CulTour

AI-powered Smart Tourism Application

Group 5: deptrAI

December 17, 2025

Contents

1 Team Information

No.	Full Name	Student ID	Role	Responsibilities
1	Le Dat Phuc An	24125052	Team Leader	AI
2	Vong Chi Van	24125049	Tech Leader	AI
3	Tran Ton Minh Ky	24125102	Deputy Leader	Backend
4	Pham Tien Dat	24125027	Member	Frontend
5	Nguyen Minh Quan	24125040	Member	Data
6	Pham Nguyen Minh Quan	24125041	Member	Data
7	Quach Thien Lac	24125092	Member	Frontend

2 Idea

The core concept of **CulTour** is to re-engineer how cultural tourism is consumed by shifting the paradigm from *Location-Based Services* to a **Semantic and Visual Discovery Engine**.

While the global travel industry is shifting towards experiential travel, digital tools have remained stagnant. Current platforms treat cultural sites as static geographical points, ignoring the rich historical and sociological context that defines them. CulTour decouples "culture" from mere location, breaking it down into granular, queryable data points such as *Architecture Style*, *Dynastic Era*, *Religious Affiliation*, and *Social Function*.

By treating cultural attributes as structured data rather than unstructured text, CulTour aims to solve the "Staged Authenticity" problem. It empowers users to bypass the curated "front-stage" of mass tourism and algorithmically locate the "back-stage" reality of Vietnamese daily life using a **Utility-First** approach.

2.1 Problem Statement

The tourism sector in Vietnam suffers from a dual failure: a sociological failure of authenticity and a technical failure of information categorization.

2.1.1 The Sociological Problem

Cultural tourism often presents a sanitized, curated performance designed to meet perceived tourist expectations. This creates a "front-stage" reality where visitors interact with a manufactured version of Vietnam, while the "back-stage"—the authentic, unstaged aspects of local life—remains hidden.

- **The Gap:** Guided tours (e.g., "We Show You Saigon!") focus exclusively on high-traffic, commercialized zones.
- **The Demand:** There is a massive market friction; 77% of travelers

explicitly seek authentic experiences[1] [2], yet the market primarily supplies generic sightseeing.

2.1.2 The Technical Problem

The primary barrier to finding authentic experiences is not a lack of existence, but a lack of *discoverability*. Market leaders like Google Maps, Agoda, and Traveloka are optimized for logistics and commerce, leading to a "flattening" of cultural value.

1. **The "Pagoda Problem" (Taxonomic Failure):** To a standard map algorithm, a 19th-century pagoda with rare Sino-Vietnamese architecture is categorized identically to a newly built concrete pagoda. The platforms lack the semantic depth to distinguish *historical significance* from *utility*.
2. **Algorithmic Bias via Popularity:** Recommendation engines on current platforms prioritize venues with high traffic and English reviews. This creates a feedback loop that continually pushes tourists toward "tourist traps," burying authentic, quieter locations under a layer of noise.

2.1.3 The Information Asymmetry

Foreign travelers face a "knowledge wall".

- **Local Sources:** High-quality blogs analyzing deep culture are predominantly written in Vietnamese, often verbose, and lack structured metadata.
- **English Sources:** Available English resources are often superficial, repetitive, and lack the specific domain knowledge required to explain *why* a location is culturally significant.

2.2 Target Users

We define our users not just by demographics, but by their search behavior and intent

2.2.1 The Authenticity Seeker

Independent travelers or expatriates who want to understand the *context* of what they are seeing. They are unsatisfied by the generic nature of Google Maps categories and are looking for a tool that filters for specific cultural nuance (e.g., "Find me a temple that is currently active, not just a museum").

[1] Booking.com 2025 Travel Predictions
[2] Hilton 2025 Slow Travel

2.2.2 The Culturally Overwhelmed

This user wants to engage with local culture but lacks the time or research skills to parse through hundreds of Vietnamese blogs. They find the current volume of unstructured information overwhelming and require a curated, trusted filter to guide them to high-value experiences.

