🚚 Storyline: International Shipment Tracking with Dynamic Routing

Instructions :

- No use of AI or AI-related plugins/tools in the Code Editor.
- Time limit:
  - 1 hour 30 minutes to complete the task.
  - 10 minutes at the end are reserved for running test cases (these will be provided).
- No commits allowed in the GitHub repo after the task completion deadline
- Candidates may use browsers to search functions and snippets.
- Share Github Link after task completion

🎯 Objective:

The core problem you're solving is to build a robust backend system for a logistics company. This system needs to track shipments as they move internationally, but with a key feature: the ability to dynamically change the route during transit.

Imagine a package being sent from Ahmedabad to New York. Initially, you might assume a direct route. However, due to various logistics factors (e.g., a planned stop, a new carrier partnership, or a rerouting due to unforeseen circumstances), the package might need to go through intermediate locations. Your system must be able to handle this.

The system is broken down into three main functionalities:

- Shipment Management: Creating a shipment and its basic route.
- Dynamic Routing: The ability to insert new transit points (hops) into the existing route at any time.
- Flight Tracking & Reporting: Associating specific flights with each segment of the journey and providing APIs to track the shipment's progress and query flight data.

Assumptions:

- Shipment number and Flight Number are unique and not to be reused in another flight.
- The backend should be listening at port 3000.

---

1. Create Shipment (Points:3)

A customer creates a new shipment from Ahmedabad to New York with unique shipment_number

- **Endpoint:** POST /shipments/create
- **Request Body:**

```
{
  "origin": "Ahmedabad", (required)
  "destination": "New York",(required)
  “shipment_number”: ‘12345’ (required)
}
```

- **Success Response (201 Created):**

```
{
  "success": true,
  "message": "Shipment created successfully.",
  "data": {
    "shipment_number": "12345",
    "hops": [
      "Ahmedabad",
      "New York"
    ]
  }
}
```

- **Error Response**

```
{
  "success": false,
  "message": "Origin and destination are required fields."
}
```

System Action: The system creates a new `Shipment` record with a unique `shipment_number` (e.g., `abc123`). The `hops` array is initialized with the origin and destination.

State of Shipment `abc123`:

- `hops`: `["Ahmedabad", "New York"]`

2. Add Hop to an Existing Route (Points:3)

Later, the logistics team decides the shipment will pass through Dubai. They need to update the route to reflect this.

- **Endpoint:** POST /shipments/:shipment_number/hops/add
- **Request Body:**

```
{
  "previous_hop": "Ahmedabad",(required)
  "next_hop": "New York",(required)
  "new_hop": "Dubai"(required)
}
```

- **Success Response (200 OK):**

```
{
  "success": true,
  "message": "Hop added successfully.",
  "data": {
    "shipment_number": "12345",
    "hops": [
      "Ahmedabad",
      "Dubai",
      "New York"
    ]
  }
}
```

- **Error Response :**

```
{
  "success": false,
  "message": "Shipment with ID not found."
}
```

System Action: The system finds `"Ahmedabad"` and `"New York"` in the `hops` array and inserts `"Dubai"` between them.

State of Shipment `12345`:

- `hops`: `["Ahmedabad", "Dubai", "New York"]`

3. Add Flight Information (Points:5)

Now that the route is defined, the logistics team assigns specific flights to each leg of the journey. There are two legs: `Ahmedabad` to `Dubai`, and `Dubai` to `New York`.

- **Endpoint:** POST /shipments/:id/flights/add
- **Request Body**

```
{
  "carrier": "Emirates",(required)
  "from": "Ahmedabad",(required)
  "to": "Dubai",(required)
  "flight_number": "em-789"(required)
  "departure": "1754043211", // TimeStamp. (OR DATE)
  "arrival": "1754007211"
}
```

- **Success Response (201 Created):**

```
{
  "success": true,
  "message": "Flight information added successfully.",
  "data": {
    "shipment_number": "12345",
    "flight_number": "em-789",
    "flight_path": "Ahmedabad - Emirates - Dubai",
    "departure": "1754043211",
    "arrival": "1754007211",
    “status”: ‘in-transit’ // default status.
  }
}
```

- **Error Response :**

```
{
  "success": false,
  "message": "Unable to add a flight. The 'from' and 'to' locations are not consecutive
hops for this shipment."
}
```

System Action: A new `Flight` record is created (e.g., `em-789`) and linked to shipment `12345`.

