

Trip Planner: An MCP-Based Multi-Agent Service Orchestration Framework for Intelligent Travel Planning

Sarvesh Salunkhe, Yashraj Kadam, Ajay Jadhav, Nikhil Vhorkate, Mr. Pradeep Patil
Vidya Pratishthan's Kamalnayan Bajaj Institute of Engineering and Technology, Baramati

Abstract—Large Language Models (LLMs) have transformed the development of intelligent systems by enabling natural language reasoning, planning, and decision support. However, deploying LLM-powered applications in real-world environments remains challenging due to fragmented integrations, limited interoperability, and dependence on custom integration logic.

The Model Context Protocol (MCP) has emerged as a standardized framework for connecting AI systems with external tools, resources, and services through a unified communication mechanism.

This paper presents Trip Planner using MCP, an MCP-based multi-agent travel planning framework that integrates heterogeneous travel services including flights, hotels, transportation, tourist attractions, and budget optimization. Trip Planner using MCP introduces a Supervisor-Agent-based orchestration architecture that dynamically selects and coordinates specialized MCP servers according to user requirements.

The framework leverages FastAPI as an MCP host, a centralized MCP client for service communication, and Large Language Models for context-aware itinerary generation. A prototype implementation demonstrates the practical applicability of MCP in travel planning scenarios.

The proposed architecture improves interoperability, modularity, maintainability, and extensibility compared to traditional API-based integration approaches. Furthermore, the study highlights the role of MCP in enabling scalable multi-agent ecosystems and discusses security considerations associated with MCP deployments.

The results indicate that Trip Planner using MCP serves as a practical demonstration of MCP-enabled service orchestration and provides insights into the future development of intelligent agent-based applications.

Index Terms—Model Context Protocol, MCP, Large Language Models, Travel Planning, Agentic AI

I. INTRODUCTION

Recent advances in Large Language Models (LLMs) have significantly expanded the capabilities of artificial intelligence systems. Modern LLMs such as GPT, Gemini, Claude, and LLaMA demonstrate strong reasoning abilities and have enabled the development of autonomous software agents capable of interacting with users and external systems.

Despite these advances, practical deployment of LLM-based applications remains challenging. Most intelligent applications rely on custom integration logic, commonly referred to as glue code, to connect language models with databases, APIs, web services, and external resources. Such integrations are often difficult to maintain, scale, and secure.

To address these challenges, Anthropic introduced the Model Context Protocol (MCP), a standardized framework that facilitates communication between AI systems and external services. MCP provides a host-client-server architecture that enables structured access to tools, resources, and prompts through a unified protocol. By standardizing interactions, MCP reduces integration complexity and promotes interoperability across AI ecosystems.

The travel planning domain represents a compelling use case for MCP-enabled systems. Traditional travel planning often requires users to navigate multiple websites and applications to gather information regarding transportation, accommodation, tourist attractions, and budgeting. Existing AI-based travel planners improve recommendation quality but frequently rely on tightly coupled architectures and domain-specific integrations.

This paper presents Trip Planner using MCP, an MCP-based multi-agent travel planning framework designed to address these limitations. Trip Planner using MCP integrates multiple travel-related services through MCP servers and employs a Supervisor Agent to coordinate specialized agents responsible for transportation, accommodation, tourist attractions, budget optimization, and itinerary generation.

Unlike conventional travel planning applications, Trip Planner using MCP demonstrates how MCP can function as a universal interoperability layer capable of orchestrating heterogeneous services while preserving modularity and scalability.

A. Research Objectives

The objectives of this research are:

- 1) Design an MCP-based architecture for intelligent travel planning.
- 2) Develop a Supervisor-Agent-based orchestration framework.
- 3) Integrate multiple travel services using MCP.
- 4) Evaluate architectural advantages of MCP-based integration.
- 5) Demonstrate the feasibility of multi-agent travel planning systems.

B. Research Contributions

The major contributions of this work are summarized as follows:

- A novel MCP-based travel planning architecture.

- Dynamic service orchestration using a Supervisor Agent.
- Integration of multiple travel microservices through MCP.
- Context-aware itinerary generation using LLMs.
- Comprehensive analysis of scalability and security considerations.

II. BACKGROUND AND RELATED WORK

A. Model Context Protocol Fundamentals

The Model Context Protocol (MCP) is a standardized communication framework that enables structured interactions between AI applications and external resources. MCP follows a host-client-server architecture in which an MCP Host coordinates communication between language models and specialized MCP servers.

The protocol introduces three primary primitives:

- 1) **Tools:** Executable functions exposed by MCP servers.
- 2) **Resources:** Structured data accessible to AI systems.
- 3) **Prompts:** Reusable instructions that guide model behavior.

Unlike traditional API integrations, MCP provides a unified communication layer based on JSON-RPC, reducing integration complexity and improving interoperability among heterogeneous systems.

Figure 1 illustrates the general MCP workflow.

