: 5.1

The SRS includes:

- Functional requirements
- Non-functional requirements
- System features
- Constraints
- Assumptions
- Hardware & software needs

For your Air Cargo Booking and Tracking System, the SRS defines the behavior of all pages, the booking process, route tracking, landing request feature, and data management.

5.1 Functional Requirements

- The system must allow users to create a cargo booking by entering origin, destination, and departure date.
- The system must fetch and display available flights based on the user's input.
- Users must be able to select a flight and enter cargo details like number of pieces and weight.

- The system must generate a unique booking reference ID for each booking.
- The search booking page must allow users to check booking details using the reference ID.
- The system must display all bookings created by the user on the “All Bookings” page.
- The system must show complete flight routes with intermediate airports.
- Users must be able to request landing at any airport along the route.
- The system must retrieve and display full booking details on the “Booking Details” page.
- The system must store all data securely in the database.

5.2 Non-Functional Requirements

These describe how the system should perform.

5.2.1 Performance Requirements

- The system should load booking results quickly without delays.
- Data retrieval must be fast even when multiple bookings exist.

5.2.2 Usability Requirements

- The UI must be simple and easy to use for all types of users.
- Navigation between pages must be clear and intuitive.

5.2.3 Security Requirements

- User data must be stored securely in the database.
- Booking information must be protected from unauthorized access.

5.2.4 Reliability Requirements

- The system should work consistently without crashes.
- All bookings must be stored accurately without data loss.

5.2.5 Availability Requirements

- The system should be available for use at all times.
- Users should be able to check bookings anytime using the reference ID.

5.2.6 Maintainability Requirements

- Code should be easy to update, modify, or expand in the future.

- The system must support new features like real-time tracking if added later.

5.3 Hardware Requirements

(For running and testing the system)

- A computer or laptop with minimum 4GB RAM (8GB recommended).
- Processor: Intel i3 or above.
- Stable internet connection.

5.4 Software Requirements

(For development and execution)

- **Frontend:** HTML, CSS, JavaScript, React (optional based on your project)
- **Backend:** Node.js + Express
- **Database:** MongoDB
- **Tools:** VS Code, Postman, Browser (Chrome/Edge), Git
- **Runtime Environment:** Node.js installed
- **Hosting (optional):** Render / Vercel / MongoDB Atlas

CHAPTER 6

DATA FLOW DIAGRAM (DFD)

A Data Flow Diagram (DFD) is a graphical representation used to depict the flow of data within a system. It illustrates how data moves through a system, the processes that transform the data, the data stores where data is held, and the data inputs and outputs. DFDs are useful for understanding and analysing the functional aspects of a system, particularly in the context of system analysis and design. DFD's provide insight into the information flow within a system, helping to understand the processes involved and how data is processed or transformed.

6.1 Symbols of DFD

The symbols used in a Data Flow Diagram (DFD) typically include:

1. Processes:

- Represented by circles or ovals.
- Indicate the transformation or manipulation of data.
- Labeled with a process name that describes the action performed.

2. Data Flows:

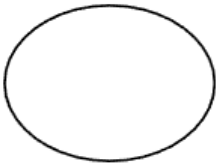
- Represented by arrows.
- Indicate the direction of data movement between processes, data stores, and external entities.
- Labeled with the name of the data being transferred.

3. Data Stores:

- Represented by open-ended rectangles or two parallel lines.
- Indicate where data is stored within the system.
- Labeled with the name of the data store.

4. External Entities:

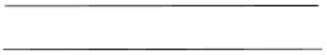
- Represented by rectangles.
- Indicate sources or destinations of data outside the system (e.g., users, external systems).
- Labeled with the name of the external entity.



Process

External Entity

Data Flow



Data Store

CHAPTER 7

TECHNOLOGY USED

7.1 HTML

HTML (HyperText Markup Language) is the core markup language used to design and structure web pages on the internet. It provides a set of predefined tags that allow developers to organize text, images, links, tables, and other multimedia elements. HTML forms the foundation of every website and works together with CSS and JavaScript to create interactive and user-friendly interfaces. Its simplicity, browser compatibility, and platform independence make it essential for front-end web development.

Figure: 7.1

Features of HTML

- **Simple and Easy to Learn:** HTML is a markup language with a straightforward tag-based structure, making it easy for beginners to learn. Each tag has a specific function, such as defining headings, paragraphs, or links, which helps organize content clearly. Its simplicity allows developers to quickly create web pages without deep programming knowledge. Because of this ease of use, HTML forms the foundation for learning web development.
- **Multimedia Support:** HTML supports embedding images, audio, and video directly into web pages using built-in tags like <img>, <audio>, and <video>. This feature allows developers to create interactive and engaging content for users. By supporting multiple media formats, HTML ensures flexibility in web design and enhances the overall user experience.
- **Hyperlinks and Navigation:** HTML provides the `<a>` tag to create hyperlinks, enabling navigation between different pages or websites. This allows developers to build interconnected web structures and improve user experience. Hyperlinks

are a core part of the World Wide Web, making HTML essential for web connectivity and information sharing.

- **Semantic Elements for Structure and SEO:** HTML includes semantic tags such as <header>, <footer>, <article>, and <section> that provide meaning to content. These tags improve accessibility for screen readers and help search engines better understand page content. Semantic structure also makes code cleaner, more maintainable, and professional for developers.
- **Cross-Browser Compatibility:** HTML is supported by all modern web browsers, ensuring consistent display and functionality for users worldwide. Developers do not need to worry about compatibility issues when designing web pages. Standardization by the W3C ensures HTML remains reliable and stable across different platforms.
- **Integration with CSS and JavaScript:** HTML serves as the foundation for styling and interactivity by working seamlessly with CSS and JavaScript. While HTML defines the structure, CSS handles visual design and layout, and JavaScript adds dynamic behavior. This combination allows developers to create responsive, interactive, and visually appealing websites efficiently.

