**🎧 MoodTune**

**MoodTune** is an intelligent music recommendation app that matches your **mood** and the **current weather** in your city to suggest the perfect track.

It blends real-time weather data with mood classification logic, served via a FastAPI backend and optionally powered by AWS Lambda + API Gateway for mood matching logic.

Whether you're feeling nostalgic in the rain or hyped under clear skies — MoodTune curates a vibe for you.

**🌟 What Makes It Stand Out**

* ⚙️ **Smart mood-weather matching** logic using curated rules and fallback keyword recognition
* ☁️ **Live weather integration** from OpenWeatherMap
* 🎶 **Mood-based or title-based track recommendations** via Last.fm
* 🔄 **Dual-mode backend** — supports both local FastAPI logic and AWS Lambda + API Gateway integration
* 🧠 **Fallback support** when no exact match is found — always suggests a vibe
* 💻 **Beautiful frontend** with a blurred translucent glassmorphic design
* 🧪 **Well-structured test suite** with Pytest
* 📦 **YAML-based configuration** with secure key handling
* 🚀 Built for extensibility, readability, and clean architecture

**🧭 Project Plan & Architecture**

**🔹 Objective**

To build a weather-aware, mood-based song recommendation system using clean backend design, third-party APIs, and optional cloud integration — while keeping the UI engaging and simple.

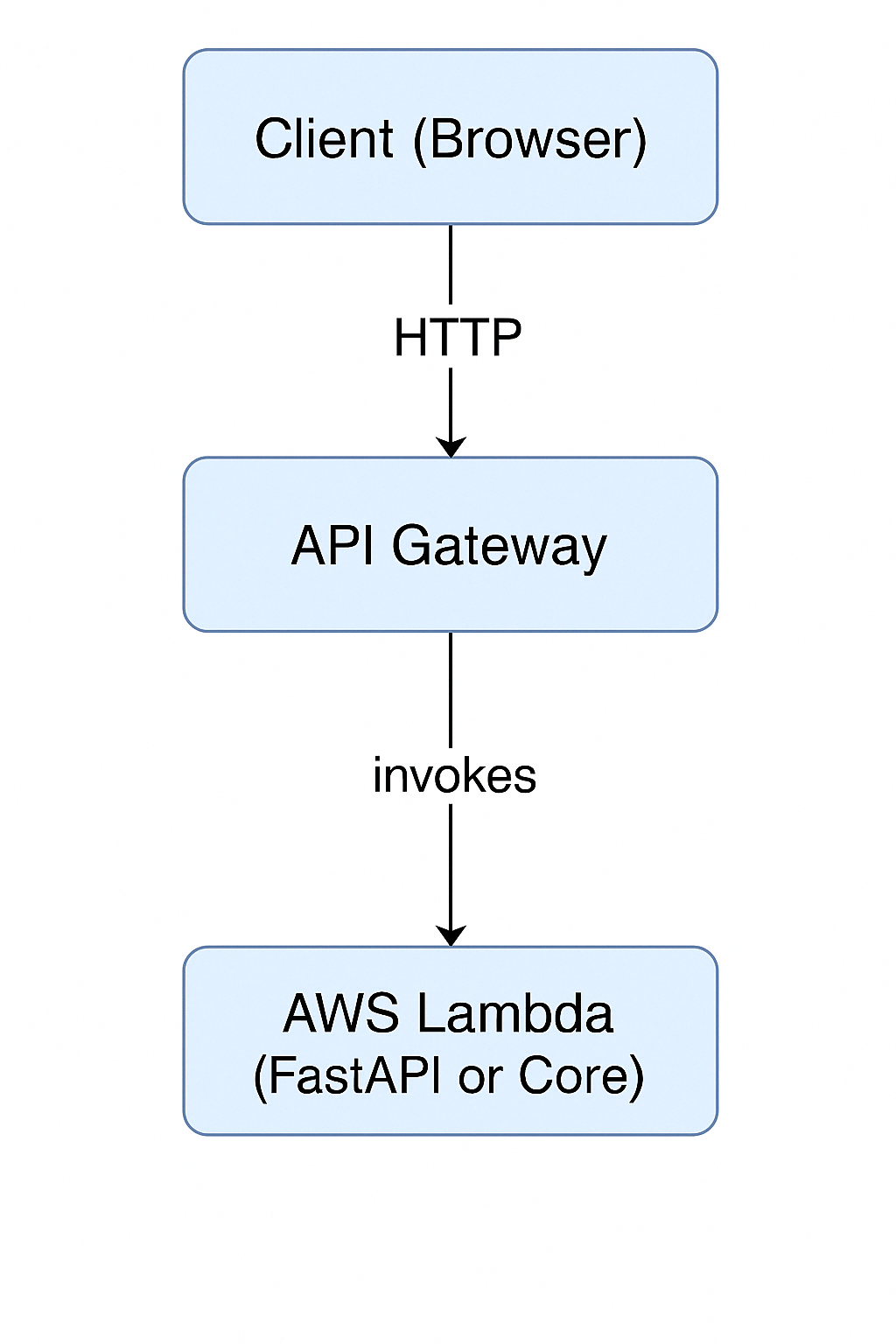
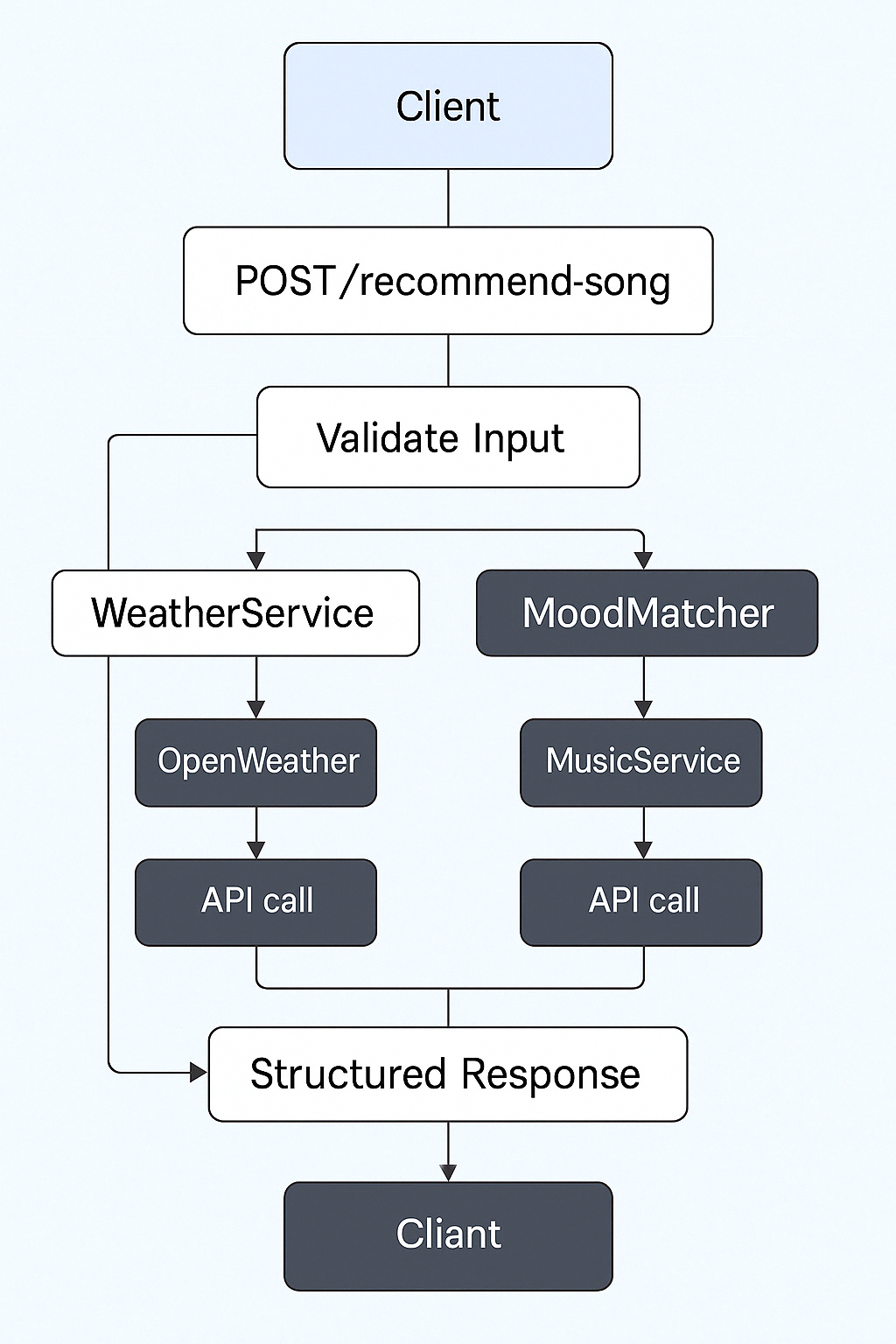
**🔹 Design Goals**