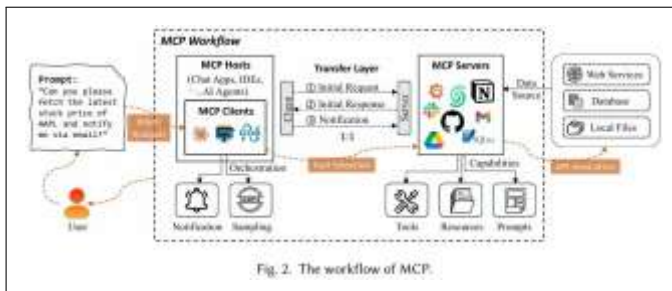


Fig. 1. General MCP Workflow

The workflow begins with a user request processed by an MCP Host. The host communicates with MCP servers through MCP clients, retrieves structured information, and provides contextual data to the language model for response generation.

B. Traditional Integration versus MCP-Based Integration

Before MCP, AI applications relied heavily on custom integrations with databases, APIs, and external services. While functional, these integrations often resulted in tightly coupled architectures that were difficult to maintain and extend.

MCP addresses these limitations by introducing a standardized intermediary layer that abstracts communication between AI applications and external systems.

Figure 2 compares traditional integration architectures with MCP-enabled architectures.

By standardizing tool access and context exchange, MCP improves modularity, maintainability, scalability, and service interoperability.

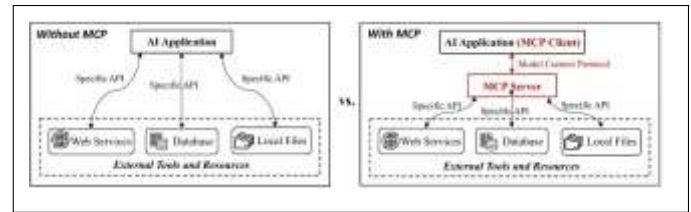


Fig. 2. Traditional Integration vs. MCP-Based Integration

Recent interoperability efforts such as Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP) focus on agent collaboration and communication. However, MCP is particularly suitable for service orchestration applications because of its standardized mechanism for tool invocation and external resource integration.

C. AI-Based Travel Planning Systems

The travel planning domain has increasingly adopted AI technologies to provide personalized recommendations and itinerary generation. Existing travel planning systems leverage Large Language Models (LLMs) to assist users in selecting destinations, transportation options, accommodations, and activities.

Despite these advances, most systems continue to rely on custom integrations and tightly coupled architectures. Common limitations include:

- Lack of interoperability standards
- Limited service extensibility
- Dependence on custom integration logic

These limitations motivate the need for MCP-based travel planning frameworks that support standardized communication and dynamic service integration.

D. Research Gap Analysis

Although substantial progress has been made in both MCP research and AI-based travel planning, limited work has explored the application of MCP within intelligent travel planning systems.

Table I summarizes the identified research gap.

The proposed Trip Planner using MCP framework addresses this gap by combining MCP-enabled interoperability, multi-agent orchestration, dynamic service selection, and intelligent travel planning within a unified architecture.

III. TRIP PLANNER USING MCP ARCHITECTURE

Trip Planner using MCP is an MCP-based multi-agent travel planning framework that integrates heterogeneous travel services through a standardized communication architecture. The framework combines MCP interoperability with intelligent orchestration to generate personalized travel itineraries while maintaining modularity, scalability, and service extensibility.

Unlike traditional travel planning systems that rely on tightly coupled APIs, Trip Planner using MCP exposes travel services through independent MCP servers. This architecture

TABLE I
RESEARCH GAP ANALYSIS

System	MCP Support	Multi-Agent	Dynamic Service Selection	Travel Planning
GPT Trip Planner	No	No	No	Yes
AI Trip Planner	No	No	Partial	Yes
Smithery.ai	Yes	Partial	No	No
MCP Ecosystem Studies	Yes	No	No	No
TripSync (Proposed)	Yes	Yes	Yes	Yes

enables dynamic service integration and simplifies system maintenance.

A. System Overview

The overall architecture of Trip Planner using MCP is shown in Figure 3.

The architecture consists of six major layers:

- 1) User Interface Layer
- 2) MCP Host Layer
- 3) Planning Layer
- 4) Orchestration Layer
- 5) MCP Communication Layer
- 6) Service Layer

The React-based frontend collects user travel requirements and forwards them to the FastAPI backend, which acts as the MCP Host. The host coordinates planning, orchestration, service invocation, and itinerary generation.

B. Planner Service

The Planner Service represents the first intelligent component within Trip Planner using MCP. It analyzes user requests, extracts travel parameters, identifies planning objectives, and generates structured context for downstream processing.

For example, the request:

“Plan a 4-day Goa trip under INR 20,000 with beach activities.”

is transformed into structured parameters including destination, duration, budget, and activity preferences. These parameters are forwarded to the Supervisor Agent for service orchestration.

