Project-High Level Design on

Entertainment Planning Assistant Agent

Course Name: Agentic AI

Institution Name: Medicaps University
Course: Datagami Skill Based Course

Student Name(s) & Enrollment Number(s):

Sr no	Student Name	Enrolment Number
1	PRABAL SHUKLA	EN22CS301713
2	PRATIK KOUSHAL	EN22CS301747
3	PRIYANSH CHOUHAN	EN22CS301759
4	PUSHPRAJ SINGH CHOUHAN	EN22CS301774
5	RAJAT MALVIYA	EN22CS301783

Group : 11 D7

Project Number:AAI-34

Industry Mentor Name: Aashruti Shah

University Mentor Name:Ajeet Singh

Academic Year:2025-2026

Table of Contents

1. Introduction

The Entertainment Planning Assistant Agent is an intelligent backend system developed to simplify the process of discovering entertainment options such as movies, cultural events, and sports matches based on user interest and location. In today's digital age, users rely on multiple platforms to search for entertainment information. These platforms include ticket booking websites, movie review sites, event listing portals, and social media platforms. Since information is distributed across different sources, users often spend a significant amount of time searching, comparing, and analyzing options before making a decision.

The primary objective of this project is to build an AI-powered assistant that automates this process. Instead of manually browsing multiple websites, the user can simply enter their entertainment preference and city. The system will then collect relevant real-time information, analyze it intelligently, and provide a summarized recommendation.

This system is based on Agentic AI architecture. In Agentic AI, the system behaves like an autonomous agent that understands goals, plans actions, gathers necessary data, and generates meaningful output. The Planner Agent in this system interprets the user's request, fetches real-world information using APIs, and uses a Large Language Model (LLM) to reason and summarize the results.

The system prioritizes backend intelligence rather than complex front-end design. A lightweight optional graphical interface can be used, but the core focus remains on system architecture, API integration, and AI reasoning.

This project demonstrates practical implementation of modern AI techniques, including:

- Real-time web data integration
- LLM-based reasoning and summarization
- Modular system design
- Stateless backend architecture
- API-driven communication

The system is designed in such a way that it can be extended in the future to support additional features such as ticket booking integration, user profile-based personalization, recommendation history tracking, and mobile deployment.

1.1 Scope of the Document

This High-Level Design document provides a comprehensive overview of the system's architecture and major components. The purpose of this document is to explain how different parts of the system interact with each other and how data flows within the system.

This document covers:

- Overall system architecture
- Detailed explanation of modules
- Process flow and execution steps
- Information flow structure
- API endpoints and request-response structure

• Data handling mechanism
• Non-functional requirements
• Security and performance considerations

This document does not include:

• Source code implementation
• Detailed database schema (since no database is used)
• UI design mockups
• Deployment instructions

The focus of this document is architectural understanding rather than implementation details.

1.2 Intended Audience

This document is prepared for:

• Faculty members evaluating the project
• Academic mentors and supervisors
• Students studying AI systems
• Software engineering students
• Developers interested in agent-based architectures

The language is kept clear and structured so that readers from both technical and semi-technical backgrounds can understand the system.

1.3 System Overview

The Entertainment Planning Assistant Agent functions as a stateless backend service that processes user requests independently. The user provides two inputs:

• Entertainment preference (Movie/Event/Sport)
• City name

The backend receives the request via an API endpoint. The Planner Agent then analyzes the request and generates a structured search plan. The system fetches real-time data from the web using Groq API and processes it using a Groq-powered Large Language Model (via Groq-based Large Language Model (LLM)).

The final output includes:

• Summary of the entertainment option
• Available locations or theatres
• Timings
• Ratings and review highlights
• Recommendations

The system does not permanently store any user data. Each request is processed independently, making the system scalable and simple to manage.

2. System Design

The system is designed using a layered and modular architecture. Each component has a clearly defined responsibility. This design ensures better maintainability, scalability, and fault isolation.

2.1 Architectural Style

The system follows a layered architecture consisting of:

1. Presentation Layer
2. API Layer
3. Agent Layer
4. Integration Layer

Each layer performs specific tasks and communicates using structured interfaces.

2.2 Application Design

Presentation Layer

This layer is optional and implemented using Streamlit. It allows users to enter input in a simple form and display results in readable format.

API Layer

The API layer is implemented using FastAPI. It handles incoming HTTP requests, validates parameters, and forwards the request to the Planner Agent.

Agent Layer

The Planner Agent is the core intelligence unit of the system. It:

• Interprets user goal
• Plans search queries
• Coordinates data fetching
• Sends information to LLM
• Collects final summarized result