7.2 CSS

CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation and appearance of HTML content on web pages. It allows developers to control the layout, colors, fonts, spacing, and overall visual design of a website. By separating content (HTML) from presentation (CSS), it makes websites more maintainable, flexible, and visually appealing. CSS is essential for creating responsive designs that adapt to different screen sizes and devices.

Figure:7.2

Features of CSS

- **Styling Flexibility:** CSS allows developers to apply colors, fonts, spacing, and borders to HTML elements. This flexibility enables creating visually appealing and professional-looking web pages. By using CSS, one can maintain a consistent style across multiple pages efficiently.
- **Separation of Content and Design:** With CSS, the design and layout are separated from the HTML content. This makes code cleaner and easier to manage. Changes in styling can be applied globally without altering the HTML structure.
- **Responsive Web Design:** CSS supports responsive design techniques, allowing web pages to adapt to different screen sizes, resolutions, and devices. Media queries and flexible layouts make websites user-friendly across desktops, tablets, and smartphones.
- **Advanced Layout Control:** CSS provides powerful layout tools like Flexbox, Grid, and positioning. Developers can control the placement of elements precisely, creating complex and dynamic layouts without using tables or excessive HTML tags.
- **Animation and Visual Effects:** CSS supports animations, transitions, and transformations, enabling interactive and engaging web designs. Effects like hover states, fading, sliding, and rotating elements can enhance user experience without requiring JavaScript.
- **Browser Compatibility and Standardization:** CSS is supported by all modern browsers and follows W3C standards. Developers can create consistent designs across platforms, ensuring a uniform user experience.

7.3 JAVASCRIPT

JavaScript is a high-level, interpreted programming language that enables dynamic and interactive behavior on web pages. Unlike HTML and CSS, which handle structure and styling, JavaScript allows developers to manipulate content, respond to user actions, and control the behavior of web elements in real-time. It is supported by all modern browsers and works alongside HTML and CSS to create fully functional and responsive websites. JavaScript is widely used for tasks such as form validation, animations, interactive maps, and server-side programming with environments like Node.js.

Figure:7.3

Features of JavaScript:

- **Client-Side Scripting:** JavaScript primarily runs on the client-side (browser), reducing server load and improving page responsiveness. This allows immediate feedback to user actions without reloading the page.
- **Dynamic and Interactive:** It enables the creation of dynamic web pages where content can change in response to user interactions. Examples include updating content, hiding/showing elements, and interactive forms.
- **Lightweight and Easy to Learn:** JavaScript has a simple syntax similar to other programming languages, making it easy for beginners to pick up. It is lightweight and does not require heavy resources to run on browsers.
- **Platform Independent:** JavaScript works across all major operating systems and browsers without modification, making it highly versatile for web development.
- **Rich Built-in Libraries and Frameworks:** JavaScript has numerous libraries (like jQuery) and frameworks (like React, Angular, and Vue.js) that simplify development and provide advanced features for building modern web applications.
- **Event Handling:** JavaScript can respond to various user events such as clicks, mouse movements, key presses, and form submissions. This allows interactive user experiences and real-time web page updates.
- **Object-Oriented Programming (OOP) Support:** JavaScript supports object-oriented programming concepts like objects, inheritance, and encapsulation, allowing modular and reusable code development.
- **Integration with HTML and CSS:** JavaScript seamlessly integrates with HTML and CSS to manipulate web page content, style elements dynamically, and enhance interactivity. This makes it an essential part of front-end web development.

7.4 BOOTSTRAP

Bootstrap is a popular open-source front-end framework used for designing responsive and mobile-first websites. It provides a collection of pre-designed HTML, CSS, and JavaScript components, which makes web development faster and easier. Developers can create visually appealing layouts, forms, buttons, navigation bars, and other UI elements without writing extensive custom code. Bootstrap follows a grid system that ensures web pages automatically adjust to different screen sizes and devices, making it highly efficient for modern web development.

Figure:7.4

Features of Bootstrap

- **Responsive Grid System:** Bootstrap uses a 12-column grid system that helps developers create responsive web layouts. It allows content to adjust automatically across desktops, tablets, and mobile devices.
- **Pre-designed Components:** Bootstrap provides ready-to-use components like buttons, forms, navigation bars, modals, alerts, and carousels. These elements save time and maintain design consistency across the website.
- **Cross-Browser Compatibility:** Bootstrap is compatible with all modern browsers, including Chrome, Firefox, Safari, and Edge. Developers can ensure a consistent look and feel for all users.
- **Mobile-First Approach:** Bootstrap is designed with mobile devices in mind. Its responsive features ensure that websites work seamlessly on small screens before scaling up to larger devices.
- **Customizable and Extensible:** Developers can customize Bootstrap's styles, components, and themes according to project requirements. It also allows integration with third-party plugins for additional functionality.

- **CSS and JavaScript Integration:** Bootstrap comes with built-in CSS classes for styling and JavaScript plugins for interactive features like dropdowns, sliders, and modals. This reduces the need for writing custom code.
- **Consistency:** By using Bootstrap, developers can maintain uniform styling and layout across different pages of a website, ensuring a professional and cohesive look.
- **Open Source:** Bootstrap is free and open-source, which allows developers to use, modify, and distribute it for any project without licensing restrictions.