2.3 System Objectives

To bridge the gap between foreign curiosity and local reality, the system has four distinct objectives:

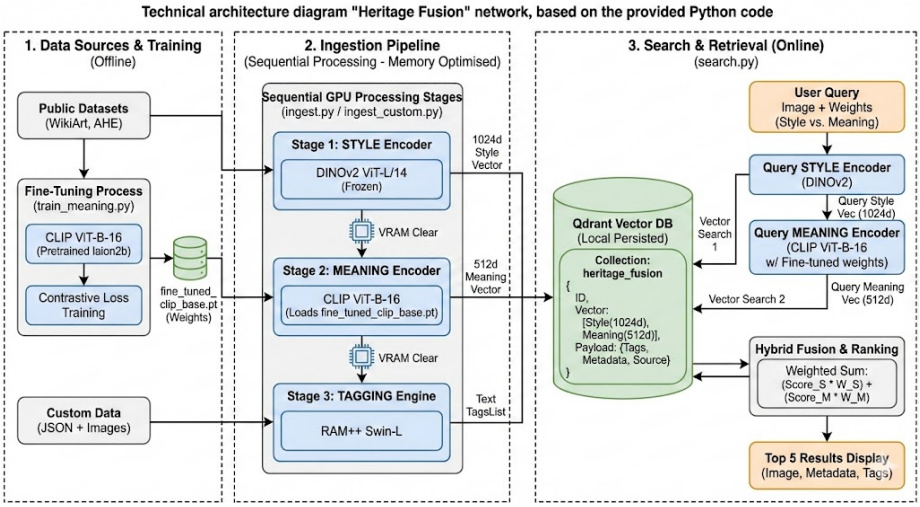
- **Construct a Cultural Knowledge Graph**: the system must move beyond keyword matching. It aims to build a semantic engine that understands the relationships between cultural entities.
- **Granular, Attribute-based Search**: the system must solve the "Pagoda Problem" by allowing users to filter based on deep attributes rather than surface-level categories.
- **Automated Curation via Preference Matching**: to address the "overwhelmed" user, the system will utilize user constraints to auto-curate itineraries.
- **The "Utility-First" Adoption Strategy**: we assume that organizers of authentic local events are difficult to onboard immediately.

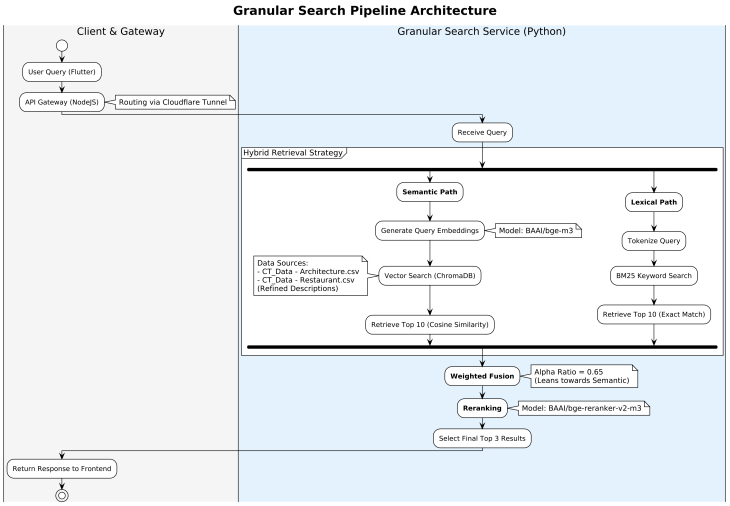
3 System Overview

3.1 Core Functions

3.1.1 Granular Search Service

The search pipeline is built upon a hybrid architecture that combines semantic search and lexical search techniques to optimize both relevance and precision in results. The key components of this pipeline are as follows:





3.1.2 Community Event Service

This module functions as a dynamic bridge between local culture providers and users, fostering active participation rather than passive observation.

- **Passive Discovery Engine:** The system automatically surfaces active festivals and events by correlating temporal data with location-specific tags. For example, a user browsing a specific pagoda will be notified of upcoming religious rites relevant to that location's history.
- **Decentralized Event Hosting:** Cultural organizations and community leaders are provided with a "Self-hosting" portal. This allows them to

create, manage, and broadcast community-driven events directly to interested users, bypassing traditional media gatekeepers.

- **Cultural Differentiation:** Unlike standard event aggregators, this service focuses exclusively on participatory cultural activities, distinguishing the platform as a hub for deep community engagement rather than general entertainment.