C. Supervisor Agent

The Supervisor Agent is the primary contribution of TripSync. Instead of invoking all available services, it dynamically determines which MCP servers are required to satisfy a user request.

The agent performs task decomposition, service selection, resource allocation, and context propagation. By selectively activating only relevant services, the Supervisor Agent reduces computational overhead and improves scalability.

For example, a hotel recommendation query may activate only Hotel and Budget services while excluding transportation-related services.

D. MCP Communication Layer

The MCP Client acts as the communication gateway between the Supervisor Agent and MCP servers. It is responsible for session initialization, tool discovery, tool invocation, context transfer, and response aggregation.

Communication occurs through JSON-RPC over MCP-compatible transport mechanisms, providing a standardized interface for service integration.

E. Travel Service Agents

Trip Planner using MCP incorporates multiple specialized travel agents including Flight, Hotel, Train, Bus, Places, Budget, Price Analyzer, and Trip Optimizer agents.

Each agent is responsible for a specific domain task such as transportation discovery, accommodation selection, attraction recommendation, budget analysis, cost optimization, or itinerary refinement. The Supervisor Agent dynamically selects the appropriate agents according to user requirements and planning objectives.

F. Large Language Model Layer

The final stage of the architecture consists of an LLM-powered itinerary generation engine. The language model receives transportation options, accommodation details, tourist attractions, budget information, and user preferences collected from MCP servers.

Using this contextual information, the model generates a personalized travel itinerary including schedules, recommendations, transportation plans, and budget allocations.

IV. SYSTEM DESIGN AND IMPLEMENTATION

This section presents the implementation of Trip Planner using MCP, developed as a prototype to demonstrate MCP-based service orchestration for intelligent travel planning. The system consists of a React frontend, a FastAPI backend, an MCP communication layer, multiple travel agents, and a relational database.

A. Technology Stack

B. User Interface

The Trip Planner using MCPfrontend provides an interactive interface through which users specify travel requirements such as destination, budget, travel duration, and preferences. User requests are forwarded to the backend for planning and itinerary generation.