7.5 REACT

React is an open-source JavaScript library developed by Facebook for building modern, interactive, and dynamic user interfaces. It focuses on creating reusable UI components, which makes web applications more efficient and easier to maintain. React uses a virtual DOM (Document Object Model) to update only the parts of the webpage that change, resulting in faster performance. It is widely used for developing single-page applications (SPAs) and supports integration with other libraries and frameworks for full-stack development.

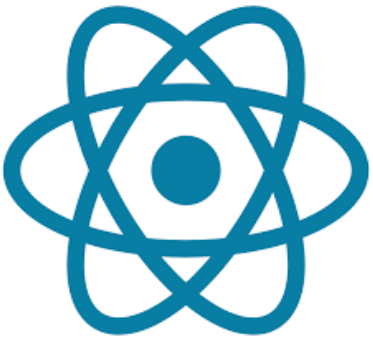


Figure:7.5

Features of React

- **Component-Based Architecture:** React follows a component-based approach, where the UI is divided into small, reusable pieces. This makes code modular, easier to maintain, and promotes reusability across different parts of the application.
- **Virtual DOM:** React uses a virtual DOM to efficiently update and render only the components that change, instead of reloading the entire page. This results in faster performance and smoother user experiences.

- **JSX Syntax:** React uses JSX (JavaScript XML), which allows writing HTML-like code within JavaScript. This makes the code more readable and simplifies the process of creating dynamic UIs.
- **One-Way Data Binding:** React implements one-way data binding, meaning data flows in a single direction. This makes the application more predictable, easier to debug, and reduces the chances of errors.
- **Declarative UI:** React allows developers to describe how the UI should look for different states. The library automatically updates the view when data changes, making development more intuitive and efficient.
- **Reusable Components:** Components in React can be reused across different pages or projects. This reduces development time and ensures consistency in the UI design.
- **Strong Community Support:** React has a large and active developer community. It is continuously maintained, updated, and has a wide range of libraries and tools available for faster development.
- **Integration with Other Libraries and Frameworks:** React can easily integrate with other libraries like Redux for state management or frameworks like Next.js for server-side rendering, enabling full-stack web application development.

7.6 TAILWIND CSS

Tailwind CSS is a modern, utility-first CSS framework used for designing responsive and highly customizable user interfaces. Unlike traditional CSS frameworks that provide predefined components, Tailwind offers utility classes that allow developers to style elements directly in the HTML. This approach gives greater flexibility, faster development, and precise control over the design. Tailwind is highly popular for building modern web applications, supporting responsive design, theming, and integration with tools like React, Vue, and Angular.

Figure:7.6

Features of Tailwind CSS

- **Utility-First Approach:** Tailwind provides small, reusable utility classes for margins, padding, colors, typography, and more. Developers can build custom designs directly in HTML without writing complex CSS.
- **Responsive Design:** Tailwind includes responsive utilities that allow developers to adjust layouts, font sizes, and spacing for different screen sizes easily. This ensures mobile-first and adaptive designs.
- **Customization and Theming:** Tailwind is highly customizable through its configuration file. Developers can define colors, fonts, spacing, and breakpoints to match the project's design requirements.
- **Faster Development**: With pre-built utility classes, Tailwind speeds up development by reducing the need to write custom CSS. It allows rapid prototyping and easy modification of designs.
- **Component-Friendly:** Tailwind integrates seamlessly with component-based frameworks like React, Vue, and Angular. It allows developers to style components efficiently while keeping the code modular.
- **Built-in Plugins and Extensions**: Tailwind supports plugins and third-party extensions that add advanced functionality, such as forms, typography, and animations, without additional custom CSS.
- **Consistency and Maintainability:** Using utility classes promotes consistent spacing, color, and typography across a project. It also improves maintainability since all styling is visible in the HTML structure.
- **Small File Size with Purge:** Tailwind can remove unused CSS classes during production builds, resulting in smaller file sizes and faster page load times.

7.7 MONGODB

MongoDB is a widely used open-source NoSQL database designed to manage large, unstructured, and semi-structured datasets efficiently. Unlike traditional relational databases that store data in rows and tables, MongoDB uses a flexible, document-oriented model. Each record is stored as a BSON (Binary JSON) document, allowing fields to vary in type, size, and structure. This makes MongoDB highly adaptable for applications that require fast development cycles and frequent changes in data formats.

MongoDB is well-known for its ability to handle big data and high-traffic workloads due to its distributed architecture. It supports horizontal scaling through sharding, which distributes data across multiple servers to maintain performance as data grows. This scalability makes MongoDB a preferred choice for modern web, mobile, IoT, and cloud applications. Additionally, its integration with JavaScript-based technologies like Node.js makes it a core component of popular stacks such as MERN and MEAN.

Beyond being flexible and scalable, MongoDB provides powerful features like automatic replication, advanced indexing, aggregation pipelines, and real-time analytics. These features enable developers to perform complex queries, maintain high availability, and ensure data reliability even during system failures. With a growing ecosystem and seamless compatibility with cloud environments, MongoDB continues to be a leading choice for modern database solutions.

Figure:7.7

Features of MongoDB

1. Flexible Document-Oriented Model: MongoDB stores data in JSON-like documents rather than rigid tables, allowing each record to have its own unique structure. This flexibility eliminates the need for predefined schemas and makes it easier to store complex, nested data such as arrays and objects. Developers can quickly modify the data model without restructuring entire databases, making it ideal for rapidly evolving applications.

2. High Scalability with Sharding: MongoDB supports horizontal scaling through sharding, a method of distributing data across multiple machines. As the dataset grows, MongoDB automatically spreads the load, ensuring smooth performance even with millions of records. This approach reduces bottlenecks and allows applications to scale efficiently in cloud-based environments.