---

4. Update Flight Status (Points:3)

As the shipment progresses, the system updates the status of each flight.The statuses are ‘`in-transit`’ and ‘`landed`

- **Initial Status:** All flights are `in-transit`.

- **After first leg completes:**
    - **API Call:** `PUT /flights/em-789/status`
    - **Payload:** `{"status": "landed"}`
- **System Action:** The status of `flight_number` `em-789` is updated.

- **Endpoint:** PUT /flights/:flight_number/status
- **Request Body:**

```
{
  "status": "landed"
}
```

- **Success Response (200 OK):**

```
{
  "success": true,
  "message": "Flight status updated successfully.",
  "data": {
    "flight_number": "em-789",
    "status": "landed"
  }
}
```

- **Error Response :**

```
{
  "success": false,
  "message": "Flight with ID 'em-789' not found."
}
```

---

5. Track Shipment Progress (Points:10)

A customer wants to see the progress of their shipment.

- **API Call:** `GET /track/shipment/12345`
- **System Action:** The system checks the shipment's route (`Ahmedabad` -> `Dubai` -> `New York`, which has 2 flights in between). It seems that one flight (`Ahmedabad` -> `Dubai`) has been completed because its associated flight (`em-789`) has a status of `landed`.
- **Calculation:**
    - Total flights: 2
    - Completed flights 1
    - Progress: (1/2) * 100 = 50%

- **Endpoint:** GET /track/shipment/:`shipment_number`
- **Success Response (200 OK):**

```
{
  "success": true,
  "message": "Shipment tracking details retrieved.",
  "data": {
    "shipment_number": "abc123",
    "current_location": "Dubai",
    "progress_percentage": 50,
    "status": "In Transit" // for 100%, Keep it as Completed
  }
}
```

- **Error Response (404 Not Found):**

```
{
  "success": false,
  "message": "Shipment with ID 12345 not found."
}
```

---

6. Get Shipment Hops (Points: 3)

Retrieves the ordered list of all transit points (hops) for a specific shipment.

- **Endpoint:** GET /shipments/:shipment_number/hops
- **Success Response (200 OK):**

```
{
  "success": true,
  "message": "Hops retrieved successfully.",
  "data": {
    "shipment_number": "12345",
    "hops": [
      "Ahmedabad",
      "Dubai",
      "New York"
    ]
  }
}
```

- **Error Response :**

```
{
  "success": false,
  "message": "Shipment with ID 12345 not found."
}
```

---

7. Query Flights by Carrier and Date Range (Points: 5)

System Action: The system searches the `Flight` records and returns all flights that match the carrier and whose `departure` or `arrival` dates fall within the specified range.

- **Endpoint:** GET /flights/query
- **Query Parameters:**
  - carrier: The carrier's name (e.g., Emirates).
  - start_date: The start of the date range in epoch(e.g. `1754043211`).
  - end_date: The end of the date range in epoch(e.g. `1754043211`).
- **Success Response (200 OK):**

```
{
  "success": true,
  "message": "Flights retrieved successfully.",
  "data": [
    {
      "flight_number": "em-789",
      "carrier": "Emirates",
      "from": "Ahmedabad",
      "to": "Dubai",
      "departure": "1754043211",
      "arrival": "1754007211"
    },
    {
      "flight_number": "em-790",
      "carrier": "Emirates",
      "from": "Dubai",
      "to": "New York",
      "departure": "1754007631",
      "arrival": "1754009971"
    }
  ]
}
```

- **Error Response (400 Bad Request):**

```
{
  "success": false,
  "message": "Required query parameters 'carrier', 'start_date', and 'end_date' must be
provided."
```

```
    }
```