3.1.3 Community Event Service

This module functions as a dynamic bridge between local culture providers and users, fostering active participation rather than passive observation.

- **Passive Discovery Engine:** The system automatically surfaces active festivals and events by correlating temporal data with location-specific tags. For example, a user browsing a specific pagoda will be notified of upcoming religious rites relevant to that location's history.
- **Decentralized Event Hosting:** Cultural organizations and community leaders are provided with a "Self-hosting" portal. This allows them to create, manage, and broadcast community-driven events directly to interested users, bypassing traditional media gatekeepers.
- **Cultural Differentiation:** Unlike standard event aggregators, this service focuses exclusively on participatory cultural activities, distinguishing the platform as a hub for deep community engagement rather than general entertainment.

3.1.4 Recommendation System

The recommendation engine employs a hybrid filtering approach to personalize the user experience, transforming raw interaction data into curated cultural journeys.

- **Explicit & Implicit Feedback Loops:** The system captures explicit signals (likes, dislikes) and implicit behaviors (visit history, dwell time, view counts). These inputs are used to build a dynamic user preference profile.
- **Preference Clustering:** By analyzing frequently selected tags (e.g., "Gothic Architecture," "Cham Culture") and visited locations, the algorithm identifies user cohorts with similar tastes. This allows for collaborative filtering, where a user is recommended niche locations that were highly rated by others in their "taste cluster," facilitating serendipitous discovery of new cultural sites.

3.2 Innovation Highlights

Unified Multimodal Search Interface: Unlike traditional travel platforms restricted to rigid keyword inputs, our system enables a seamless search-by-anything experience. Users can discover locations using natural language prompts, visual inputs, categorical filters, or direct entity names. This flexibility bridges the gap between vague user intent and specific cultural data.

Table 1: Examples of Multimodal Search Capabilities

Search Modality	**Example User Inputs**
Natural Language	"Quiet places in Saigon to read a book with colonial architecture" "Where can I see traditional water puppet shows?"
Visual Input	*[User uploads photo of Notre Dame Cathedral]* $\rightarrow$ System finds similar Gothic/Romanesque churches.
Categorical Filters	{ `"arch_style": "French Colonial", "district": "1"` } { `"religion": "Buddhism", "active_worship": true` }
Direct Entity Lookup	"Ben Thanh Market" "Ho Chi Minh City Fine Arts Museum"