3. High Availability with Replication: MongoDB uses replica sets to maintain multiple copies of data across different servers. If the primary server fails, a secondary server automatically takes over without any manual intervention. This ensures continuous availability, improved reliability, and protection against data loss, making MongoDB suitable for mission-critical applications.

4. Powerful Query and Aggregation Framework: MongoDB provides an expressive query language that supports filtering, sorting, indexing, text search, and complex data analysis. The aggregation pipeline allows developers to process data in multiple stages—similar to data processing workflows—making it easy to generate analytics, summaries, and reports directly within the database.

5. Seamless Integration with Modern Technologies: MongoDB integrates smoothly with modern programming languages and frameworks, especially JavaScript through Node.js. It is an essential part of full-stack development environments like MERN (MongoDB, Express, React, Node.js) and MEAN, enabling fast, scalable web application development. Its compatibility with cloud platforms like MongoDB Atlas further enhances deployment flexibility.

7.8 NODE.JS

Node.js is an open-source, cross-platform JavaScript runtime environment used for building fast, scalable, and efficient server-side applications. It allows developers to run JavaScript outside the browser, enabling full-stack development using a single programming language. Node.js is built on Google's V8 JavaScript engine, which compiles code directly into machine language, resulting in extremely high performance and fast execution.

One of the biggest advantages of Node.js is its event-driven, non-blocking I/O architecture, which makes it ideal for handling large numbers of simultaneous connections. This makes Node.js particularly suitable for real-time applications like chat systems, streaming platforms, cloud services, and IoT devices. It's also widely used in modern web development stacks such as the MERN stack (MongoDB, Express, React, Node.js), where it acts as the backend layer.

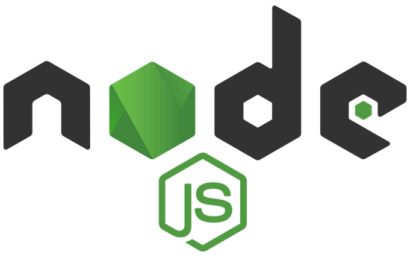


Figure:7.8

Features of Node.js

1. Non-Blocking, Asynchronous Architecture: Node.js uses an event-driven, non-blocking I/O model, meaning it can handle multiple tasks simultaneously without waiting for one task to complete before starting another. This leads to fast processing, efficient resource usage, and improved performance when handling multiple users.

2. High Performance Powered by V8 Engine: Node.js runs on Google's V8 JavaScript engine, which compiles JavaScript into machine code. This makes execution extremely fast and efficient. As a result, Node.js applications often outperform traditional server-side technologies.

3. Single Programming Language for Full-Stack Development: With Node.js, both frontend and backend development can be done using JavaScript. This allows developers to work more smoothly, reuse code, and maintain consistency across the entire application.

4. Rich Ecosystem with NPM: Node Package Manager (NPM) provides millions of ready-to-use modules and libraries. These packages simplify tasks like authentication, database connectivity, input validation, and API creation, significantly reducing development time.

5. Scalability for Real-Time Applications: Node.js is highly scalable and can handle many simultaneous connections, making it perfect for real-time applications such as chat apps, live notifications, online games, and streaming platforms.

6. Cross-Platform Support: Node.js supports multiple operating systems including Windows, macOS, and Linux. Developers can build applications that run consistently across all platforms without major modifications.

7.9 EXPRESS.JS

Express.js is a lightweight, flexible, and fast web application framework built on top of Node.js. It simplifies the process of building server-side applications by providing a set of powerful tools and features for routing, handling requests, managing middleware, and creating APIs. Express allows developers to structure backend logic easily, making it one of the most widely used frameworks for Node.js applications.

Express is especially popular because of its minimalistic and unopinionated design, giving developers complete freedom to organize the project according to their needs. It is the backbone of many modern full-stack web applications and forms an essential part of the MERN and MEAN stacks, where it is used to build RESTful APIs and manage server-to-client communication. Its simplicity, speed, and flexibility make Express a preferred choice for both beginners and professional developers.

Figure:7.9

Features of Express.js

1. Fast and Lightweight Framework: Express provides a minimal and efficient approach to building server-side applications. It does not force any strict project structure, enabling developers to build applications faster with cleaner and more manageable code.

2. Powerful Routing System: Express includes a robust routing mechanism that helps define URLs, handle different HTTP methods (GET, POST, PUT, DELETE), and manage client requests smoothly. This makes building RESTful APIs simple and structured.

3. Middleware Support: One of the core features of Express is its support for middleware—functions that execute at various stages of a request-response cycle.

Middleware helps with tasks like authentication, logging, validation, error handling, and parsing incoming data.

4. Easy Integration with Databases: Express integrates easily with databases such as MongoDB, MySQL, Firebase, and PostgreSQL. It works especially well with MongoDB through libraries like Mongoose, making backend development faster and more efficient.

5. Highly Customizable and Flexible: Express does not impose any strict rules on how applications should be built, allowing developers to customize every part of the server. This flexibility makes it suitable for small apps as well as large enterprise-level systems.

6. Supports Template Engines: Express works with template engines like EJS, Pug, and Handlebars, which help generate dynamic HTML pages on the server side. This is useful for applications that mix backend logic with UI rendering.

