Low-Level Design Document for Flight Price Prediction Web Application

1. Database Schema (MongoDB)

- **Collection: flight_data**
  - **Document Structure:**
    - **_id: ObjectId (Auto-generated by MongoDB)**
    - **airline: String**
    - **source: String**
    - **destination: String**
    - **journey_day: Integer**
    - **journey_month: Integer**
    - **dep_hour: Integer**
    - **dep_minute: Integer**
    - **arrival_hour: Integer**
    - **arrival_minute: Integer**
    - **total_stops: Integer**
    - **price: Float (Optional, stored after prediction)**

2. API Endpoints

- **/ (GET):**
  - **Renders the home page with the input form.**
- **/predict (POST):**
  - **Receives form data, preprocesses it, stores it in MongoDB, and calls the model to predict the flight price.**
  - **Updates the MongoDB document with the predicted price.**
  - **Returns the prediction result to be displayed on the front-end.**

3. Backend Logic (Flask)

- **Form Handling:**
  - **Extract form data using request.form.**
  - **Convert and validate input data.**
- **Data Processing:**
  - **Preprocess data to match model requirements (e.g., date splitting, categorical encoding).**
  - **Store user input in MongoDB.**
- **Prediction:**
  - **Load the trained model (flight_price_model.pkl).**
  - **Apply preprocessing to the input data.**
  - **Predict the price and update the corresponding MongoDB document.**
- **Response Handling:**
  - **Render the HTML template with the prediction result.**

4. Frontend Components

- **HTML/CSS:**

    - **Input fields for flight details (airline, source, destination, etc.).**
    - **Display predicted price after submission.**
- **Templates:**
    - **index.html: Renders form and displays results using Jinja2.**

5. Error Handling

- **Form Validation:**
    - **Ensure all fields are filled out correctly.**
- **Database Operations:**
    - **Handle potential MongoDB connection errors.**
    - **Implement retry logic if the insertion or update fails.**
- **Prediction Failures:**
    - **Catch and log model prediction errors.**

6. Testing Strategy

- **Unit Tests:**
    - **Test individual components like data processing, database operations, and prediction functions.**
- **Integration Tests:**
    - **Ensure end-to-end flow from form submission to prediction and database storage works correctly.**
- **Database Tests:**
    - **Verify data insertion, update, and retrieval in MongoDB.**

7. Deployment

- **Environment Configuration:**
    - **Set up environment variables for MongoDB connection URI, Flask secret key, etc.**
- **Deployment Pipeline:**
    - **Deploy Flask app using a WSGI server (e.g., Gunicorn) on platforms like Heroku or AWS.**
    - **Ensure MongoDB connection is properly configured in the deployment environment.**

This LLD provides a clear, concise guide to the implementation details of My project, focusing on MongoDB integration, API design, and key backend processes.