AIRLINE TRACKING

MINI PROJECT REPORT

Submitted by

GOKULANAND P (9517202209013)

ARUNKUMAR K (9517202209006)

KANAGABALA R (9517202209023)

in

19AD581 – NOSQL DATABASES

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

MEPCO SCHLENK ENGINEERING COLLEGE

SIVAKASI

NOVEMBER 2024

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

AUTONOMOUS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

BONAFIDE CERTIFICATE

This is to certify that it is the bonafide work of **“GOKULANAND P (9517202209013), ARUN KUMAR K (9517202209006),KANAGA BALA R(9517202209023)”** for the mini project titled **“AIRLINE TRACKING”** in 19AD581 –NOSQL Database during the fifth semester July 2024 – November 2024 under my supervision.

SIGNATURE

Dr. R.Nirmalan, B.Tech,ME,PhD
Assistant Professor
AI&DS Department,
Mepco Schlenk Engg. College, Sivakasi

SIGNATURE

Dr. J. Angela Jennifa Sujana,BE, MTech,PhD
Professor & Head,
AI&DS Department
Mepco Schlenk Engg. College, Sivakasi

ACKNOWLEDGEMENT

I sincerely thank our respected principal **Dr.S.Arivazhagan,** for providing necessary facilities to carry out this work successfully.

I wish to express my sincere gratitude to **Dr.J.Angela Jennifa Sujana B.E, M.TECH , PhD ,** Professor and Head of the Artificial Intelligence & Data Science department for her stimulating support and encouragement for the completion of this project work.

I am grateful to our Project guides and **Dr. R.Nirmalan, B.Tech,ME,PhD ,** Assistant Professor and **Mrs.C.Anitha BE, ME, (PhD),** Assistant Professor(Sr.G).

With deep sense of gratitude, I would like to thank our Project Mentor **Dr. R.Nirmalan** for his insightful comments and valuable suggestions which helped me to complete this project work successfully.

My sincere thanks to our revered **faculty members** and **lab technicians** for their help over this project work.

Last but not the least, I extend my indebtedness towards my **beloved family and friends** for their support which made this project a successful one

ABSTRACT

IATA codes, or International Air Transport Association codes, are a standardized system of three-letter alphanumeric designations that uniquely identify airports and airlines across the globe. This coding system plays a pivotal role in facilitating communication within the aviation industry, significantly improving operational efficiency in various aspects of air travel, including ticketing, logistics, and customer service.Each airport and airline is assigned a distinct IATA code, which serves as a concise identifier that enhances clarity in flight itineraries, schedules, and boarding processes. For travelers, these codes simplify the complexities of air travel, making it easier to navigate through different airline services and airport facilities without confusion.The robust integration of IATA codes into electronic ticketing systems ensures that passengers can seamlessly manage their bookings online, while airlines utilize these codes for effective baggage tracking and flight operations. The use of IATA codes also extends to agencies and organizations involved in aviation, including travel agents and air traffic control, where precise identification is crucial for safety and efficiency.Furthermore, the standardization of IATA codes aids in the globalization of air travel, enabling airlines and passengers to connect across various regions with ease. Despite their simplicity, these codes represent a sophisticated framework that underpins the modern aviation industry, contributing to its safety, efficiency, and accessibility.This abstract emphasizes the significance of IATA codes in the operational logistics of air travel, demonstrating how they are integral to the overall passenger experience and the functionality of the aviation sector as a whole. The continued reliance on and evolution of this coding system will be essential in meeting the demands of future air travel as the industry expands and adapts to new challenges and technologies

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The airline tracking application is a sophisticated web-based platform designed to provide real-time updates and detailed information about flights to passengers and aviation enthusiasts alike. As air travel continues to play a pivotal role in global connectivity, having immediate access to flight statuses—notably regarding arrivals, departures, and delays—has become essential for travelers making informed decisions. This application addresses those needs by offering a user-friendly interface that allows individuals to easily input specific parameters, such as airline name and flight number, to retrieve comprehensive data related to their chosen flights. Underlying the application is a robust Express.js server that efficiently processes requests, rapidly querying a MongoDB database filled with extensive flight information. This data encompasses critical aspects such as expected arrival times, actual arrival times, and current flight statuses, including any delays. The application utilizes EJS templating to dynamically generate informative responses that are displayed back to users in a clear, accessible format. With the integration of real-time data handling and a responsive design, the airline tracking application not only streamlines the process of obtaining flight information but also enhances the overall travel experience by keeping users informed and prepared for their journeys. Whether for personal use or operational purposes, this application stands as a comprehensive tool catered to the needs of modern air travel.

1.1 NODE JS

In the airline tracking application, Node.js serves as the backbone, handling server-side operations and facilitating communication between the user interface and the MongoDB database. When a user inputs their desired flight information, such as the airline and flight number, and clicks the submit button, this action triggers an HTTP request to the Node.js server. Utilizing Express.js, a framework built on top of Node.js, the server listens for incoming requests on a defined endpoint related to flight inquiries. Upon receiving a request, the server executes a callback function that processes the input data and formulates a query to the MongoDB database. This query is designed to retrieve specific flight records based on the user’s input parameters.

Once the database returns the flight data, the server performs essential logic, including calculating any delays by comparing the expected and actual arrival times. This processing is fundamental to providing users with accurate and meaningful feedback. The server subsequently uses the EJS templating engine to create a dynamic HTML response, embedding the retrieved flight information into the webpage context. This response is then sent back to the user's browser, where it is rendered in a user-friendly format. Node.js’s non-blocking, event-driven architecture allows it to handle multiple requests concurrently and efficiently, ensuring rapid response times even under high traffic conditions. This capability is crucial, as it contributes to a smooth user experience by delivering timely updates and enabling seamless interactions with the flight tracking application. In essence, Node.js enables the application to function as a cohesive system that can swiftly retrieve, process, and display flight information, making it an invaluable tool for travelers seeking real-time airline updates.