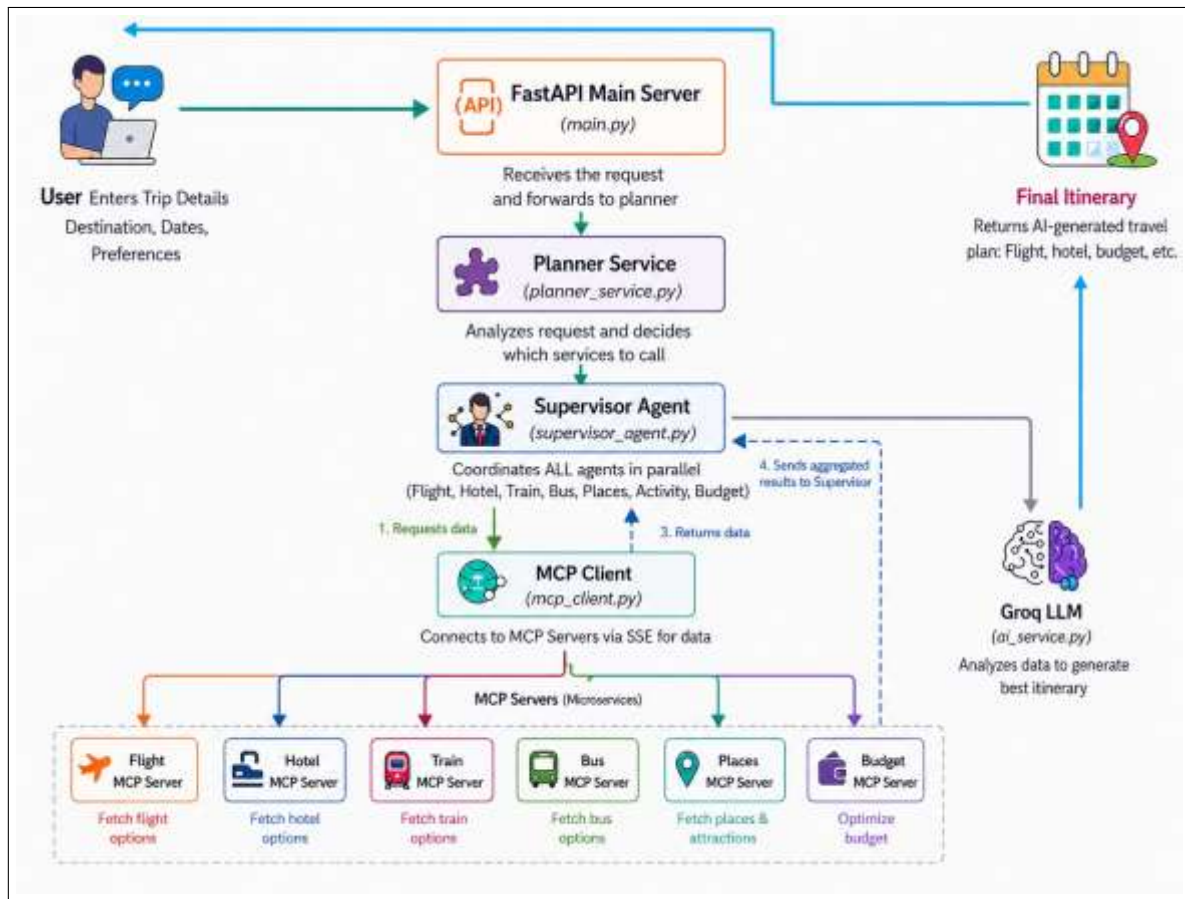


Fig. 3. Trip Planner using MCP System Architecture

TABLE II
TECHNOLOGY STACK

Component	Technology
Frontend	React, TypeScript
Backend	FastAPI
Protocol	MCP
Transport	SSE / HTTP
Database	SQLModel, SQLite
Authentication	JWT
AI Engine	Gemini / Groq
Language	Python

C. Backend Architecture

The backend is implemented using FastAPI and acts as the MCP Host. It manages user requests, coordinates MCP communication, invokes relevant travel services, aggregates responses, and forwards contextual information to the language model for itinerary generation.

The backend serves as the central orchestration component connecting the frontend, travel agents, MCP servers, and database layer.

D. MCP Communication Workflow

The MCP Client acts as an intermediary between the Supervisor Agent and MCP Servers.

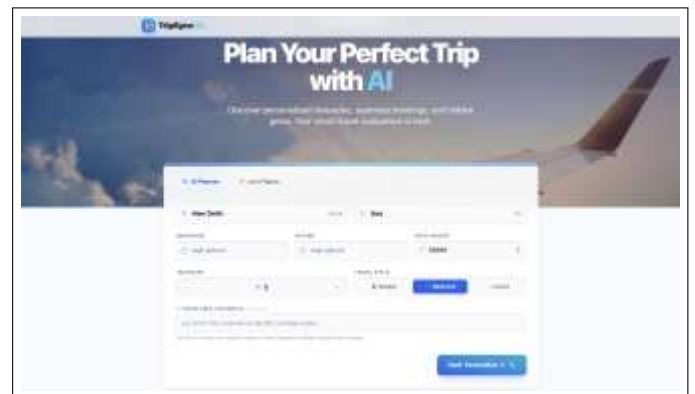


Fig. 4. TripSync User Interface

The communication workflow consists of the following steps:

- 1) Session Initialization
- 2) Tool Discovery
- 3) Service Invocation
- 4) Response Aggregation
- 5) Context Generation
- 6) Itinerary Creation

Communication occurs through JSON-RPC messages, al-

lowing standardized interaction between the MCP Host and external services.

E. Activity Flow

The activity diagram illustrates the complete workflow from user request submission to itinerary generation through planning, orchestration, and service invocation stages.

F. Sequence Flow

The sequence diagram depicts interactions among the frontend, backend, Planner Service, Supervisor Agent, MCP Client, MCP Servers, and the language model during itinerary generation.

G. Generated Itinerary

The generated itinerary presents transportation options, accommodation recommendations, tourist attractions, scheduling information, and estimated budget allocation in a structured format.

V. MATHEMATICAL MODEL AND ALGORITHMS

This section presents the mathematical formulation of the travel planning problem addressed by Trip Planner using MCP. The objective is to generate a personalized travel itinerary that satisfies user preferences while adhering to budget and time constraints.

A. Problem Definition

Given a user request, the system must determine:

- Transportation options
- Accommodation options
- Tourist attractions
- Budget allocation
- Activity scheduling

The generated itinerary should maximize user satisfaction while satisfying planning constraints.

B. Input Set

The input set is defined as:

$$I = \{O, D, B, T, P\} \quad (1)$$

where:

- O = Origin
- D = Destination
- B = Budget
- T = Travel Duration
- P = User Preferences

