# Flight Delay Prediction App – Design Document

## 1. Project Overview

This project predicts whether a flight will be delayed based on flight details such as departure time, distance, and airline. The goal is to provide an interactive user interface (UI) for delay prediction using machine learning.

## 2. Goals & Use Cases

Primary Goals:

1. Use Bureau of Transportation Statistics (BTS) flight data to train a prediction model.
2. Implement a Random Forest classifier to predict delays.
3. Build a Streamlit-based web app to provide an easy-to-use prediction tool.

Use Cases:

• Travelers – Check if their flight might be delayed.

• Airlines – Analyze delay patterns to optimize scheduling.

• Data Scientists – Experiment with machine learning models for flight delay predictions.

## 3. Tech Stack

|  |  |
| --- | --- |
| **Component** | **Technology Used** |
| Programming Language | Python |
| Machine Learning | Scikit-Learn (Random Forest) |
| Data Processing | Pandas; SQLite3 |
| Web UI | Streamlit |
| Data Source | Bureau of Transportation Statistics On-Time Performance Data |
| Serialization | Pickle (saving/loading model) |

## 4. System Architecture

Data Flow:

1. User enters flight details in the Streamlit UI.
2. Input data is processed and converted into the same format as training data.
3. Random Forest model predicts whether the flight will be delayed.
4. Prediction result is displayed to the user.

**[BTS Dataset] → [Data Cleaning] → [Random Forest Model] → [Streamlit UI] → [Prediction Output]**

## 5. Feature Breakdown

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Train Model | Uses a Random Forest Classifier to learn from historical flight data (given January 2024 from BTS) |
| Web App UI | Allows users to input flight details and get a prediction |
| Data Preprocessing | Converts user input into a structured format with one-hot encoding |
| Feature Importance Analysis | Uses Random Forest feature importance to analyze which features contribute most to predictions |

## 6. Challenges & Future Improvements

Challenges Encountered:

• Mismatched feature names when loading the model in Streamlit.

• Handling missing values in the BTS dataset.

• Optimizing feature selection for better prediction accuracy.

Future Enhancements:

• Add live flight data integration using an API (e.g., FlightAware).

• Improve model accuracy by trying gradient boosting models (e.g., XGBoost).

• Deploy the app to Streamlit Cloud or use Flask/Django for a full-stack web app.

• Add a confidence score to predictions

• Add more fields for more accurate model responses (e.g. “Airline”, “Destination” etc.)