1.2 MONGODB DATABASE

MongoDB is a leading open-source NoSQL document database that excels in handling large volumes of unstructured or semi-structured data through its document-oriented storage model, where data is stored in BSON (Binary JSON) format as flexible documents within collections. This schema-less design allows developers to dynamically modify data structures without the need for expensive migrations, which is particularly beneficial for applications with rapidly evolving requirements. MongoDB's architecture supports horizontal scaling using sharding, enabling seamless distribution of data across multiple servers, which ensures high availability and performance under heavy loads. Furthermore, it features a rich query language that facilitates complex queries, data aggregation, and indexing, allowing for efficient data retrieval and real-time analytics. MongoDB also incorporates replication through replica sets, providing data redundancy and failover capabilities, making it a reliable choice for mission-critical applications. Its ability to integrate with various data processing frameworks and tools enhances its versatility across diverse use cases, such as content management systems, real-time analytics, IoT applications, mobile applications, and social networking platforms. Overall, MongoDB stands out as a powerful solution for modern software development, enabling organizations to leverage its flexibility and scalability to meet the demands of today's data-driven environments.

1.3 PROBLEM STATEMENT

Online airline tracking systems have significantly streamlined the way travelers monitor flights, but they come with a range of challenges that can hinder their effectiveness. One of the primary issues is data accuracy; flight status information can sometimes be outdated or incorrect, leading to confusion among passengers. This problem is exacerbated by connectivity issues, where poor internet access or disruptions in satellite data can interrupt the real-time updates that tracking services rely on. Furthermore, latency is another concern, as there can be delays in the data being refreshed, resulting in passengers receiving notifications about departures, arrivals, and delays that are not current.The coverage of online tracking services can also be limited, particularly in remote areas where comprehensive tracking may be less reliable. Attempting to draw data from multiple sources, including airlines, airports, and third-party services, can lead to inconsistencies in the information provided. This challenge highlights the dependence on various data sources, where discrepancies between official airline data and third-party tracking applications can create a lack of trust among users.

Moreover, privacy concerns are an increasingly prominent issue; many travelers may feel uneasy about the extent to which their travel details are tracked and monitored, raising questions about data security and personal information safeguards. User interface challenges cannot be overlooked either; not all tracking systems are designed with user-friendliness in mind, and convoluted interfaces can frustrate users trying to access critical flight information.

Technical shortcomings also pose significant risks; software bugs or malfunctions can lead to outages or incorrect data displays, which further compromise the reliability of these tracking systems. In terms of geolocation accuracy, factors like atmospheric conditions or equipment errors may impair the precision of aircraft positioning, leaving users with suboptimal information. Additionally, an overreliance on technology may result in passengers neglecting to verify flight updates directly from the airline, potentially causing them to miss crucial travel information.

Another drawback is the proliferation of fraudulent tracking services; numerous third-party applications may promise reliable tracking but can turn out to be unreliable or, worse, malicious in nature. Integration challenges with other systems, such as airport management and ground services, can also lead to data silos and inefficiencies. Lastly, inconsistency in updates across different platforms can confuse passengers; various tracking apps or websites may offer differing information, leaving users uncertain about the status of their flights. In the context of online airline tracking, MongoDB plays a vital role in addressing several operational challenges through its flexible document-oriented database model and real-time data processing capabilities. Airlines must manage vast amounts of data generated from various sources, including flight schedules, passenger information, baggage tracking, maintenance logs, and real-time flight statuses, all of which can be highly variable and frequently changing. With MongoDB's schema-less design, developers can efficiently store and aggregate this diverse data without compromising on structure, allowing them to integrate new data sources or adjust data formats as needed without complex migrations.

For instance, a flight tracking application can easily accommodate the unpredictable nature of updates caused by delays or cancellations, ensuring that passengers receive accurate, real-time flight information. Moreover, MongoDB's powerful querying and aggregation framework enables airlines to perform intricate analysis, such as predicting delays based on historical data patterns or assessing route performance, which in turn improves operational efficiency and enhances customer experience.

CHAPTER 2

DATA MODEL

2.1 SCHEMA DESIGN

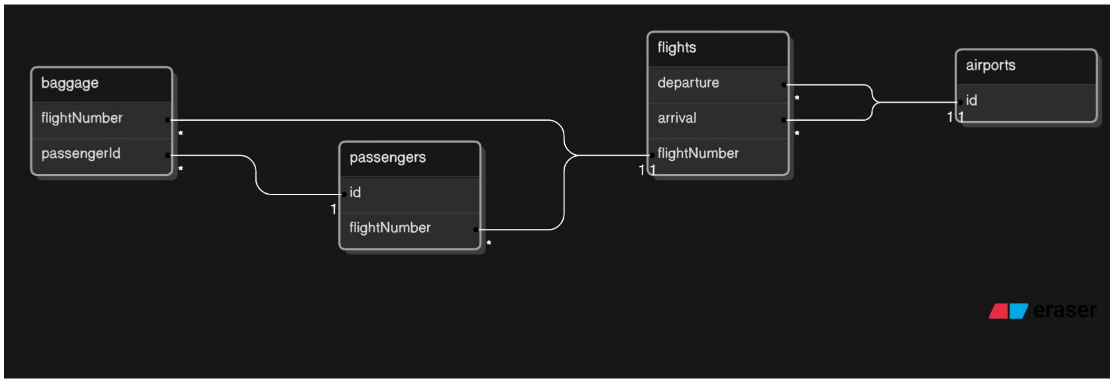


Figure No: 2.1 Schema Design

2.1.1 Flights Collection

The Flights Collection stores detailed information about individual flights. Each entry includes a unique identifier (_id), the flight number (e.g., "AA123"), and the name of the airline operating the flight (e.g., "American Airlines"). It features embedded documents for departure and arrival, which detail the IATA code of the respective airports (e.g., "JFK" for departure), scheduled departure and arrival times, terminal information, and gate numbers. The status field reflects the current state of the flight (e.g., "On Time" or "Delayed"). Additional fields include scheduledDeparture and scheduledArrival times, passengerCount indicating the number of passengers booked, baggageCount for the total count of bags, and lastUpdated to track when the flight information was last modified.

