Low Level Design

Flight Fare Prediction

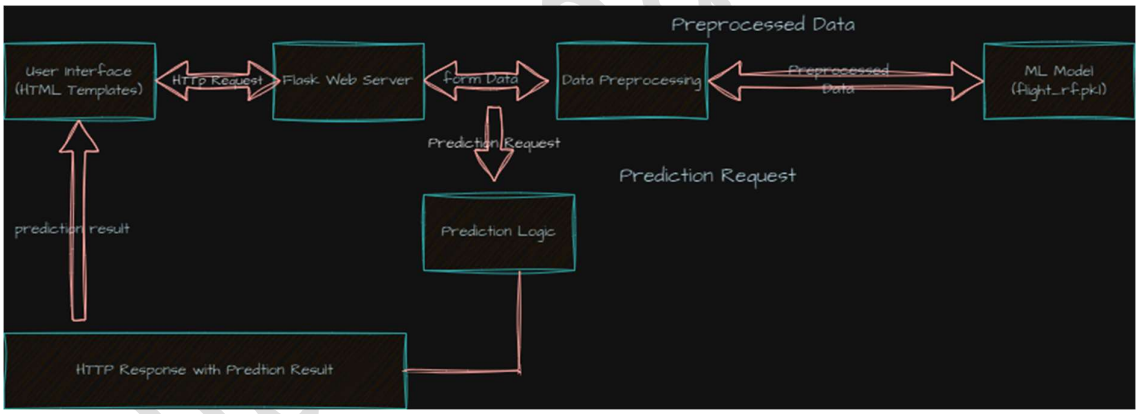
Written By	Vijay Kumar Shah
Document Version	0.1
Last Revised Date	03/09/2024

1. Introduction

This document provides a detailed low-level design for the Flight Price Prediction system. It covers the architecture, module breakdown, data flow, and interactions between different components. The system predicts flight prices based on various user inputs like journey details, airline, and stops.

2. Architecture Overview

- **Frontend:** HTML templates (home.html) for user input.
- **Backend:** Flask application handling requests and using a pre-trained Random Forest model to predict flight prices.
- **Data Processing:** Data extracted from user input is processed into a format suitable for the model.
- **Model:** A pre-trained Random Forest model serialized using pickle.



3. Module Breakdown

3.1. Flask Application

App Initialization (app.py):

- Flask app is initialized and the pre-trained model is loaded using pickle.
- **Modules**:
    - Flask: For creating the web application.
    - request: For handling incoming HTTP requests.
    - render_template: For rendering HTML templates.
    - cross_origin: For handling CORS (Cross-Origin Resource Sharing).
    - sklearn, pickle: For loading and using the machine learning model.
    - pandas: For processing date-time data.

Routes:

- /: Renders the home page (home.html).
- /predict: Handles the prediction logic based on user input.

3.2. User Input Handling

- **Date_of_Journey**:
    - Extracts and processes the journey date and time.
    - Converts the date into Journey_day and Journey_month.
- **Departure Time**:
    - Extracts and processes the departure time.
    - Converts it into Dep_hour and Dep_min.
- **Arrival Time**:
    - Extracts and processes the arrival time.
    - Converts it into Arrival_hour and Arrival_min.

- **Duration Calculation**:
    - Calculates the flight duration by subtracting departure time from arrival time.
    - Converts duration into hours (dur_hour) and minutes (dur_min).

3.3. Categorical Variables Handling

- **Airline Selection**:
    - Identifies the airline from user input.
    - One-hot encoding is applied to represent each airline in a binary format (e.g., Jet Airways = 1, IndiGo = 0).
- **Source and Destination**:
    - Identifies the source and destination locations from user input.
    - One-hot encoding is applied similarly to the airline handling.
- **Total Stops**:
    - Extracts the number of stops from user input and converts it into a numerical format.

3.4. Prediction

- **Model Input**:
    - Combines all processed input variables into a list that matches the feature order expected by the model.
- **Model Prediction**:
    - The model predicts the flight price using the processed inputs.
- **Result Rendering**:
    - The predicted price is returned to the frontend and displayed to the user.

4. Data Flow

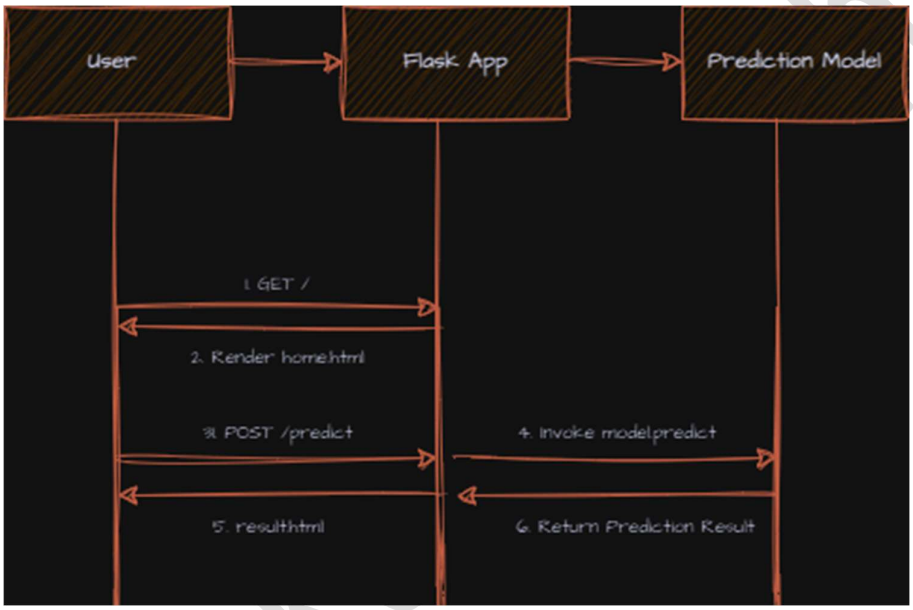
1. **User Input**: The user enters journey details (date, time, airline, stops, etc.) on the home page.
2. **Data Processing**: The backend processes the input data and converts it into the required format.
3. **Prediction**: The processed data is fed into the machine learning model for prediction.
4. **Result**: The predicted price is returned to the frontend and displayed to the user.

5. Detailed Code Explanation

- **File: app.py**
    - **Imports**: Necessary libraries and modules are imported.
    - **Model Loading**: The trained model is loaded from the serialized pickle file.
    - **Routes**:
        - **Home Route (/)**: Renders the home page where users input flight details.
        - **Predict Route (/predict)**: Processes the input, runs the model prediction, and returns the result.
- **Template (home.html)**:
    - A form collects flight details from the user.
    - The form action directs the input to the /predict route for processing.

6. Sequence Diagrams

1. **User Input Flow**: User enters data → Form Submission → Data Processing in Flask → Model Prediction → Result Display.
2. **Backend Flow**: Flask receives request → Data Processing → Prediction → Return Response.



7. Error Handling

- **Form Validation**: Ensure all required fields are filled before submitting the form.
- **Type Checking**: Validate the input types (e.g., dates, numbers) before processing.
- **Model Input Validation**: Ensure that the input data is correctly formatted before passing it to the model.

8. Assumptions

- The input data format is consistent with the trained model's expectations.
- The model is pre-trained and loaded successfully during application startup.
- Users provide valid inputs.

9. Conclusion

This LLD document provides a detailed breakdown of the flight price prediction system, covering the architecture, modules, and data flow. The system is designed to be scalable, with a clear separation of concerns between the frontend and backend.