CHAPTER 8

SCREENSHOT

8.1 Create Booking

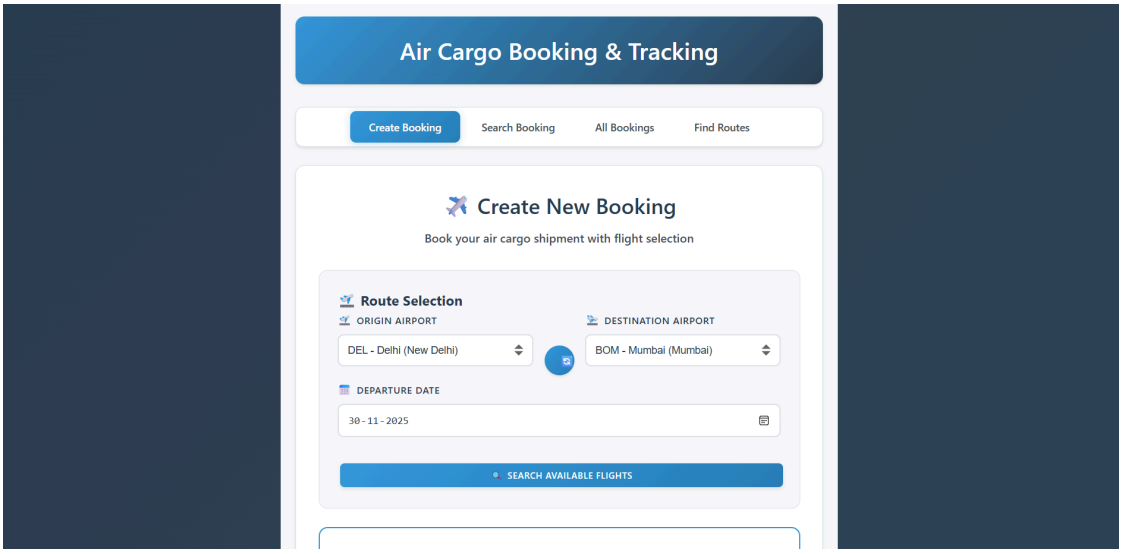


Figure: 8.1

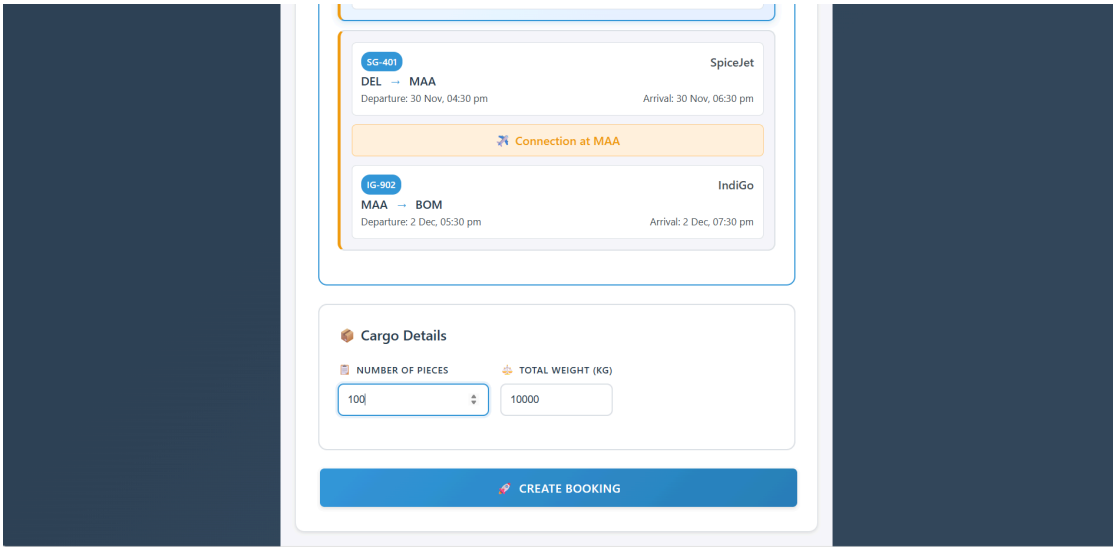


Figure:8.2

8.2 Search Booking

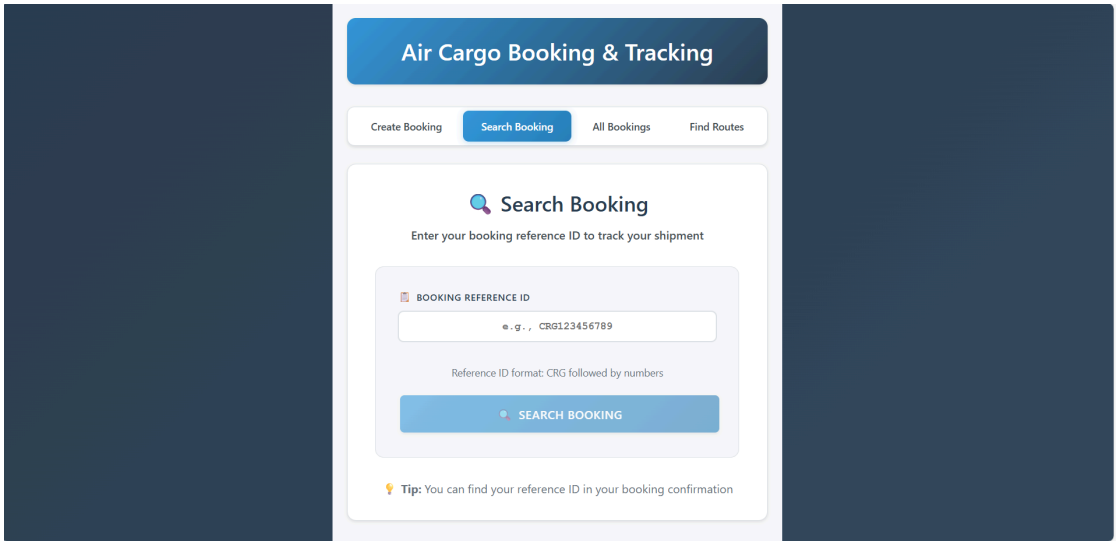


Figure:8.3

8.3 All Bookings

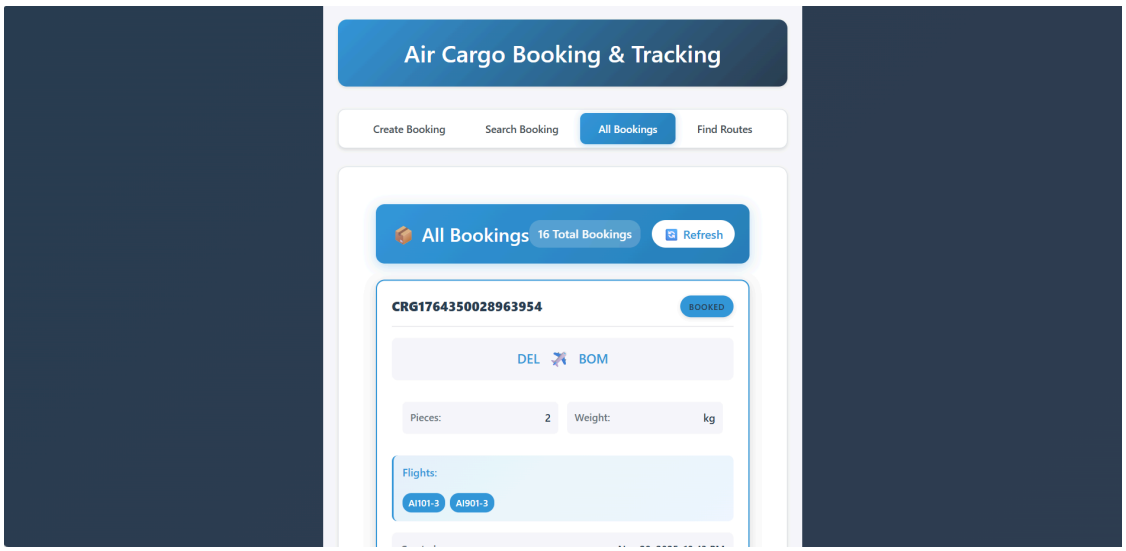


Figure:8.4

8.4 Find Routes

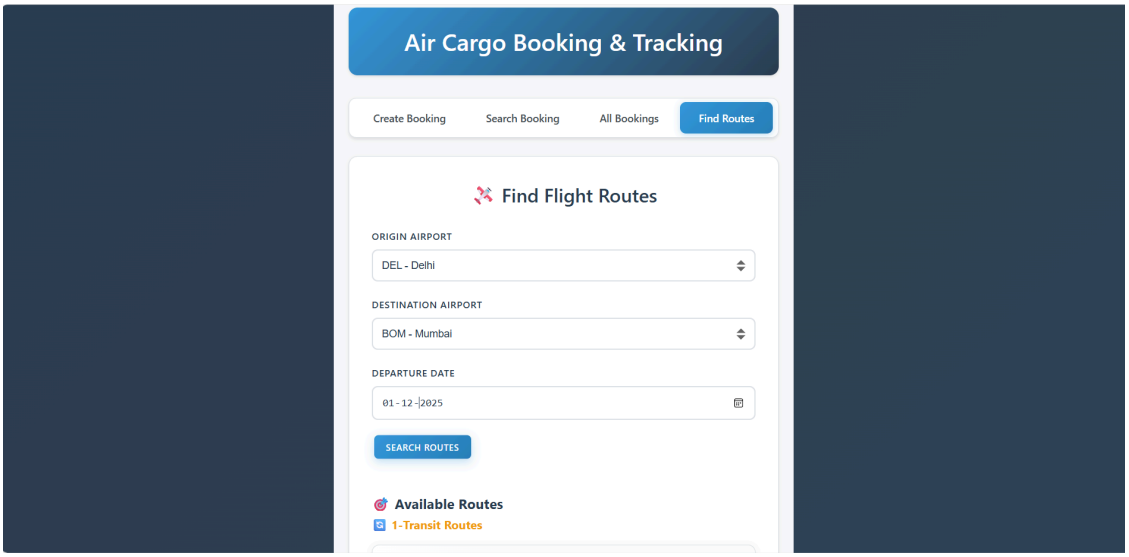


Figure:8.5

8.5 Booking Details

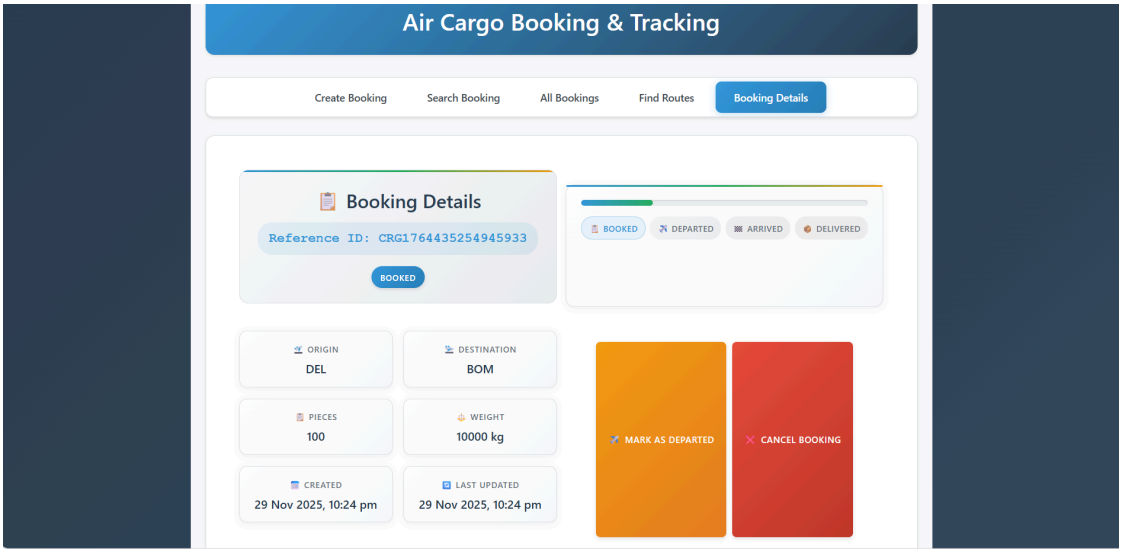


Figure:8.6

CHAPTER 9

SYSTEM TESTING

System Testing is a critical phase in the development of the Air Cargo Booking and Tracking System, where the entire application is tested as a complete, integrated solution. In this stage, all modules—Create Booking, Search Booking, All Bookings, Find Routes, Landing Request, and Booking Details—are combined and tested end-to-end to ensure the system works exactly as defined in the requirements. This testing verifies that the system performs correctly in real-world conditions, behaves consistently across all pages, and handles user inputs, database operations, and workflows without failures. System Testing also ensures that the booking flow, flight route display, cargo tracking, and landing request functionalities operate together smoothly as a fully functional product.

In this project, System Testing checks the behavior of every major function, such as booking creation, validating flight availability, storing cargo details, generating unique booking reference IDs, displaying route maps, and retrieving booking information accurately. It also evaluates the system's non-functional aspects such as performance, security, usability, and compatibility across devices and browsers. Since the system handles sensitive logistics information and user-specific cargo details, accuracy and reliability are essential. System Testing ensures that the backend (Node.js, Express), database (MongoDB), and frontend (HTML, CSS, JavaScript) integrate properly and communicate without any errors. It verifies that all pages give the correct output, user data is validated properly, and the system remains stable under different scenarios.

