Travel AI Assistant Backend Assignment

1. Objective

Build a Travel AI Assistant backend using FastAPI that integrates with an AI API.
The system must suggest travel locations, packages, prices, apply budget filtering, validate input, and return structured responses.

---

2. Technology Stack (Mandatory)

- Python
- FastAPI
- Uvicorn
- python-dotenv
- Pydantic
- OpenAI API (or any LLM API)

---

3. How to Obtain AI API Key

Using OpenAI:
1. Visit https://platform.openai.com
2. Create an account or log in.
3. Navigate to API Keys section.
4. Generate a new secret key.
5. Store it inside .env file securely.

Important:
- Never hardcode API keys.
- Never upload .env to GitHub.

---

4. Project Structure (Expected)

travel_ai/
■
■■■ main.py
■■■ travel_data.py
■■■ schemas.py
■■■ .env
■■■ .gitignore
■■■ requirements.txt

---

5. Functional Requirements

A) FastAPI Setup
- Server must run successfully using uvicorn.
- Swagger documentation must be available at /docs.

B) Travel Package Data
- Minimum 4 hardcoded travel packages.
- Each must include destination name, type, price per person, and number of days.

C) Input Validation
- message (required, minimum length validation)
- budget (optional, positive integer)
- travelers (optional, positive integer)

D) GET /packages Endpoint
- Returns all travel packages.
- Supports optional budget query parameter.
- Filters packages where price_per_person <= budget.

E) Budget Filtering Logic
- Must be implemented as a reusable separate function.
- Must not be written directly inside route logic.

F) POST /travel-chat Endpoint
- Accept validated request body.
- Filter packages based on budget.
- Send filtered package data to AI API.
- AI suggests best suitable package.
- If travelers provided, calculate total price.
- Return structured JSON response.

---

6. Expected Flow

1. User sends travel request.
2. Input validation occurs.
3. Budget filtering logic applied.
4. Filtered data sent to AI API.
5. AI response processed.
6. Total cost calculated (if travelers provided).
7. Structured JSON returned.

---

7. Evaluation Criteria

- Clean project structure
- Proper API key security
- Correct AI integration
- Proper input validation
- Working filtering logic
- Clean and modular code
- Meaningful JSON responses

---

8. Submission Requirements

- Push project to GitHub.
- Do NOT include .env file.
- Include README.md with setup instructions.
- Provide example API request and response formats.
- Share repository link before deadline.