4 Sequence Diagram

4.1 Sign in

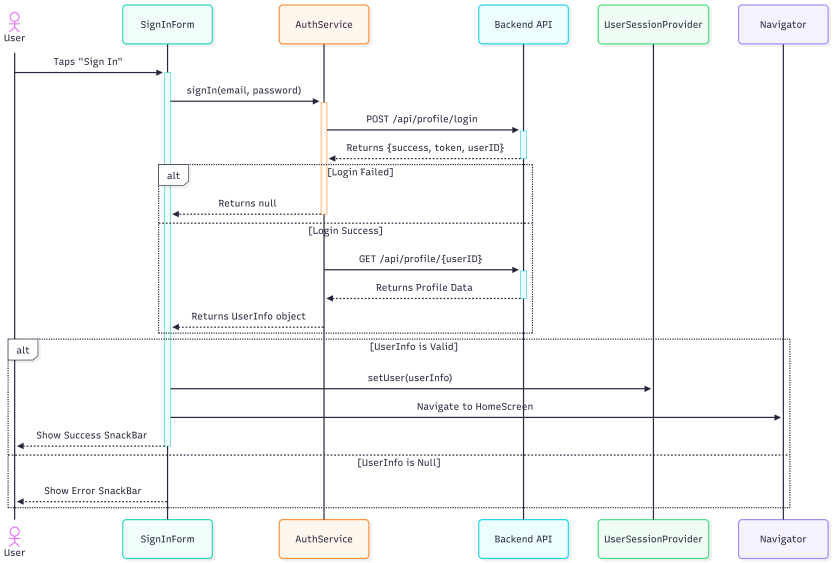


Figure 1: Sign in sequence diagram

4.2 Sign up

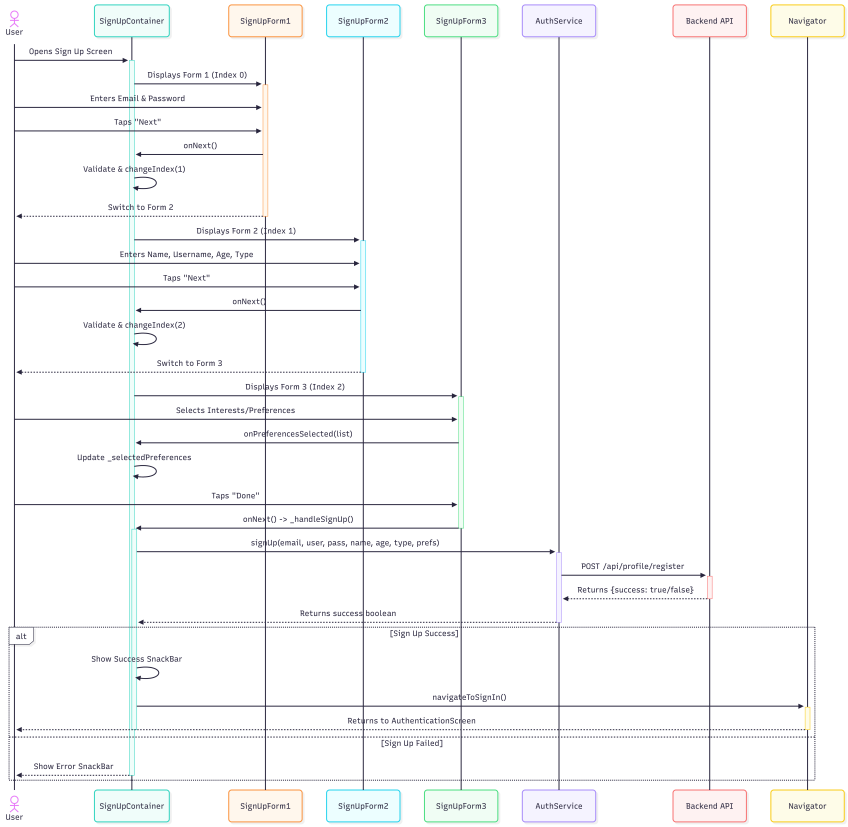


Figure 2: Sign up sequence diagram

4.3 Create Event

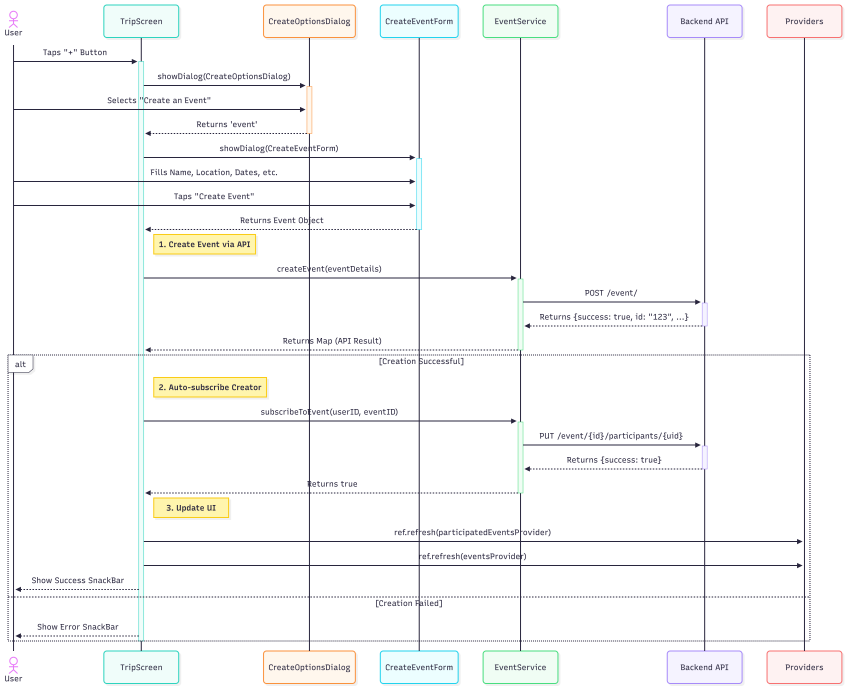


