Smart Event Planner - Backend Assignment

Problem Statement

Build a backend service that helps users plan outdoor events by integrating multiple APIs to provide intelligent recommendations. The system should consider weather forecasts, venue availability, traffic conditions, and user preferences to suggest optimal event timing and locations.

Think of scenarios like planning a cricket match, outdoor wedding, hiking trip, or corporate team outing where weather and location factors are critical for success.

Time Allocation: 2 Days

MUST HAVE Features (Core Requirements)

1. Single API Integration (Weather Focus)

- **Weather API**: Integrate with OpenWeatherMap for current weather and 5-day forecast
- Parse and store weather data in your internal format
- Handle API responses, errors gracefully

2. Event Management System

- Create events with basic details (name, location, date, event type)
- Store events with proper schema design
- Link events to weather data for the specified location and date

3. Simple Weather Analysis

- Fetch weather for event location and date
- Compare weather conditions against event type requirements
- Provide basic suitability score (Good/Okay/Poor)
- Show weather details relevant to outdoor events

4. Basic Recommendation Logic

- Suggest alternative dates within a date range if weather is poor
- Simple scoring based on temperature, precipitation, and wind
- Provide "better weather" alternatives for the same week
- Handle cases when no good weather days are available

Expected API Endpoints

```
# Event Management
POST   /events                    # Create event with location and date
GET    /events                    # List all events with weather status
PUT    /events/:id                # Update event details

# Weather Integration
GET    /weather/:location/:date   # Get weather for specific location and date
POST   /events/:id/weather-check  # Analyze weather for existing event
GET    /events/:id/alternatives   # Get better weather dates nearby (Optional)

# Simple Analytics
GET    /events/:id/suitability    # Weather suitability score for event
```

Core Technical Challenges

1. OpenWeatherMap API Integration

- **Authentication**: API key management and secure storage
- **Data Fetching**: Current weather and 5-day forecast endpoints
- **Response Parsing**: Extract relevant weather data (temp, precipitation, wind)
- **Error Handling**: API downtime, invalid locations, rate limit exceeded

2. Internal Data Transformation

Design your own data format for:

1. Events
2. Event_Weather_Analysis

3. Simple Caching Strategy (Optional)

- **Cache Duration**: Weather data valid for 3-6 hours
- **Cache Keys**: Location + date combination

- **Refresh Logic**: When to fetch new data vs serve cached data
- **Storage**: In-memory cache or simple database table

4. Weather Scoring Algorithm

Basic scoring based on event type:

Note: This is just an example to illustrate how a weather scoring algorithm could work. Feel free to come up with your own scoring system tailored to your specific event type or preferences.

For instance:

Outdoor Sports:
- Temperature 15-30°C: +30 points
- Precipitation <20%: +25 points
- Wind <20km/h: +20 points
- Clear/Partly cloudy: +25 points

Wedding/Formal Events:
- Temperature 18-28°C: +30 points
- Precipitation <10%: +30 points
- Wind <15km/h: +25 points
- Aesthetic weather: +15 points

OPTIONAL Features (Extra Credit)

1. Enhanced Weather Analysis

- **Historical Weather**: Compare with past week's weather for same day
- **Hourly Breakdown**: Detailed hour-by-hour weather for event duration
- **Weather Trends**: Improving/worsening forecast analysis
- **Multiple Locations**: Compare weather across nearby cities

2. Smart Notifications

- **Weather Change Alerts**: Email/SMS when forecast changes significantly
- **Event Reminders**: Day-before weather summary for upcoming events
- **Alternative Suggestions**: Automatic rescheduling recommendations
- **Threshold Alerts**: Custom weather threshold notifications

3. Simple Frontend Interface (Optional)

- **Event Creation Form**: Basic HTML forms for adding events
- **Weather Dashboard**: Display current weather and event recommendations
- **Event List**: Show all events with weather status indicators

---

Deliverables & Testing Instructions

CRITICAL: Real API Integration Required

Public Postman collection required

You MUST integrate with actual external APIs and demonstrate working weather-based recommendations. Include your secret key in the code itself.

Required External APIs:

- **Weather**: OpenWeatherMap or equivalent (free tier: 1000 calls/day)
- **Optional**: Simple location geocoding if needed (or use predefined city coordinates or hardcoded presets)

Test Scenarios in Postman Collection:

📁 Weather Event Planner APIs
 📁 Event Management
  ➢ Create Cricket Tournament (Mumbai, March 16)
  ➢ Create Wedding (Goa, December 10)
  ➢ Create Hiking Trip (Lonavala, October 20)
  ➢ List All Events
  ➢ Update Event Details

 📁 Weather Integration
  ➢ Get Weather for Mumbai (March 16)
  ➢ Check Weather Suitability for Cricket Event
  ➢ Get Alternative Dates for Poor Weather
  ➢ Weather Cache Status Check

 📁 Error Handling
  ➢ Invalid Location Test
  ➢ Weather API Downtime Handling

Expected Test Results:

- **Cricket Tournament**: Should get weather data, calculate suitability score, suggest alternatives if needed
- **Wedding Event**: Should handle different weather requirements than sports events

---

Key Technical Challenges

API Integration Issues:

- ❌ Not handling API rate limits efficiently
- ❌ Storing raw API responses without transformation
- ❌ Missing error handling for API failures

Business Logic Issues:

- ❌ Overly simplistic scoring algorithms
- ❌ Not considering event-specific weather requirements
- ❌ Poor handling of timezone conversions
- ❌ Missing edge cases (no suitable dates, extreme weather)

Data Management Issues:

- ❌ Inconsistent data formats across APIs
- ❌ Not handling partial data availability

Recommended Tech Stack

- **Backend**: Java (Spring Boot), Node.js, or Python
- **Database**: PostgreSQL (for relational data) or MongoDB or local json file
- **Caching**: Redis or in-memory caching
- **Deployment**: Railway.app, Render.com, or Heroku

API Resources:

- **OpenWeatherMap**: https://openweathermap.org/api (free tier available)
- **Google Places**: https://developers.google.com/places/web-service
- **Mapbox**: https://docs.mapbox.com/api/ (alternative to Google)

Feel free to explore solutions using Generative AI tools—they can help you get unstuck, brainstorm ideas, or speed things up. Best of luck!