2.1.2 Passengers Collection

The Passengers Collection maintains records of passengers associated with flights. Each entry contains a unique passenger identifier (_id), the passenger's first and last name, and an embedded contact document that includes their email address (e.g., "john.doe@example.com") and phone number (e.g., "+1234567890"). The collection also tracks the flightNumber of the flight they are booked on, a boolean checkedIn value indicating whether they have checked in, and an array of embedded baggage documents. Each baggage entry includes a unique baggage ID (e.g., "BAG123"), the weight of the baggage, its status (e.g., "Checked In"), and its intended destination.

2.1.3 Airports Collection

The Airports Collection holds essential information about airports involved in flights. Each airport entry features a unique identifier (_id), the IATA airport code (e.g., "JFK") and its full name (e.g., "John F. Kennedy International Airport"). An embedded document provides geographic details such as the city (e.g., "New York") and country (e.g., "USA"). Additionally, the collection includes an array for terminals, each containing a unique terminal ID (e.g., "T5") and a list of available facilities (e.g., ["Wifi", "Lounges", "Restaurants"]).

2.1.4 Data Flow and Relationships

The data flow in the airline tracking system is structured such that Flights store information pertaining to individual flights, capturing both departure and arrival details through embedded documents. Passengers reference specific flights, maintaining personal details and tracking the baggage associated with them. The Airports Collection provides data about the locations relevant to flights, as each flight entry connects implicitly to both a departure and arrival airport. Finally, Baggage entries are tied directly to individual passengers, linking baggage items to their respective passenger ID and flight number, ensuring that baggage logistics are effectively managed throughout the flight process.

2.2 KEYS

2.2.1 Primary Key

In your FAISS index, each vector represents a segment of a document, and each segment is stored with a unique identifier (such as document_id). This unique key allows the system to trace back to the specific document or segment when performing similarity searches.

2.2.2 Foreign Key

When searching for document segments related to a query, the system retrieves these segments based on their vector similarities. The foreign key relationships could be used in a database (if needed) to reference the document associated with a particular vector segment.

CHAPTER 3

SYSTEM DESIGN

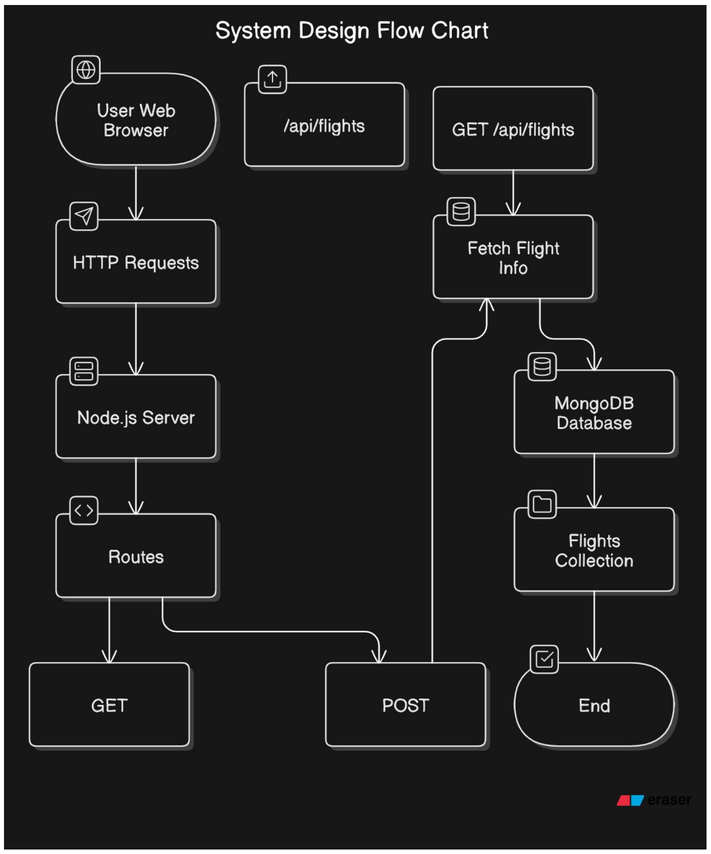


Figure No: 3.1 System Design

3.1 CAP THEOREM

MongoDB can play a crucial role in online airline tracking systems, particularly in the context of the CAP theorem, which states that a distributed data store can only guarantee two of the following three characteristics: Consistency, Availability, and Partition Tolerance. In such a system, real-time data from various sources (like flight status updates, weather data, and passenger information) can be stored in a MongoDB database, which is designed to handle large volumes of unstructured and semi-structured data.

Since airline tracking relies heavily on real-time updates and must be highly available to users across different regions, many implementations of MongoDB in this scenario may prioritize Availability and Partition Tolerance over strict Consistency. For example, when a flight's status changes, updates can be distributed across multiple nodes in the database, allowing travelers to access the latest information even if some nodes are temporarily unreachable. However, this could lead to eventual consistency, where a slight delay occurs before all users see the most recent updates.

3.2 CONSISTENCY

In the CAP theorem, Consistency means that all nodes in the distributed system reflect the most recent write. For our flight tracking system, this implies that when a flight's status is updated (for example, a flight delay or gate change), all queries across different collections (Flights, Passengers, Airports, Baggage) must return the latest data. However, ensuring strong consistency can be challenging in a distributed MongoDB environment. This is particularly true when updates occur concurrently, such as when multiple passengers check in simultaneously or when baggage tracking information changes frequently.