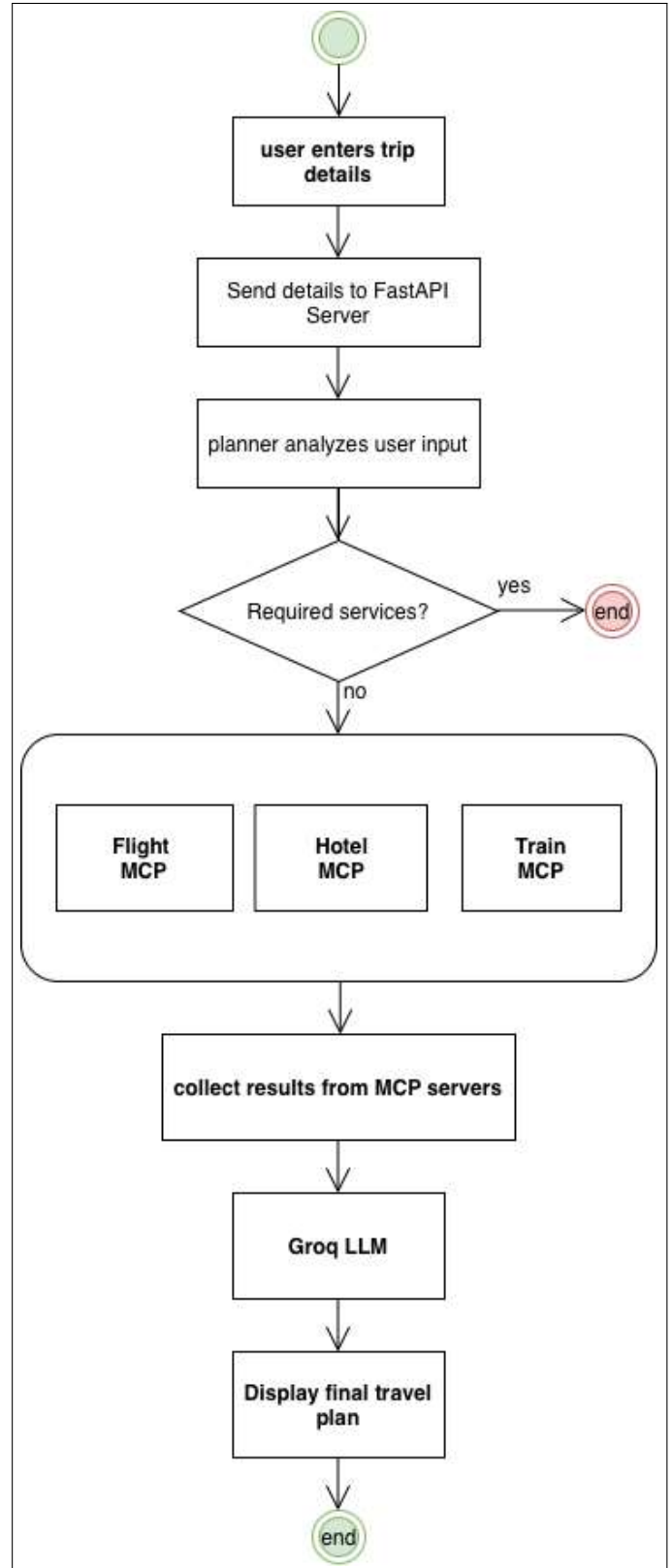


Fig. 5. Trip Planner using MCP Activity Diagram

C. Service Set

The available travel services are represented as:

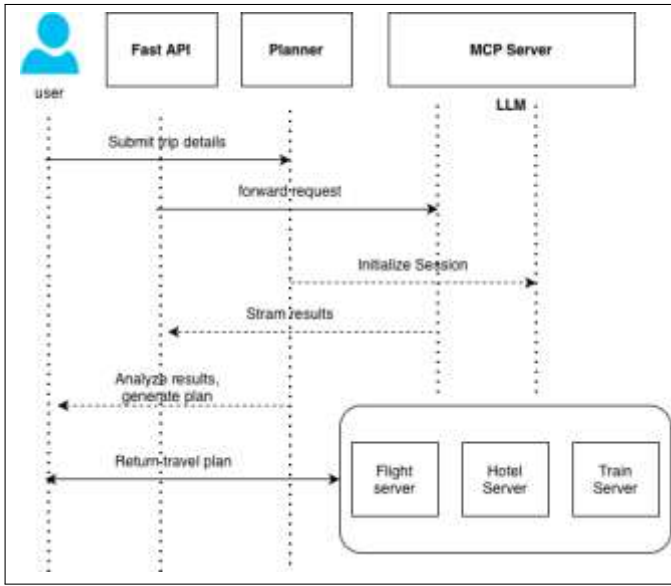


Fig. 6. Trip Planner using MCP Sequence Diagram



Fig. 7. Generated Travel Itinerary

$$S = \{F, H, R, BS, PL, BD\} \quad (2)$$

where:

- F = Flight Service
- H = Hotel Service
- R = Train Service
- BS = Bus Service
- PL = Places Service
- BD = Budget Service

D. Planning Function

The travel planning process is represented by:

$$P(I, S) = T_p \quad (3)$$

where:

- P = Planning Function
- I = User Inputs
- S = Selected Services
- T_p = Generated Travel Plan

E. Objective Function

The primary objective is to maximize user satisfaction:

$$\max U(T_p) \quad (4)$$

where:

- U = User Utility Function
- T_p = Generated Itinerary

The utility score depends on preference matching, cost efficiency, attraction coverage, transportation convenience, and accommodation quality.