9.1 System Testing Process

- Prepare test plan & environment: staging system with the backend, DB (MongoDB), frontend, and any external flight-data/mock APIs. Create test data (sample airports, flights, bookings).
- Smoke tests: verify the build is testable: app loads, Create Booking page opens, booking can be created, search by reference works.
- Execute test suites: run functional and non-functional test cases mapped to requirements.
- Log & track defects: use a tracker (JIRA/GitHub Issues), retest after fixes.
- Regression & E2E: run automated regression on critical flows (booking → route → landing request → booking details).

- Sign-off: when critical/high defects resolved and non-functional targets met, hand over to UAT.

9.2 Type of Testing Performed

9.2.1 Functional Testing

Purpose: Verify each feature works per specification.

Approach: Create positive and negative test cases for each requirement.

Examples / Test cases for Air Cargo:

- Create Booking: valid origin/destination/date → displays available flights → selecting a flight and entering cargo details creates booking and returns unique reference ID.
- Invalid inputs: enter invalid airport code or negative weight → app shows proper validation messages and prevents submission.
- Search Booking: enter existing reference ID → correct details shown; enter non-existent ID → "Booking not found" message.

  Expected: All flows complete correctly; validations prevent bad data.

9.2.2 Unit Testing

- **Purpose:** Verifies individual components or modules of the system in isolation.
- **Explanation:** In this project, each module—like booking creation, flight search, cargo details form, and reference ID generation—is tested independently to ensure it works correctly. Developers write unit tests to validate that inputs produce the expected outputs. For example, a unit test can verify that entering cargo weight and pieces correctly calculates the total weight and stores it in the database. This ensures that small parts of the system function properly before integrating them.

9.2.3 Integration Testing