3.2.1 Eventual Consistency

In an online airline tracking system utilizing MongoDB, various consistency models can be employed to balance the need for real-time data access with the system's performance demands. The most prominent type of consistency in MongoDB is eventual consistency, where updates to data eventually propagate through the system, allowing users to obtain the latest information albeit with some possible delay. This is particularly useful in scenarios like tracking flight statuses, where instant updates across multiple database replicas may not be feasible at all times due to network latency.

3.2.2 Strong Consistency

Moreover, MongoDB supports strong consistency through its write concern configuration, which ensures that data is acknowledged only once it has been written to the specified number of replicas, thereby providing up-to-date information for critical operations, such as user check-ins. For applications requiring more robust integrity during transactional operations, MongoDB also offers multi-document transactions, enabling developers to enforce stronger consistency guarantees across multiple related documents, which is essential for maintaining coherent state in operations that depend on multiple data points, such as connecting flights.

3.3 COMPARATIVE ANALYSIS

The primary purposes of an online airline tracking system encompass several essential functions that significantly enhance the travel experience and operational efficiency within the aviation industry. First and foremost, these systems provide real-time flight monitoring, offering passengers and airlines up-to-date information on flight statuses, including departure and arrival times, gate assignments, and delays. This real-time tracking reduces uncertainty for travelers by allowing them to monitor the progress of their flights, ultimately leading to an improved travel experience.

Moreover, online airline tracking systems contribute to improved customer service by enabling passengers to conveniently access flight information without needing to make phone calls or visit the airport. This level of accessibility allows airlines to proactively communicate with travelers about flight changes or disruptions, which not only enhances customer satisfaction but also alleviates some of the burden on customer support teams. From an operational standpoint, these tracking systems play a crucial role in helping airlines and airports manage their operations more effectively. By providing real-time data on flight statuses and gate utilization, airlines can make informed decisions regarding flight scheduling, resource allocation, and contingency planning.

While online airline tracking systems provide valuable real-time information, they are not without their disadvantages. One of the primary concerns is the potential for inaccuracies or delays in the data presented. If the system experiences connectivity issues or is not updated in real time, passengers may receive erroneous information about flight statuses, leading to confusion and unnecessary stress.

In our project minimizing delays in an online airline tracking system offers multiple advantages that significantly enhance both customer experience and operational efficiency. First and foremost, real-time updates allow passengers to receive immediate information regarding flight status, gate changes, delays, and cancellations, empowering them to make informed decisions about check-in, boarding, and connecting flights. This timely information not only reduces stress but also contributes to a smoother travel experience. Additionally, airlines benefit from operational efficiency as access to near real-time data enables better resource allocation and decision-making, which can lead to cost savings.

Furthermore, providing reliable and up-to-date information fosters customer trust and loyalty; passengers are more likely to choose the same airline for future travels when they know they can depend on accurate communication. In cases of disruptions, a system that minimizes lags allows airlines to quickly communicate with affected passengers and offer alternatives, enhancing overall satisfaction. This immediacy also supports integration with other systems—such as booking platforms and airport information systems—facilitating coordinated responses to delays.

3.3 FLOW CHART

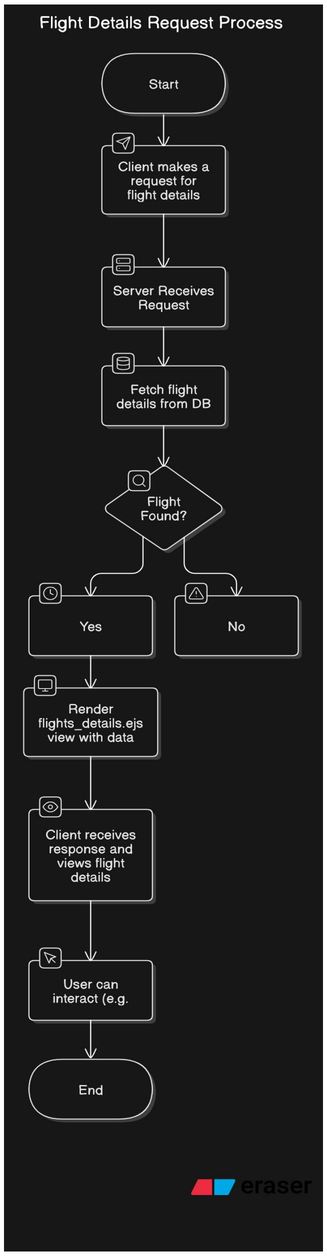

Flight Details Request Process
Start
Client makes a request for flight details
Server Receives Request
Fetch flight details from DB
Flight Found?
Yes
No
Render flights_details.ejs view with data
Client receives response and views flight details
User can interact (e.g.
End
eraser

Figure No: 3. 3 Flow chart

The flow of the flight details application begins when the client initiates a request to view specific flight information by supplying the airline and flight number as query parameters. Upon receiving this request, the server processes it by querying the database to retrieve the relevant flight details. If the request successfully locates the flight information, the server calculates any delays by comparing the actual arrival time with the expected arrival time. This calculation determines whether the flight was on time or delayed, quantified in minutes.

Once the delay is assessed, the server renders the flights_details.ejs view, injecting it with the pertinent flight data, including the airline, flight number, departure and arrival times, flight status, and any relevant delay information. In the case where no flight details are found, the server returns an appropriate error message to the client, indicating the failure to locate the requested information.

The client then receives and displays the formatted flight details to the user on the web page. The user has the option to interact further with the application, such as clicking a back button to return to the previous screen. This interaction maintains a smooth user experience, and the application sits idle, awaiting further requests, ready to process more client queries efficiently.

Thus, the flow diagram encapsulates the core processes and decision points, highlighting the sequential nature of actions taken within the application as it responds to user interactions and database queries.

