# **Smart Tourism – Documentation (Summary in English)**

#### Introduction

Smart Tourism is a web application that manages and displays country data based on criteria such as **Quality of Life, Adventure, Heritage, Cost of Living Index, and Restaurant Price Index**.

- The backend is built with Node.js/Express and uses a MongoDB database.
- The **frontend** is developed in **React** with **TailwindCSS**.

#### **Execution Instructions**

#### **Starting the Backend**

- 1. Open a terminal in the **Smart Tourism** folder and navigate to the **backend** directory:
- 2. cd backend
- 3. Install dependencies:
- 4. npm install
- 5. Create and configure the .env file (explained below).
- 6. Start the backend server:
- 7. npm start

The server will run at <a href="http://localhost:5000/">http://localhost:5000/</a>.

# **Starting the Frontend**

- 1. Open a new terminal in the **Smart Tourism** folder and navigate to the **frontend** directory:
- 2. cd frontend
- 3. Install dependencies:
- 4. yarn install

or

npm install

- 5. Start the frontend:
- 6. yarn start

or

npm start

The frontend will run at <a href="http://localhost:3000/">http://localhost:3000/</a>.

# **Environment Variables (.env - Backend)**

Create a .env file inside the **backend** folder (if it doesn't exist) and add the following:

 $MONGO\_URI = your\_mongodb\_connection\_string$ 

PORT=5000

Replace your\_mongodb\_connection\_string with the actual **MongoDB connection string**.

#### **Backend Description**

#### Folder Structure (Backend)

backend/

— config/
— db.js
— controllers/
— countryController.js
— data/
— Merged\_Country\_Data.json
— models/
— country.js
— routes/
— countryRoutes.js
— server.js
— .env

— package.json
— package-lock.jsor
— importData.js

## **File Descriptions**

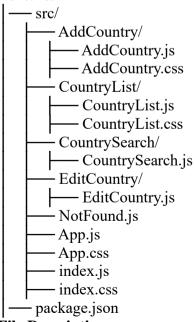
- $\mathbf{db.js} \rightarrow \text{Connects to MongoDB}$
- **countryController.js** → Contains API CRUD operations
- Merged\_Country\_Data.json → Stores country data with various indices
- **country.js** → Defines MongoDB schema
- **countryRoutes.js** → Manages API routes
- **server.js**  $\rightarrow$  Starts the Express server
- **importData.js** → Imports JSON data into MongoDB

API	<b>Endpoints</b>	(Backend)	)
$\Delta$ III	Linupoints	(Dackellu	,

Method	Endpoint	Description
GET	/api/v1/countries	Retrieves all countries
GET	/api/v1/countries/:id	Retrieves a country by ID
GET	/api/v1/countries/search/:name	Retrieves a country by name
POST	/api/v1/countries	Adds a new country
<b>PUT</b>	/api/v1/countries/:id	Updates a country
DELETE	/api/v1/countries/:id	Deletes a country
GET	/api/v1/countries/filter?criterion=Adventure&type=highest&limit=5	Filters and sorts countries based on a criterion

# Frontend Description

# **Folder Structure (Frontend)** frontend/



**File Descriptions** 

• CountryList.js → Displays and manages the country list, allowing filtering, sorting, and

modifications.

- **EditCountry.js** → Page for editing country details.
- AddCountry.js  $\rightarrow$  Page for adding a new country.
- CountrySearch.js  $\rightarrow$  Allows users to search for a country by name.
- **NotFound.js** → Displays a 404 error page for invalid routes.
- **App.js**  $\rightarrow$  The main component handling navigation between pages.
- **index.js**  $\rightarrow$  The entry point for the React application.

#### Using the API with Postman

To test the API in **Postman**:

- 1. Open Postman and **import** the Postman Collection file.
- 2. Use the following requests:

# **Example Requests**

- **GET** all countries:
- GET http://localhost:5000/api/v1/countries
- **GET** a country by name:
- GET http://localhost:5000/api/v1/countries/search/Greece
- **GET** a country by ID:
- GET http://localhost:5000/api/v1/countries/67adf338e29cb8dbdfbc9c89
- Filter countries (e.g., by Adventure rating, highest first, limit 5):
- GET

http://localhost:5000/api/v1/countries/filter?criterion=Adventure&type=highest&limit=5

- POST a new country:
- POST http://localhost:5000/api/v1/countries

```
Body (JSON):
 "Country": "FakeCountry",
 "Quality Of Life": 85.6,
 "Adventure": 78.2,
 "Heritage": 90.3,
 "Cost of Living Index": 70.1,
 "Restaurant_Price_Index": 65.4
      PUT (Update a country by ID):
      PUT http://localhost:5000/api/v1/countries/67adf338e29cb8dbdfbc9c89
Body (JSON):
 "Country": "Testing".
 "Ouality of Life": 5,
 "Adventure": 3,
 "Heritage": 2,
 "Cost_of_Living_Index": 1,
 "Restaurant Price Index": 0
```

- **DELETE** a country by ID:
- DELETE http://localhost:5000/api/v1/countries/67adf338e29cb8dbdfbc9c89

### Frontend UI Screenshots (Examples)

- Country List Page: Displays all countries.
- **Country Search Page:** Allows searching for a country by name.
- Country Filtering Page: Sorts countries based on selected criteria.
- Edit Country Page: Allows modification of country details.

• Add Country Page: Enables adding new countries.

# **Final Notes**

This documentation provides a detailed guide on setting up, running, and using the **Smart Tourism** web application. It covers backend and frontend setup, API functionality, and Postman API testing.