Summary of the Assignment

Objective:
Develop an **API for Smart Tourism**, which will handle country-related data from a **JSON file**. The API should support **CRUD operations** and store the data in a **MongoDB database** using **Mongoose**.

Data Description:
The JSON file contains information about different countries with the following fields:

- **Country** – Name of the country (e.g., "Greece").
- **Quality of Life** – Ranking indicator (lower value means better quality).
- **Adventure** – Ranking indicator (lower value means better adventure opportunities).
- **Heritage** – Ranking indicator (lower value means richer cultural heritage).
- **Cost of Living Index** – Higher values indicate more expensive countries.
- **Restaurant Price Index** – Higher values indicate more expensive dining costs.

Data sources:

1. **Cost of Living Index by Country 2024**
2. **Best Countries Rankings 2023**

Example JSON Entry:

```
{
  "Country": "Greece",
  "Quality of Life": 28,
  "Adventure": 3,
  "Heritage": 2,
  "Cost of Living Index": 52.0,
  "Restaurant Price Index": 51.3
}
```

Implementation Steps:
1. Store Initial Data in the Database

- Read the JSON file containing country data.
- Save this data to a **MongoDB database** using **Mongoose**.

2. API CRUD Operations
Create API endpoints for:
a) Read Data (GET requests):

- Retrieve all countries.
- Filter and sort countries by:
  - **Cost of Living Index** (cheapest or most expensive).
  - **Restaurant Price Index** (cheapest or most expensive).
  - **Quality of Life, Adventure, Heritage** (lower ranking is better).

b) Update Data (PUT requests):

- Modify a country's data (e.g., update cost of living or restaurant prices).

c) Add New Data (POST requests):

- Insert a new country (data can be fictional).

d) Delete Data (DELETE requests):

- Remove a country from the database.

Project Guidelines
1. Project Structure:

- Follow the **MVC architecture**.
- Include a **package.json** file for dependencies and project settings.

2. API Parameters:

- Requests will accept parameters such as:
  - criterion: Field to filter by (e.g., "Quality of Life", "Cost of Living Index").
  - type: Sorting type (lowest, highest).
  - limit: Number of records to return.

Example API Endpoints:

- **Retrieve all countries:**
- GET /api/v1/countries
- **Filter cheapest countries:**
- GET /api/v1/countries/filter?criterion=Cost of Living Index&type=lowest&limit=5

3. Filtering Explanation:

- **Quality of Life, Adventure, Heritage → Lower values are better.**
- **Cost of Living Index, Restaurant Price Index → Higher values mean higher costs.**

Bonus (Optional, +2 points)
Develop a **simple front-end interface** that allows users to view and filter API data.

Additional Requirements:

✅ **Use Express.js** for API development.
✅ No authentication required.
✅ **MongoDB + Mongoose** for database management.
✅ Use **environment variables** in the application.
✅ Ensure the project starts with npm start.
✅ Submit a **README file** if necessary.
✅ Include a **Postman collection** for testing API requests.
✅ Deliver a **document with API requests and responses**.

Submission Details:

- Submit a **.zip file** containing:
  - The **code**.
  - The **documentation** with API requests and responses.
  - The **Postman collection** (if applicable).
- **Deadline:** Upload the file to e-Class before the deadline.
- **Partial submissions** are accepted as long as they are functional.

Good Luck! 🚀