CHAPTER 4

PROPOSED DESIGN

4.1 DESIGN DIAGRAM

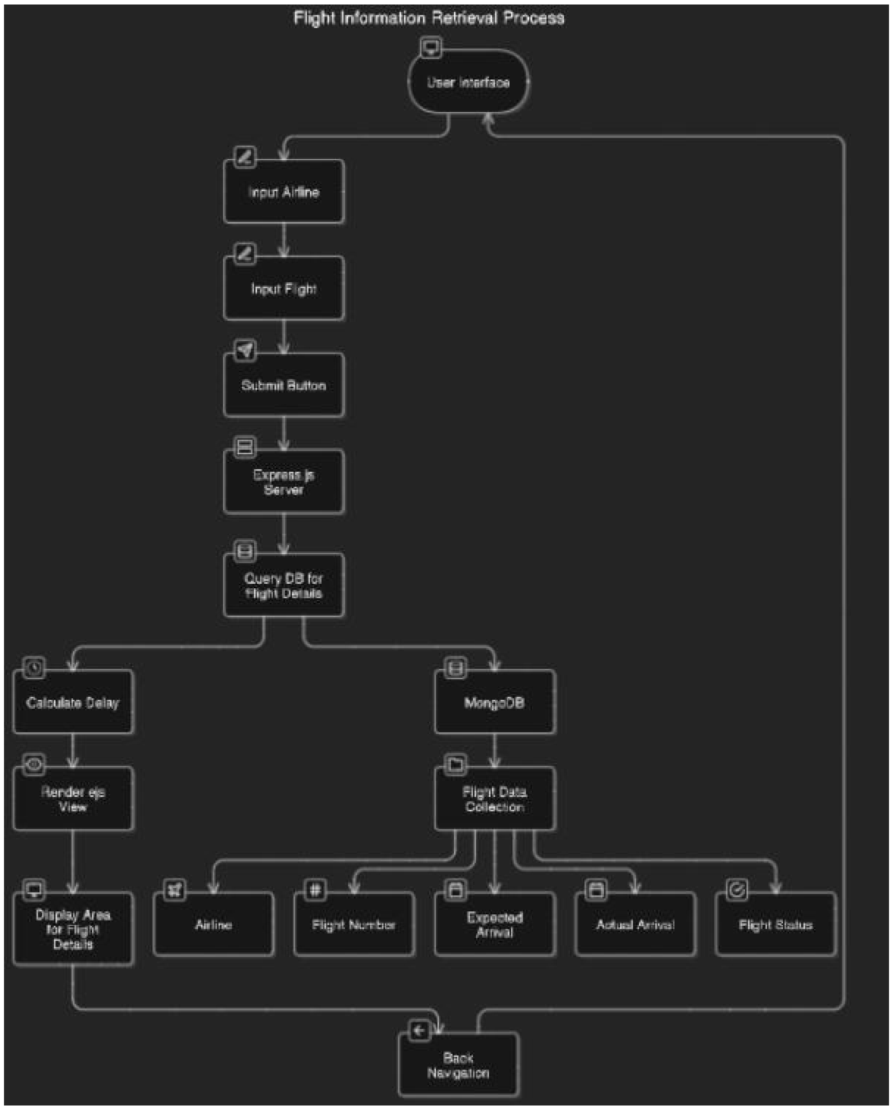


Figure No: 4.1 Flight details design diagram

The design of the flight tracking application is structured to ensure a seamless interaction between the user, the server, and the database. At the top layer, the User Interface serves as the entry point for users. It includes input fields where users can enter both the airline and flight number, necessary for querying the flight information. A prominent submit button facilitates the submission of these queries, while a designated display area presents the retrieved flight details or any error messages, thus enhancing user experience. Additionally, a back navigation option allows users to return to previous screens effortlessly.

Descending into the middle layer, the Express.js Server acts as the backbone of the application. It handles incoming requests from the user interface, specifically targeting an endpoint dedicated to flight details. Upon receiving a request, the server executes the logic needed to interface with the MongoDB database, where all flight data is stored. This database includes critical information such as the airline, flight number, expected arrival time, actual arrival time, and current flight status. The server queries this collection to locate the relevant flight information based on the user's input.

Once the server retrieves the necessary data, it proceeds to calculate any delays by comparing the actual arrival time to the expected arrival time. This information is essential for informing users about the status of their flight, as delays are quantified in minutes. After processing the request, the server returns a rendered view, using the EJS templating engine, which displays the flight details directly to the user interface.

If the flight information cannot be located, the server responds with an appropriate error message, ensuring that the user is informed of the issue. The entire architecture is designed to be intuitive and efficient, with a clear flow of information from the user through the server and into the database, and back to the user. This design not only improves the application's functionality but also guarantees a smooth and responsive user experience, positioning the flight tracking application as a reliable tool for users seeking timely and accurate flight information.

