Architecture Design

Flight Fare Prediction

**Contents**

Document Version Control**……………………………………………………………………….**

Abstract**…………………………………………………………………………………………..**

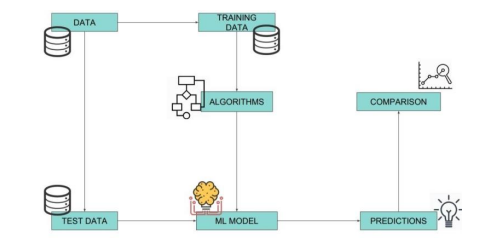
1. Introduction**………………………………………………………………………………….**
   * 1.1 Why this Architecture Design Document?**……………………………………...**
2. Architecture **…………………………………………………………………………………**
3. Architecture Design **………………………………………………………………………….**
   * 3.1 Data Collection **…………………………………………………………………..**
   * 3.2 Data Description **…………………………………………………………………**
   * 3.3 Importing Data into Database **……………………………………………………**
   * 3.4 Exporting Data from Database **……………………………………………………**
   * 3.5 Data Preprocessing **……………………………………………………………….**
   * 3.6 Modeling Process **…………………………………………………………………**
   * 3.7 UI Integration **……………………………………………………………………..**
   * 3.8 Data from User**……………………………………………………………………**
   * 3.9 Data Validation**…………………………………………………………………….**
   * 3.10 Rendering the Results**……………………………………………………………**
   * 3.11 Deployment**……………………………………………………………………...**

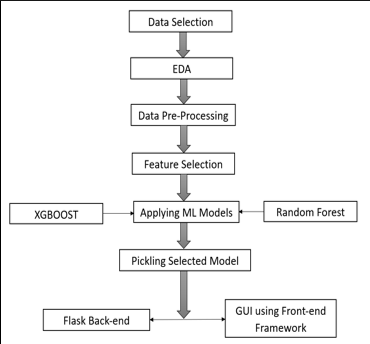
**Abstract**

The recent changes in the international market have had a significant impact on the aviation sector due to various reasons. This impact affects both business and customer perspectives. The major reason for such an impact is the implementation of different rules by governments for various airline companies worldwide. As a result, the prices of flight tickets have varied from one place to another. Booking a flight ticket involves different criteria that determine the price, such as online bookings and offline bookings. In this machine learning implementation, we will explore the factors that impact the price of flight tickets and predict the appropriate price.

1. **Understanding the System:** The document offers a comprehensive overview of the system architecture, making it easier for developers and stakeholders to grasp the overall design and how different components interact with each other.
2. **Coding Guidance:** By providing detailed module descriptions, the Architecture Design Document acts as a coding guide for programmers. It outlines the functionalities and responsibilities of each module, enabling developers to write code more efficiently and accurately.
3. **Consistency and Collaboration:** With a well-defined architecture, the development team can work cohesively towards a common goal. The document ensures consistency in coding practices and facilitates collaboration among team members.
4. **Scalability and Maintenance:** A clear architecture design allows for better scalability and easier maintenance. As the project evolves, developers can easily understand and modify different modules without disrupting the entire system.
5. **Future Enhancements:** The document lays the groundwork for future enhancements and updates. When new features need to be added or existing ones improved, the architecture design serves as a reference point for making informed decisions.
6. **Communication with Stakeholders:** The Architecture Design Document also serves as a communication tool between developers and stakeholders. It helps stakeholders understand the technical aspects of the project and assess its feasibility.
7. **Troubleshooting and Debugging:** When issues or bugs arise, having a well-documented architecture design aids in the troubleshooting process. Developers can refer back to the document to identify potential problem areas and find solutions more efficiently.

**2. Architecture**



****

**3. Architecture Design**

**3.1 Data Collection**

The data for this project is collected from the Kaggle dataset available at [Dataset Used](https://www.kaggle.com/datasets/nikhilmittal/flight-fare-prediction-mh)

**3.2 Data Description**

The Flight Fare Prediction dataset contains over 10,000 entries and is publicly available on Kaggle. The dataset is divided into two separate Excel files: train.xlsx and test.xlsx. It includes information such as the Date of Journey, Source, Destination, Arrival Time, Departure Time, Total Stops, Airlines, Additional Info, and Price.

**3.3 Importing Data into Database**

An API is created to import the data into a Cassandra database. The following steps are performed:

* Establish a connection with the database.
* Create a database named FlightInfo.
* Use the cqlsh command to create the necessary tables in the database.
* Finally, use a cqlsh command to bulk insert the dataset into the table.

**3.4 Exporting Data from Database**

The API also provides a download URL, allowing users to export the data from the database to a CSV file.

**3.5 Data Preprocessing**

* Verify the correct data types of the columns by checking the dataset's information.
* Check for null values that may affect the model's accuracy.
* Convert the desired columns into datetime format.
* Perform one-hot encoding on the desired columns.
* Check the distribution of the columns to interpret their importance.

After preprocessing the data, it is ready to be used for training a machine learning model.

**3.6 Modeling Process**

After preprocessing the data, visualize the insights gained from the data. Then, randomly split the data into training and testing datasets. Use the Random Forest Regressor algorithm to model the data and predict the flight fare price.

**3.7 UI Integration**

Create CSS and HTML files and integrate them with the machine learning model. Integrate all the required files into the app.py file and test the integration locally.

**3.8 Data from User**

Retrieve data from the user through the created HTML web page.

**3.9 Data Validation**

Process and validate the user-provided data in the app.py file. Send the validated data to the trained model for prediction.

**3.10 Rendering the Results**

Render the predicted flight fare results on the web page.

**3.11 Deployment**

Deploy the tested model to any Server as per your adjustments to make the project accessible from any internet-enabled device.

This document represents the Architecture Design for the Flight Fare Prediction project. It provides an overview of the project's architecture and design details, including data collection, data description, data preprocessing, modeling process, UI integration, data validation, rendering the results, and deployment.Top of Form