- **Purpose:** Ensures that different modules or components work together correctly.
- **Explanation:** Once unit testing is done, integration testing verifies communication between modules like Create Booking → All Bookings → Find Routes → Landing Request. For instance, after creating a booking, the integration test checks that the booking appears correctly in "All Bookings" and that the route is displayed accurately. It ensures smooth data flow between frontend, backend, and database, reducing errors in multi-module interactions.

9.2.4 Black Box Testing

- **Purpose:** Tests the system based on requirements without knowing the internal code structure.
- **Explanation:** Testers focus on inputs and expected outputs without examining how the code works internally. For this project, black box testing can be used to verify scenarios like: submitting a booking with valid details returns a reference ID; searching with an invalid reference ID displays an error; landing request updates status correctly. It ensures the system meets user expectations and functional specifications.

9.2.5 White Box Testing

- **Purpose:** Tests the internal logic, code structure, and control flow of the system.
- **Explanation:** Developers or testers examine the source code to ensure all conditions, loops, and branches execute correctly. In this project, white box testing can verify that backend logic for flight availability, cargo weight calculation, route mapping, and landing request handling is correct. It also helps detect hidden bugs, optimize code, and improve reliability of the system.

9.2.6 Grey Box Testing

- **Purpose:** Combines both black box and white box testing techniques.
- **Explanation:** Testers have partial knowledge of the internal workings but test the system from a user perspective. For this project, grey box testing might involve testing the landing request workflow knowing how the backend updates database records while also checking if the route display on the frontend works correctly. It helps find defects related to data flow, security, and integration issues that may not be obvious in black box testing alone.