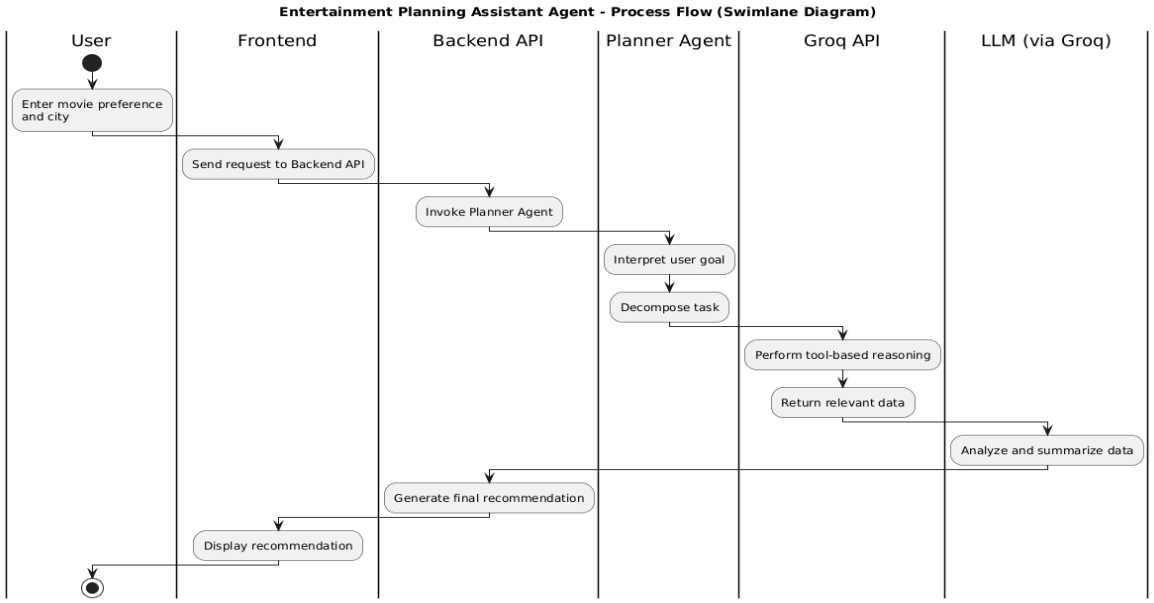
Integration Layer

This layer connects the system to:

• Groq API
• Groq-based Large Language Model (LLM) LLM runtime

2.3 Detailed Process Flow

Step 1: User enters movie/event and city
Step 2: FastAPI receives request
Step 3: Input validation performed
Step 4: Planner Agent generates search plan
Step 5: Groq API fetches relevant web data
Step 6: Raw data passed to LLM
Step 7: LLM analyzes ratings, reviews, timings
Step 8: Structured summary generated
Step 9: Response returned to user



2.4 Work Flow

User Input

↓

Backend API

↓

Planner Agent

↓

Web Search API

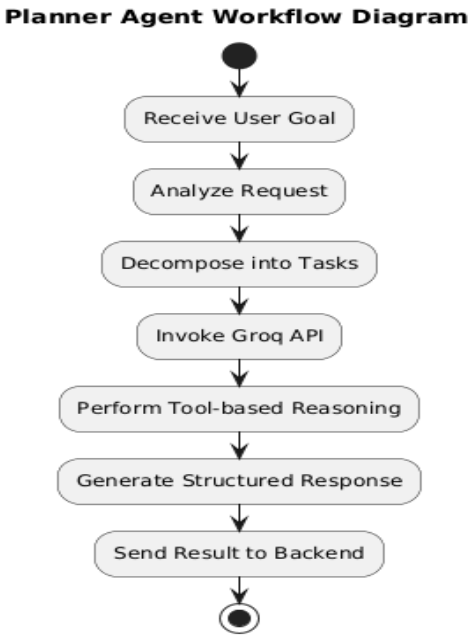
↓

LLM Processing

↓

Final Recommendation

The system ensures that raw data is never directly shown to the user.



2.5 Component Responsibilities

User Interface
• Collect user input
• Display results
FastAPI Backend
• Handle HTTP requests
• Validate input
• Route requests
Planner Agent
• Interpret goals
• Manage reasoning process
• Coordinate between modules
• Fetch real-time web information
LLM Module (Groq-based Large Language Model (LLM))
• Analyze textual data
• Summarize and generate recommendations

2.6 Error Handling Strategy

The system includes basic error handling mechanisms:
• Invalid input validation
• API failure detection
• Timeout management
• Graceful error messages
If external API fails, the system returns a structured error response instead of crashing.

3. Data Design

The system uses dynamic data processing instead of database storage.

3.1 Data Model

Input:
• String (Movie/Event name)
• String (City name)
Intermediate Data:
• Web search results
• Text reviews
• Event details
Output:
• Structured JSON response
• Summarized recommendation

3.2 Data Access Mechanism

The system retrieves real-time information using Groq API. The Planner Agent

coordinates with the Groq API to perform reasoning and generate structured responses.
No direct database queries are used.

3.3 Data Retention Policy

• No personal data stored
• No history saved
• Stateless request handling

This improves privacy and reduces security risks.

3.4 Data Security

• API keys stored securely
• Environment variable configuration
• No sensitive data logging

4. Interfaces

The system supports the following interfaces:

• REST API Interface
• Streamlit GUI
• Internal Agent-to-LLM Interface

Each interface is clearly defined and structured.

5. State and Session Management

The system is completely stateless. Every request is treated independently. No session IDs or cookies are maintained.
Benefits:

• Easier scaling
• Reduced complexity
• Improved reliability

6. Caching Strategy

Currently, caching is not implemented. However, future improvements may include:

• Frequently searched movie caching
• Short-term memory buffer
• API response caching

This would improve response time and reduce API calls.

7. Non-Functional Requirements

The system must satisfy the following non-functional requirements:
Reliability
The system must respond correctly even under API failure.
Scalability
The architecture supports horizontal scaling.

Maintainability
Modular code allows easy updates.
Usability
Simple and clear response format.

7.1 Security Aspects
• API keys protected
• No database storage
• Groq API ensures secure and efficient AI inference
• Minimal attack surface

7.2 Performance Considerations
• LLM processing time may vary
• Web API latency affects speed
• Local model ensures consistent reasoning
Optimization strategies:
• Response size limitation
• Efficient prompt engineering
• Future caching implementation

8. Future Enhancements
• Ticket booking integration
• User account personalization
• Mobile application version
• Multi-language support
• Recommendation ranking algorithm
• Integration with payment gateways

9. References
• FastAPI Documentation
• Groq API Documentation
• Groq-based Large Language Model (LLM) Documentation
• Agentic AI research literature