LOBB_ASSIGNMENT__ANKIT_TAMBOLI

Description

LOBB_ASSIGNMENT is a mood-based music recommendation app that suggests songs based on your current mood and the weather in a given city. The app accepts a city and mood as input, fetches the current weather data for the city, and matches it to the provided mood. Based on this match, it then suggests a song track that aligns with the user's mood.

Tech Stack

- **FastAPI**: Web framework for building APIs

- **OpenWeatherMap**: Weather data for the given city

- **Last.fm**: Music recommendation based on mood

- **pytest**: Testing framework for unit and integration tests

- **Uvicorn**: ASGI server for running FastAPI

To run the app locally, follow the steps below:

1. Create a Virtual Environment
   python3 -m venv venv

2. Install Dependencies
   Install the required dependencies listed in `requirements.txt`:
   pip install -r requirements.txt

3. Configure API Keys
   Create a `.env` file and add the necessary API keys:
   OPENWEATHER_API_KEY=your_openweather_api_key
   LASTFM_API_KEY=your_lastfm_api_key

4. Run the App Locally
   Once dependencies are installed and environment variables are set, you can run the app with:
   uvicorn main:app --reload
   Your API will be available at `http://127.0.0.1:8000/`.

API Endpoints

POST /recommendation/

- **Description**: Accepts the user's mood and city, fetches the weather, and recommends a song based on the mood and weather match.

- **Request Body**:

```
{
  "mood": "happy",
  "city": "London"
}
```

- **Response**:

```
{
  "weather": "Clear",
  "mood_weather_match": true,
  "recommended_song": {
    "title": "Don't Stop Me Now",
    "artist": "Queen"
  }
}
```

GET /getmoods/

- **Description**: Returns a list of moods that we have used in mood-weather-matrix.

- **Response**:

```
{
  "moods": ["happy",  "sad", "calm", "energetic"]
}
```

Running Tests

To run the tests for this app, simply use command `pytest`

Error Handling

I have used try except blocks and raise below-mentioned errors

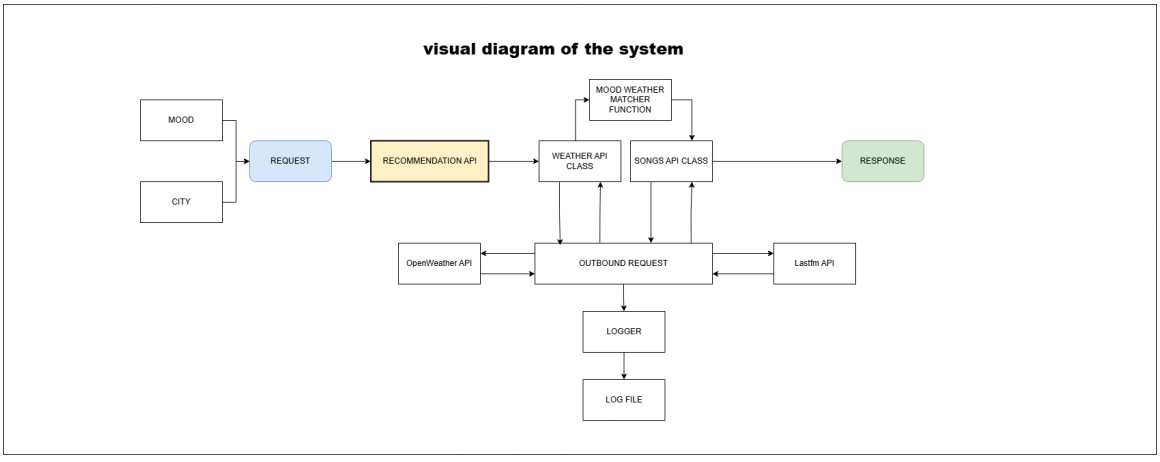
- Invalid City
- External service error.
- Mood must be provided.
- No tracks found.
- Failed after MAX_RETRIES attempts:
- No weather data found for city:
- Rate limit exceeded. Please try again after some time.

Design Decisions

- Mood-Weather Mapping : A matrix was created to map moods to weather conditions:

```
mood_weather_map = {
    "happy": ["Clear", "Clouds"],
    "sad": ["Rain", "Drizzle", "Thunderstorm"],
    "calm": ["Fog", "Mist"],
    "energetic": ["Clear", "Wind"],
}
```

Visual Diagram of System



Known Challenges

- **Handling 429 (Rate Limit) Errors in Tests** : I faced challenges when trying to test API rate-limiting errors (HTTP 429) with `pytest`. To resolve this, I had to refactor some parts of the exception handling. While it works, I acknowledge this could be improved with more time to follow best practices for error handling in tests.

Future Improvements

- More robust handling of edge cases in error management

- Further optimizations for handling API rate-limiting errors

- We can also add handlers to push logs to CloudWatch, Loki, Datadog, database etc.