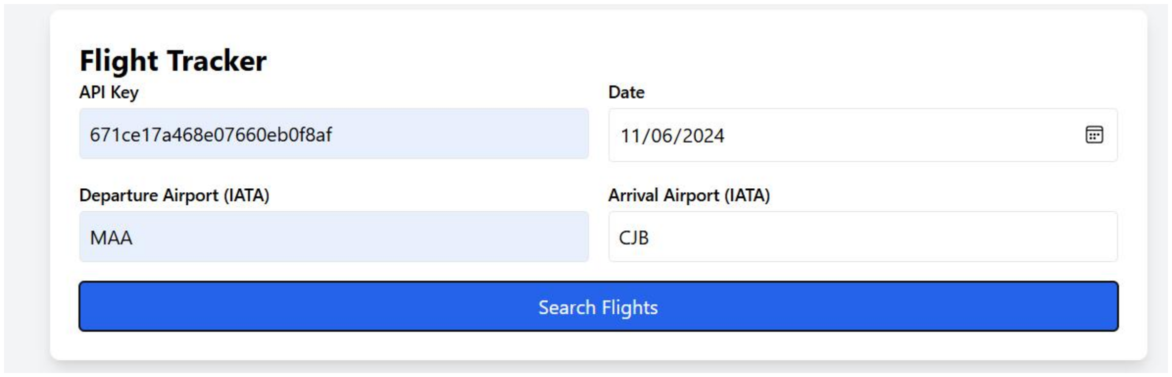
4.2 CODE

4.2.1 Server.js

```
const express = require('express');

const path = require('path');

const { MongoClient } = require('mongodb');

const bodyParser = require('body-parser');

const cors = require('cors');



const app = express();

const port = 3000;

const uri = 'mongodb://127.0.0.1:27017'; // Use the correct MongoDB URI

let client;

app.use(bodyParser.json());

app.use(cors());

app.set('view engine', 'ejs');

app.set('views', path.join(__dirname, 'views'));

app.use(express.static(path.join(__dirname, 'public')));

async function connectDB() {
    try {
        client = new MongoClient(uri);
        await client.connect();
        console.log("Connected to MongoDB");
    } catch (error) {
        console.error('MongoDB connection error:', error);
        process.exit(1); // terminate the app if the DB connection fails
    }
}
app.post('/api/flights', async (req, res) => {
    const flightDetails = req.body;
    try {
        const database = client.db('flight_tracker');
        const collection = database.collection('flights');
        const result = await collection.insertOne(flightDetails);
        res.status(201).json(result);
    } catch (error) {
        console.error(error);
        res.status(500).json({ error: 'Failed to save flight details' });
    }
});
app.get('/api/flights', async (req, res) => {
    const { airline, flightNumber } = req.query;
    try {
        const database = client.db('flight_tracker');
        const collection = database.collection('flights');
        const flightDetails = await collection.findOne({ Airline: airline, FlightNumber:
flightNumber });
        if (!flightDetails) {
            return res.status(404).json({ error: 'Flight not found' });
        }
        res.status(200).json(flightDetails);
    } catch (error) {
        console.error(error);
        res.status(500).json({ error: 'Failed to retrieve flight details' });
    }
});
```

```
app.get('/', (req, res) => {

    res.render('index');

});

app.get('/flightDetails', (req, res) => {

    res.render('flights_details');

});

connectDB().then(() => {

    app.listen(port, () => {

        console.log(`Server is running on http://localhost:${port}`);

    });

});
```

4.3 RESULT


Flight Tracker

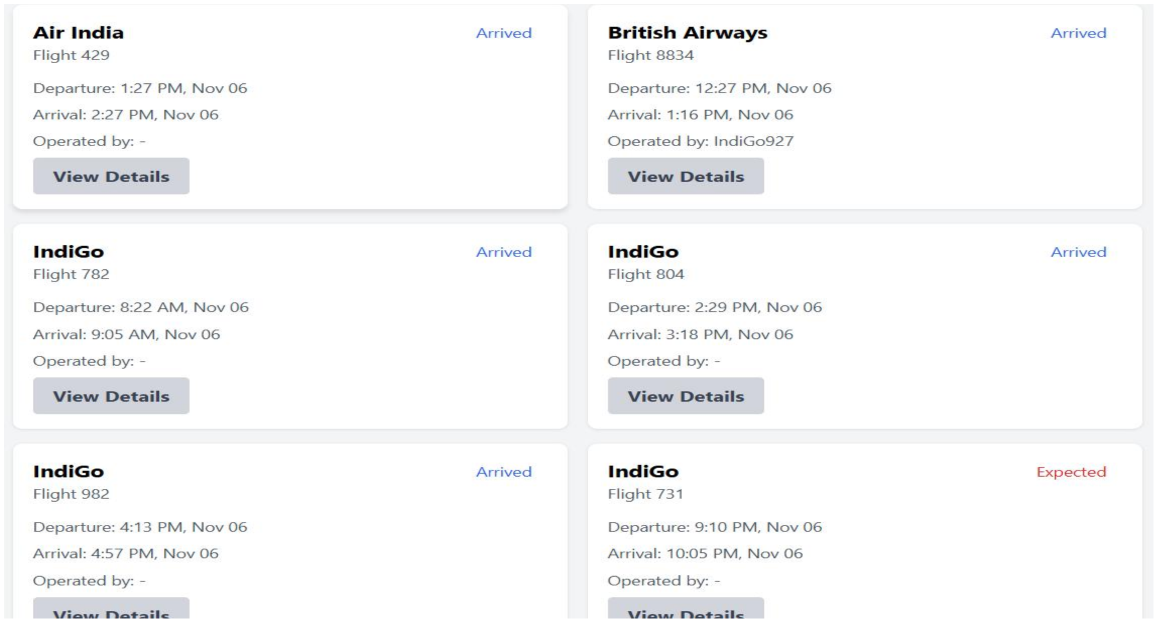
API Key: 671ce17a468e07660eb0f8af

Date: 11/06/2024

Departure Airport (IATA): MAA

Arrival Airport (IATA): CJB

Figure No: 4.3.1 Search Flights


Air India Arrived
Flight 429
Departure: 1:27 PM, Nov 06
Arrival: 2:27 PM, Nov 06
Operated by: -

British Airways Arrived
Flight 8834
Departure: 12:27 PM, Nov 06
Arrival: 1:16 PM, Nov 06
Operated by: IndiGo927

