Architecture Design Document for Flight Price Prediction Web Application

1. Overview

- **Objective**: To build a web-based application that predicts flight prices based on user-provided flight details using a machine learning model.
- **Scope**: The architecture encompasses the user interface, backend services, database management, and machine learning model integration.

2. System Components

1. **Frontend (User Interface)**
   - **Technology**: HTML, CSS, Bootstrap, JavaScript (optional).
   - **Purpose**: To provide users with a web-based form to input flight details and display the predicted price.
   - **Components**:
     - **Input Form**: Fields for airline, source, destination, date, time, stops, etc.
     - **Display Area**: Section to show the predicted flight price.
2. **Backend (Flask Application)**
   - **Technology**: Python, Flask.
   - **Purpose**: To handle user requests, process data, interact with the database, and generate predictions using the ML model.
   - **Components**:
     - **Routing Layer**:
       - GET /: Serve the homepage with the form.
       - POST /predict: Handle form submissions, process data, and return the prediction.
     - **Data Processing Layer**:
       - Preprocess input data to match the ML model's requirements.
       - Store and retrieve data from MongoDB.
     - **Prediction Engine**:
       - Load the trained machine learning model.
       - Generate price predictions based on processed input data.
3. **Database (MongoDB)**
   - **Technology**: MongoDB.
   - **Purpose**: To store user inputs, processed data, and prediction results.
   - **Components**:
     - **Collections**:
       - flight_data: Stores user input data along with prediction results.
     - **Operations**:
       - Insert user data upon form submission.
       - Update data with prediction results.
       - Retrieve past predictions if needed.
4. **Machine Learning Model**
   - **Technology**: Scikit-learn (RandomForestRegressor).
   - **Purpose**: To predict flight prices based on input features.
   - **Components**:

        - **Model Storage**: Saved as flight_price_model.pkl.
        - **Prediction Logic**: Code to load the model, preprocess input, and generate predictions.

3. Data Flow

1. **User Interaction**: The user enters flight details via the frontend form.
2. **Request Handling**: The form data is sent to the Flask backend through a POST request.
3. **Data Processing**: The backend processes the input, stores it in MongoDB, and prepares it for the ML model.
4. **Prediction**: The preprocessed data is fed into the ML model to predict the flight price.
5. **Response**: The prediction result is stored in MongoDB and sent back to the frontend to be displayed to the user.

4. Infrastructure

- **Deployment**:
    - **Flask App**: Deployed on a cloud platform like Heroku or AWS.
    - **MongoDB**: Hosted on MongoDB Atlas or locally deployed.
    - **Model Hosting**: The trained model is stored on the server where the Flask app is deployed.
- **Scaling**:
    - **Horizontal Scaling**: For the backend to handle increased user load.
    - **Database Scaling**: Using MongoDB's sharding features to distribute data across multiple servers if necessary.

5. Security Considerations

- **Data Security**: Ensure all data in MongoDB is encrypted at rest.
- **Input Validation**: Sanitize and validate all user inputs to prevent XSS and SQL injection.
- **HTTPS**: Implement SSL/TLS for secure data transmission.

6. Error Handling

- **Backend**: Implement error handling mechanisms to catch and log any exceptions, especially during data processing and prediction phases.
- **Frontend**: Provide user-friendly error messages and validation checks.

7. Future Enhancements

- **Model Improvement**: Periodic retraining of the ML model with new data.
- **Advanced Features**: Adding user authentication, historical price trends, or multi-user support.