F. Budget Constraint

The total estimated trip cost must remain within the user budget:

$$C_f + C_h + C_a \leq B \quad (5)$$

where:

- C_f = Transportation Cost
- C_h = Hotel Cost
- C_a = Activity Cost
- B = User Budget

G. Time Constraint

The generated itinerary must fit within the available travel duration:

$$\sum_{i=1}^n t_i \leq T \quad (6)$$

where:

- t_i = Duration of Activity i
- T = Total Available Time

H. Preference Constraint

The selected attractions should align with user interests:

$$\text{Match}(P, A) \geq \vartheta \quad (7)$$

where:

- P = User Preferences
- A = Selected Attractions
- ϑ = Similarity Threshold

I. Service Selection Model

The Supervisor Agent dynamically selects the required services:

$$S_r \subseteq S \quad (8)$$

where:

- S_r = Required Services
- S = Available Services

The selection function is defined as:

$$S_r = f(I) \quad (9)$$

where the selected services depend on user requirements.

Algorithm 1 Supervisor Agent Service Selection

```

1: Receive User Request
2: Extract Travel Parameters
3: Identify Required Services
4: for each Service do
5:   if Service is Relevant then
6:     Add Service to Active Set
7:   end if
8: end for
9: Invoke MCP Servers
10: Aggregate Responses
11: Send Context to LLM
12: Generate Final Itinerary
13: return Travel Plan

```

J. Travel Plan Score

Each generated itinerary is assigned a quality score:

$$Score = w_1P_m + w_2C_e + w_3A_c + w_4T_c \quad (10)$$

where:

- P_m = Preference Match Score
- C_e = Cost Efficiency
- A_c = Attraction Coverage
- T_c = Transportation Convenience
- w_i = Weight Factors

The itinerary with the highest score is selected.

K. Supervisor Agent Algorithm

The mathematical model provides a formal representation of travel planning in Trip Planner using MCP by incorporating user requirements, service selection, budget limitations, and preference-based optimization. The Supervisor Agent algorithm enables dynamic orchestration of MCP services while ensuring efficient and personalized itinerary generation.

VI. METHODOLOGY AND EXPERIMENTAL EVALUATION

This section presents the methodology used to develop and evaluate Trip Planner using MCP. The objective is to demonstrate how MCP-based service orchestration improves interoperability, modularity, and extensibility in intelligent travel planning systems.

A. Prototype Development

The Trip Planner using MCP prototype was implemented using a modular MCP-based architecture consisting of a FastAPI MCP Host, React frontend, Supervisor Agent, MCP Client, multiple travel service agents, and a relational database.

The prototype demonstrates service orchestration through standardized MCP communication and dynamic agent selection.

TABLE III
EXPERIMENTAL ENVIRONMENT

Component	Configuration
Frontend	React + TypeScript
Backend	FastAPI
Protocol	MCP
Database	SQLite / SQLModel
LLM	Gemini / Groq
Communication	JSON-RPC
Transport	SSE / HTTP

TABLE IV
ARCHITECTURE COMPARISON

Criterion	Traditional	TripSync
Integration Type	Custom APIs	MCP
Scalability	Medium	High
Maintainability	Medium	High
Extensibility	Low	High
Modularity	Medium	High

B. Experimental Environment

C. Evaluation Metrics

The framework was evaluated using the following criteria:

- Service Interoperability
- Scalability
- Modularity
- Maintainability
- Extensibility
- Context Awareness

These metrics focus on architectural quality and service orchestration effectiveness.

D. Architecture Evaluation

One of the primary objectives of Trip Planner using MCP is reducing integration complexity.

Let n represent the number of services.

Traditional service integration requires multiple custom interactions, resulting in complexity proportional to:

$$O(n^2) \quad (11)$$

In contrast, MCP introduces a unified communication layer, reducing integration complexity to:

$$O(n) \quad (12)$$

The comparison demonstrates the architectural benefits achieved through MCP-based orchestration.

E. Functional Evaluation

The prototype was evaluated by verifying support for major travel planning functions.

The results demonstrate that Trip Planner using MCP successfully integrates multiple travel services within a unified MCP-based framework.

TABLE V
FUNCTIONAL EVALUATION

Feature	Supported
Flight Search	Yes
Hotel Search	Yes
Train Search	Yes
Bus Search	Yes
Places Discovery	Yes
Budget Planning	Yes
Itinerary Generation	Yes
Trip Optimization	Yes

F. Discussion

The evaluation highlights several advantages of the proposed framework:

- Improved interoperability through standardized communication
- Reduced integration complexity
- Dynamic service orchestration
- Modular service expansion
- Simplified maintenance and extensibility

The Supervisor Agent further improves scalability by activating only the services required for a given travel request.

G. Threats to Validity

Several limitations should be acknowledged:

- Prototype-scale evaluation
- Limited real-world deployment
- Dependence on external travel APIs
- Dynamic variability of travel data

Future work should incorporate large-scale user studies and production-level deployments to further validate the proposed architecture.

VII. SECURITY ANALYSIS

Security is a critical consideration in MCP-based systems because language models interact directly with external tools, resources, and services. While MCP improves interoperability and extensibility, it also introduces new security challenges that must be addressed to ensure reliable and trustworthy operation.

A. Threat Analysis

The primary security threats relevant to Trip Planner using MCP are summarized in Table VI.