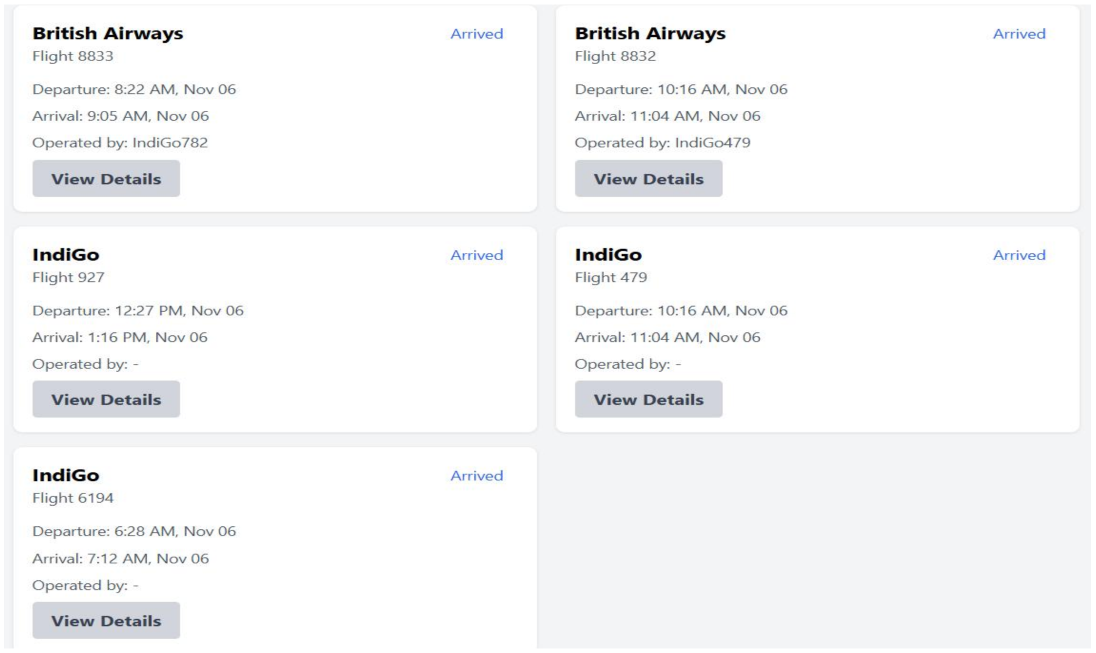
IndiGo Arrived
Flight 782
Departure: 8:22 AM, Nov 06
Arrival: 9:05 AM, Nov 06
Operated by: -

IndiGo Arrived
Flight 804
Departure: 2:29 PM, Nov 06
Arrival: 3:18 PM, Nov 06
Operated by: -

IndiGo Arrived
Flight 982
Departure: 4:13 PM, Nov 06
Arrival: 4:57 PM, Nov 06
Operated by: -

IndiGo Expected
Flight 731
Departure: 9:10 PM, Nov 06
Arrival: 10:05 PM, Nov 06
Operated by: -

Figure No: 4.3.2 Flight Details



British Airways
Flight 8833
Departure: 8:22 AM, Nov 06
Arrival: 9:05 AM, Nov 06
Operated by: IndiGo782

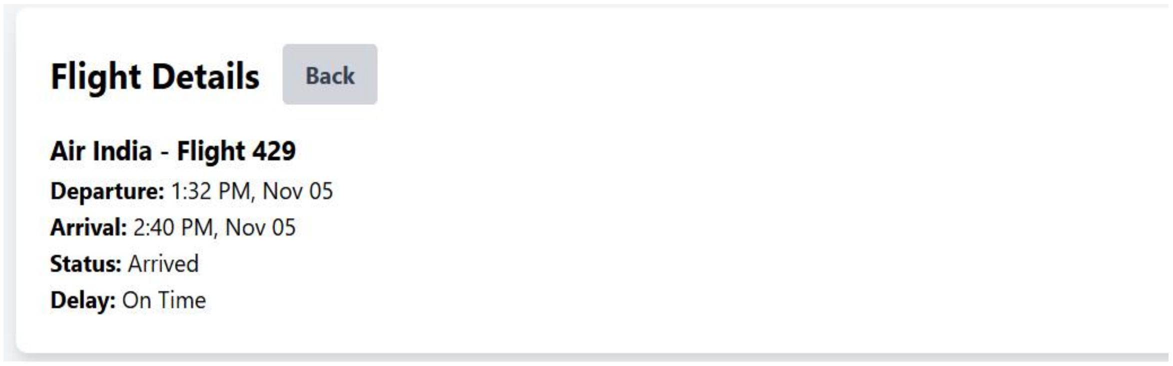
British Airways
Flight 8832
Departure: 10:16 AM, Nov 06
Arrival: 11:04 AM, Nov 06
Operated by: IndiGo479

IndiGo
Flight 927
Departure: 12:27 PM, Nov 06
Arrival: 1:16 PM, Nov 06
Operated by: -

IndiGo
Flight 479
Departure: 10:16 AM, Nov 06
Arrival: 11:04 AM, Nov 06
Operated by: -

IndiGo
Flight 6194
Departure: 6:28 AM, Nov 06
Arrival: 7:12 AM, Nov 06
Operated by: -