9.2.7 Regression Testing

- **Purpose:** Ensure new changes don’t break existing functionality.
- **Approach:** Maintain a regression suite that covers critical flows and run it after each significant change; automate where possible.
- **Example Suite Items:** booking creation, unique ID generation, search function, all bookings listing, route display, landing request workflow, booking details.
- **Expected:** No previously passed test fails; if failure, prioritize fix.

9.2.8 End-to-End (E2E) Testing

- **Purpose:** Validate complete user journeys across modules and integrations.
- **Approach:** Simulate real user scenarios from start to finish. Automate high-value E2E flows.
- **Example Scenarios:** User books cargo → system assigns flight → route displayed → user requests landing at intermediate airport → admin confirms landing → booking status updates.
- **Expected:** Workflow completes and state changes persist (e.g., booking status, landing requests).

9.2.9 Performance Testing (Load, Stress, Scalability)

Purpose: Ensure the system meets response-time and concurrency requirements.

Approach: Use tools (JMeter/Gatling) to simulate concurrent users and increasing loads. Define SLAs (e.g., search results < 2s under 200 concurrent searches).

Project tests:

- **Load test:** 100–500 concurrent users searching flights/creating bookings. Measure API response times and DB latency.
- **Stress test:** Gradually increase load until system degrades to identify breaking point and recovery behavior.
- **Scalability:** Validate horizontal scaling of backend (stateless APIs + MongoDB scaling) keeps performance acceptable.

  Expected: System handles expected peak with acceptable latency and recovers gracefully under stress.

9.2.10 Security Testing

Purpose: Identify vulnerabilities and protect booking data and operations.

Approach: Pen tests, OWASP checks, authentication/authorization verification, input validation.

Project tests:

- Verify booking reference IDs are not predictable; ensure proper access control so users cannot view others' bookings.
- Test for injection (NoSQL injection), XSS in input fields (airport names), CSRF protection for landing-request endpoints.

- Ensure secure storage/transit: HTTPS enforced, secrets not exposed, role-based access for admin tasks.

  Expected: No high-severity vulnerabilities; remediate critical issues before release.

9.2.11 Compatibility Testing

- **Purpose:** Ensure the UI & features work across browsers/devices.
- **Approach:** Test on Chrome, Firefox, Edge, Mobile Safari, and common screen sizes; use BrowserStack if needed.
- **Project tests:** Create Booking and Route Visualization should be usable on desktop and mobile (responsive maps, touch interactions).
- **Expected:** No critical UI/functionality break on supported browsers/devices.

9.2.12 Usability Testing

- **Purpose:** Ensure the system is intuitive for target users.
- **Approach:** Conduct moderated sessions with sample users, gather qualitative feedback, measure task completion times.
- **Project tests:** New users should be able to create a booking and request an intermediate landing within a short time; measure confusion points.
- **Expected:** Identify UX improvements; minor iterative changes before UAT.

CHAPTER 10

CONCLUSION

The Air Cargo Booking and Tracking System provides a comprehensive and automated solution for managing cargo bookings, tracking shipments, and visualizing flight routes. By integrating all core modules—Create Booking, Search Booking, All Bookings, Find Routes, and Booking Details—the system ensures seamless end-to-end functionality. Users can easily book cargo, check flight availability, enter shipment details, and retrieve booking information through a secure, centralized platform. A key feature of this system is the Landing Request, allowing users to request intermediate airport landings along the route, adding flexibility and convenience that traditional systems lack.

This project demonstrates the effective use of modern full-stack technologies to create a reliable, user-friendly, and efficient platform. It reduces manual effort, minimizes errors, and improves transparency in cargo management. System testing confirms the application's performance, security, and usability, making it suitable for real-world deployment. Overall, the system not only addresses current operational challenges but also provides a foundation for future enhancements such as real-time flight tracking, automated notifications, and route optimization, making it a scalable solution for modern air cargo logistics.

CHAPTER 11

FUTURE SCOPE

The Air Cargo Booking and Tracking System has significant potential for further development and enhancement to meet evolving industry needs. One future improvement is the integration of real-time flight tracking using GPS or airline APIs, which would allow users to monitor their cargo's location live throughout the journey. This would improve transparency and enable more precise delivery planning. Additionally, automated notifications via email or SMS can be implemented to alert users about booking confirmations, flight departures, arrival updates, and landing requests, making the system more interactive and user-friendly.

Another area of expansion is route optimization using AI or machine learning algorithms, which can suggest the most efficient paths for cargo delivery while considering flight schedules, airport capacities, and potential delays. The system can also be extended to support multiple users and organizations, enabling logistics companies to manage bulk cargo operations through a single platform. Furthermore, integrating payment gateways, advanced analytics, and reporting modules can enhance the system's commercial viability and provide actionable insights for business planning. Overall, the system has the potential to evolve into a fully automated, intelligent, and scalable platform for modern air cargo logistics.

REFERENCES

1. HTML Documentation:https://www.w3schools.com/html/
2. CSS Documentation:https://developer.mozilla.org/en-US/docs/Web/CSS
3. JavaScript Documentation:https://devdocs.io/javascript/
4. React Documentation:https://react.dev/
5. Tailwind CSS:https://tailwindcss.com/docs/installation/using-vite
6. Node.Js :https://nodejs.org/en/learn/getting-started/introduction-to-nodejs
7. Express.Js :https://www.w3schools.com/nodejs/nodejs_express.asp
8. MongoDB :https://www.mongodb.com/docs/
9. GitHub:https://github.com/Saksham-911/AIR-CARGO-BOOKING-TRACKING