B. Security Mechanisms

To mitigate these threats, Trip Planner using MCP incorporates multiple security controls:

- JWT-based user authentication
- Role-based access control
- Input validation and request sanitization
- Secure session management
- Audit logging and activity monitoring
- Controlled access to MCP services

These mechanisms reduce the risk of unauthorized access and improve the overall security of the platform.

TABLE VI
SECURITY THREATS IN MCP-BASED SYSTEMS

Threat	Potential Impact
Tool Poisoning	Manipulated recommendations and incorrect outputs
Prompt Injection	Altered model behavior and unsafe responses
Credential Leakage	Unauthorized access to external services
Unauthorized Tool Invocation	Misuse of system functionality
Session Manipulation	Compromised communication sessions
Privacy Violations	Exposure of sensitive user information

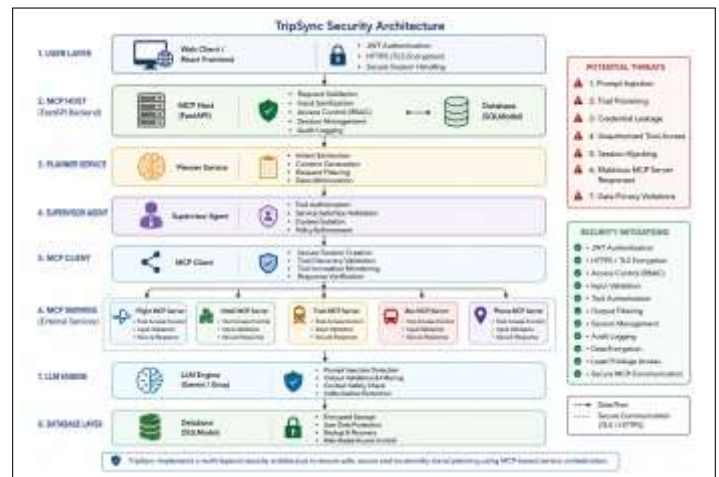


Fig. 8. Trip Planner using MCP Security Model

C. Threat Model

Figure 8 illustrates the security architecture adopted by TripSync.

Potential attack targets include MCP servers, MCP clients, API endpoints, databases, and user sessions. Therefore, security controls are applied across all layers of the architecture.

D. Privacy Considerations

Trip Planner using MCP processes user travel preferences and itinerary information. To protect user privacy, the system follows the following principles:

- Data minimization
- Encryption of sensitive information
- User consent management
- Secure data storage
- Controlled data retention policies

These practices help ensure compliance with privacy requirements and support secure deployment of MCP-based travel planning systems.

VIII. FUTURE WORK

Although Trip Planner using MCP successfully demonstrates MCP-based travel planning, several opportunities remain for future enhancements.

A. Additional MCP Servers

Future versions may integrate:

- Weather MCP Server
- Visa Assistance MCP Server
- Currency Exchange MCP Server
- Emergency Services MCP Server
- Event Recommendation MCP Server

These additions would improve itinerary quality and contextual awareness.

B. Real-Time Dynamic Planning

Current itinerary generation occurs before trip execution.

Future systems may support:

- Dynamic route replanning
- Weather-aware adjustments
- Traffic-aware scheduling
- Real-time activity recommendations

C. Multi-Agent Collaboration

Future work may investigate:

- Agent-to-Agent communication
- Cooperative planning
- Distributed decision making
- Autonomous task delegation

through protocols such as A2A and ACP.

D. Reinforcement Learning Optimization

Future versions may employ reinforcement learning techniques to optimize itinerary quality.

Potential objectives include:

- Cost minimization
- User satisfaction maximization
- Travel time reduction
- Resource utilization optimization

E. Large-Scale User Studies

Comprehensive user studies should evaluate:

- Usability
- Recommendation quality
- Personalization
- Trustworthiness

under real-world conditions.

F. Enterprise MCP Ecosystems

Beyond travel planning, the proposed architecture may be extended to:

- Healthcare
- Finance
- Education
- Smart Cities
- Industrial Automation

This demonstrates the broader applicability of MCP-enabled service orchestration.

IX. CONCLUSION

This paper presented *Trip Planner using MCP*, an intelligent travel planning framework that addresses interoperability and integration challenges in modern travel applications. The proposed system combines the Model Context Protocol (MCP), a Supervisor-Agent-based orchestration mechanism, multiple domain-specific MCP servers, and Large Language Models (LLMs) to generate personalized and budget-aware travel itineraries.

The implementation demonstrates how MCP can be utilized as a standardized communication layer for integrating heterogeneous travel services, including flights, hotels, trains, buses, tourist attractions, and budget planning modules. By decoupling service providers from the application logic, the framework improves modularity, scalability, maintainability, and extensibility compared to traditional API-centric travel systems.