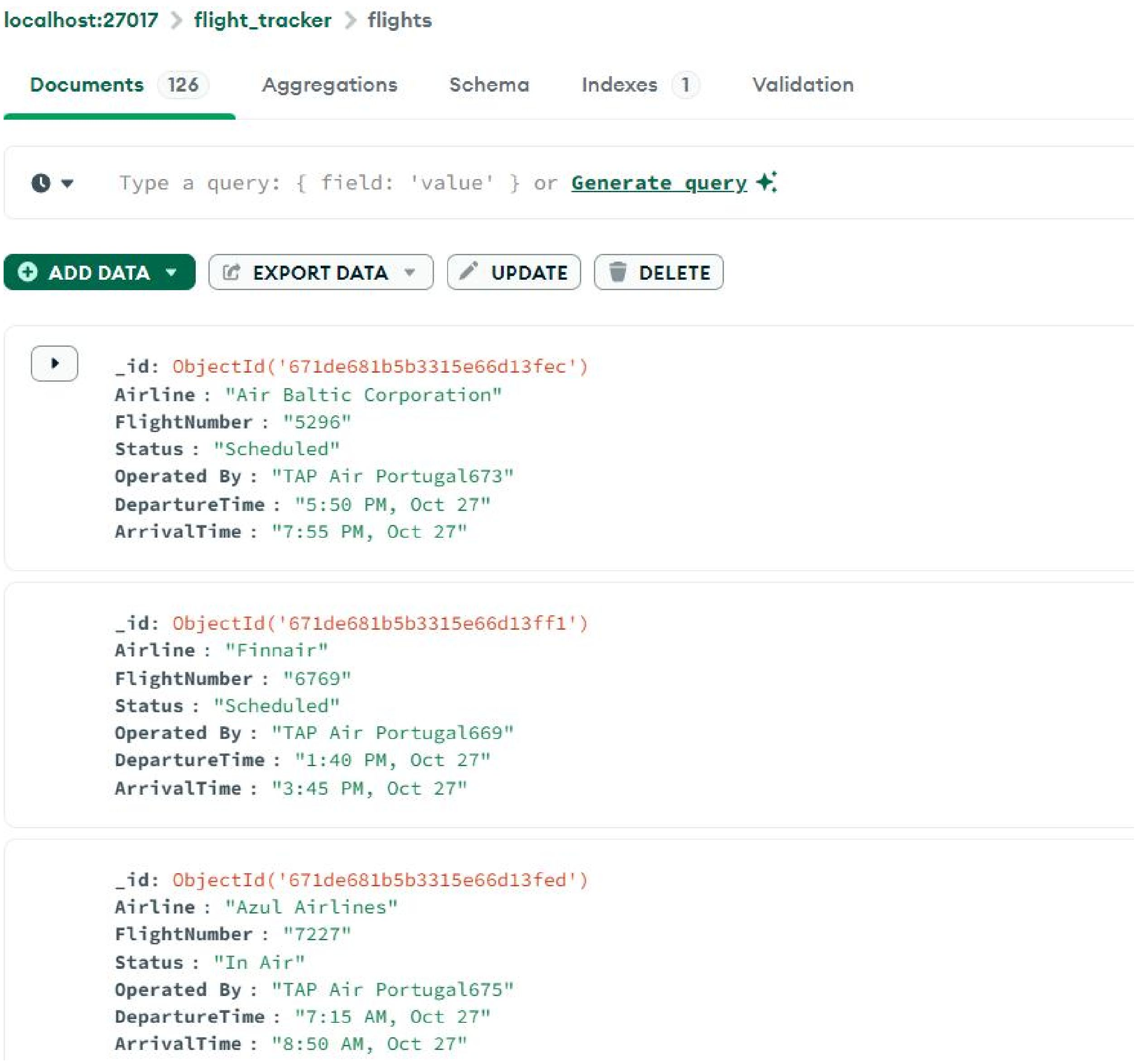
Figure No: 4.3.3 Flight Details



Flight Details

Air India - Flight 429
Departure: 1:32 PM, Nov 05
Arrival: 2:40 PM, Nov 05
Status: Arrived
Delay: On Time

Figure No: 4.3.4 Specific Flight


localhost:27017 > flight_tracker > flights
Documents 126
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
ADD DATA
EXPORT DATA
UPDATE
DELETE

```
_id: ObjectId('671de681b5b3315e66d13fec')
Airline : "Air Baltic Corporation"
FlightNumber : "5296"
Status : "Scheduled"
Operated By : "TAP Air Portugal673"
DepartureTime : "5:50 PM, Oct 27"
ArrivalTime : "7:55 PM, Oct 27"

_id: ObjectId('671de681b5b3315e66d13ff1')
Airline : "Finnair"
FlightNumber : "6769"
Status : "Scheduled"
Operated By : "TAP Air Portugal669"
DepartureTime : "1:40 PM, Oct 27"
ArrivalTime : "3:45 PM, Oct 27"

_id: ObjectId('671de681b5b3315e66d13fed')
Airline : "Azul Airlines"
FlightNumber : "7227"
Status : "In Air"
Operated By : "TAP Air Portugal675"
DepartureTime : "7:15 AM, Oct 27"
ArrivalTime : "8:50 AM, Oct 27"
```

Fig 4.3.5 Mongodb

CHAPTER 5

CONCLUSION

In conclusion, the airline tracking application represents a significant advancement in how travelers access real-time flight information, leveraging modern web technologies to enhance user experience. By integrating Node.js and Express.js to handle server operations efficiently, coupled with the robust data management capabilities of MongoDB for querying flight information, the application delivers timely and accurate updates to users. The dynamic rendering of information using EJS ensures that users receive a seamless interface that is both intuitive and responsive. As air travel continues to evolve, the need for reliable information is paramount; this application effectively addresses that need, empowering passengers with essential insights that can transform their travel experience. Furthermore, the architecture and design lend themselves to scalability and future enhancements, making the airline tracking application not just a valuable tool for today but also a foundation for growth and innovation in the aviation sector. Overall, this project stands out as a testament to the power of web technologies in solving real-world challenges in air travel, facilitating smoother journeys and providing peace of mind to users on the go.

Moreover, this project highlights the practical applications of AI in addressing real-world problems related to document management and information retrieval. It also demonstrates how technologies like machine learning, vector search, and natural language processing can be integrated to create more intelligent and user-friendly systems. The design of this system is particularly valuable in fields where time-sensitive and precise access to information is critical, and it has the potential to transform workflows by reducing the time and effort required to locate and synthesize relevant data from vast repositories of documents.

REFERENCES

- https://www.amazon.com/Node-js-Design-Patterns-Designing-Scalable/dp/1839214118
- https://www.flightaware.com/commercial/firehose/documentation
- https://docs.python-requests.org/en/latest/
- https://github.com/search?q=flight+fetcher&type=repositories
- https://www.mongodb.com/docs/
- https://www.nationalacademies.org/trb/transportation-research-board
- Aviation Week-https://aviationweek.com/
- https://nodejs.org/docs/latest/api/
- https://www.tutorialspoint.com/nodejs/index.html
- https://www.youtube.com/results?search_query=node.js+tutorial
- https://www.w3schools.com/mongodb/
- https://www.freecodecamp.org/learn/back-end-development-and-apis/mongodb-and/