* ✅ Clean, modular backend with FastAPI
* ✅ Async architecture using httpx
* ✅ Separation of concerns (weather, music, mood logic as services)
* ✅ Environment-agnostic config via config.yaml
* ✅ Optional use of AWS Lambda + API Gateway to showcase cloud skills
* ✅ Full test coverage with pytest
* ✅ Visually engaging frontend for a polished user experience

**🔹 Technical Stack**

| **Layer** | **Technology / Service** |
| --- | --- |
| **Backend API** | FastAPI (async) |
| **Weather Data** | OpenWeatherMap API |
| **Music Data** | Last.fm API |
| **Cloud** | AWS Lambda + API Gateway (for mood matching logic) |
| **Frontend** | HTML, CSS (glassmorphic design), JS (JavaScript) |
| **Testing** | Pytest + pytest-asyncio |
| **Config** | YAML-based config loading |
| **Logging** | Python logging (structured in logger.py) |

**🔹 Architecture Diagram**



If explicitly we use AWS   
(lambda and + ApiGateway)

UI

**🔹 Project Flow**

1. **User inputs city & mood** from the frontend UI
2. **Weather data is fetched** from OpenWeatherMap
3. **Mood & weather are matched** using either local logic or AWS Lambda/ AWS Api-gateway
4. **Song is fetched** from Last.fm based on mood
5. **Result is displayed** in the UI with weather, mood alignment, and song recommendation

**🚀 Deployment Versions**

MoodTune is designed with flexibility in mind. It supports multiple deployment modes to fit different use cases and environments:

**1. 🖥️ Local FastAPI Server**

* **Description:** Runs locally using FastAPI’s built-in ASGI server.
* **Use Case:** Development and testing.
* **Command to run:**
* uvicorn main:app --reload

**2. 🧩 AWS Lambda (Standalone)**

* **Description:** The core *mood-matching logic* is deployed as an AWS Lambda function.
* **Invocation:** Triggered manually or via another AWS service (like Step Functions or another Lambda).
* **Use Case:** Modular, scalable backend logic.

**3. 🌐 AWS Lambda + API Gateway**

* **Description:** Full FastAPI application or Lambda wrapped as an API, exposed via API Gateway.
* **Use Case:** Production-grade, serverless REST API setup.
* **Features:**
  + Minimal server maintenance
  + Scales automatically
  + Can be consumed as a public API