Experimental evaluation indicates that the proposed architecture successfully aggregates travel information from multiple services and generates coherent, context-aware itineraries based on user preferences and constraints. The results highlight the effectiveness of MCP in enabling dynamic service orchestration while reducing integration complexity and improving personalization capabilities.

Overall, *Trip Planner using MCP* demonstrates the practical applicability of MCP in real-world intelligent systems and contributes a domain-specific implementation to the growing MCP ecosystem. Future work will focus on integrating real-time travel APIs, weather-aware recommendations, enhanced security mechanisms, and advanced optimization techniques to further improve travel planning performance and user experience.

REFERENCES

- [1] X. Hou, Y. Zhao, S. Wang, and H. Wang, "Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions," *arXiv preprint arXiv:2503.23278v2 [cs.CR]*, 2025.
- [2] A. Ehtesham, A. Singh, S. Kumar, and G. K. Gupta, "A Survey of Agent Interoperability Protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP)," *arXiv preprint arXiv:2505.02279v1*, 2025.
- [3] N. Krishnan, "Advancing Multi-Agent Systems Through Model Context Protocol: Architecture, Implementation, and Applications," *arXiv preprint arXiv:2504.21030v1*, 2025.
- [4] M. M. Hasan et al., "Model Context Protocol (MCP) at First Glance: Studying the Security and Maintainability of MCP Servers," *arXiv preprint arXiv:2506.13538v4*, 2025.
- [5] H. Guo et al., "A Measurement Study of Model Context Protocol Ecosystem," *arXiv preprint arXiv:2509.25292v2*, 2025.
- [6] P. P. Ray, "A Survey on Model Context Protocol: Architecture, State-of-the-art, Challenges and Future Directions," *TechRxiv*, Apr. 2025.
- [7] P. Pajo, "Smithery.ai: A Model Context Protocol for Enhanced Large Language Model Integration and Cross-Industry Applications," *ResearchGate*, Mar. 2025.
- [8] V. S. Narajala and I. Habler, "Enterprise-Grade Security for the Model Context Protocol (MCP): Frameworks and Mitigation Strategies," *arXiv preprint arXiv:2504.08623v2*, 2025.
- [9] Q. Li and Y. Xie, "From Glue-Code to Protocols: A Critical Analysis of A2A and MCP Integration for Scalable Agent Systems," *arXiv preprint arXiv:2505.03864v1*, 2025.

- [10] H. Li, Y. Xu, and T. Hong, "EnergyPlus-MCP: A model-context-protocol server for ai-driven building energy modeling," *SoftwareX*, vol. 32, p. 102367, 2025.
- [11] Z. Li et al., "NGENT: Next-Generation AI Agents Must Integrate Multi-Domain Abilities to Achieve Artificial General Intelligence," *arXiv preprint arXiv:2504.21433v1*, 2025.
- [12] G. Chhetri et al., "Model Context Protocols in Adaptive Transport Systems: A Survey," *arXiv preprint arXiv:2508.19239v1*, 2025.
- [13] N. Arora, M. R. Shashank, V. Manasa, C. M. Metri, and H. S., "Full Stack Trip Planner Using AI," *International Journal of Research Publication and Reviews*, vol. 6, no. 5, pp. 13444–13449, 2025.
- [14] E. L. Gonza'lez-Sanz, I. Cantador, and A. Bellog'in, "LLM-based Generation of Personalized, Context-aware City Tourist Itineraries: A User Study with GPT Trip Planner," in *Proc. Workshop on Recommenders in Tourism (RecTour 2025)*, 2025.
- [15] Anthropic, "Introducing the Model Context Protocol," *Anthropic News*, Nov. 2024. [Online]. Available: <https://www.anthropic.com/news/model-context-protocol>
- [16] L. Wang et al., "A survey on large language model based autonomous agents," *Frontiers of Computer Science*, vol. 18, no. 6, 2024.
- [17] Z. Xi et al., "The rise and potential of large language model based agents: A survey," *arXiv preprint arXiv:2309.07864*, 2023.
- [18] OpenAI, "Function calling in OpenAI models," *OpenAI Documentation*, 2023. [Online]. Available: <https://platform.openai.com/docs/guides/function-calling>
- [19] T. Schick et al., "Toolformer: Language models can teach themselves to use tools," in *NeurIPS*, 2023.
- [20] S. Yao et al., "ReAct: Synergizing reasoning and acting in language models," in *ICLR*, 2023.
- [21] Google, "Agent2agent (a2a) protocol documentation," 2024. [Online]. Available: <https://google.github.io/A2A/>
- [22] IBM BeeAI, "Introduction to agent communication protocol (acp)," 2024. [Online]. Available: <https://docs.beeai.dev/acp/alpha/introduction>
- [23] Agent Network Protocol Contributors, "Agent network protocol (anp)," 2024. [Online]. Available: <https://github.com/agent-network-protocol/AgentNetworkProtocol>
- [24] S. Zhou et al., "WebArena: A Realistic Web Environment for Building Autonomous Agents," in *Proc. ICLR*, 2024.