Figure 3: Create event sequence diagram

4.4 Subscribe Event

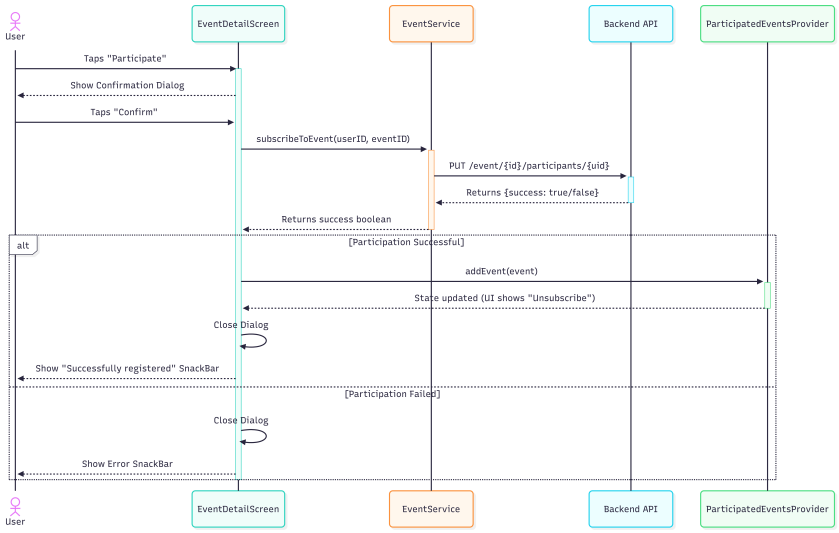


Figure 4: Subscribe event sequence diagram

4.5 Search Places

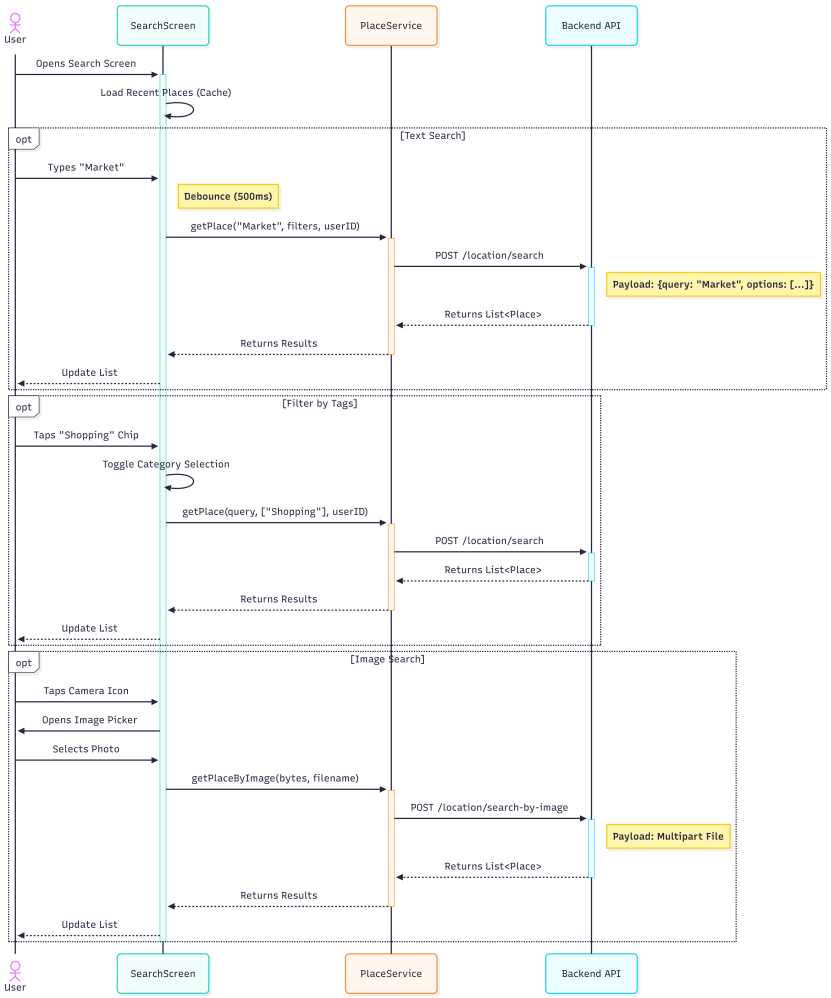


Figure 5: Search places sequence diagram

4.6 Save Location

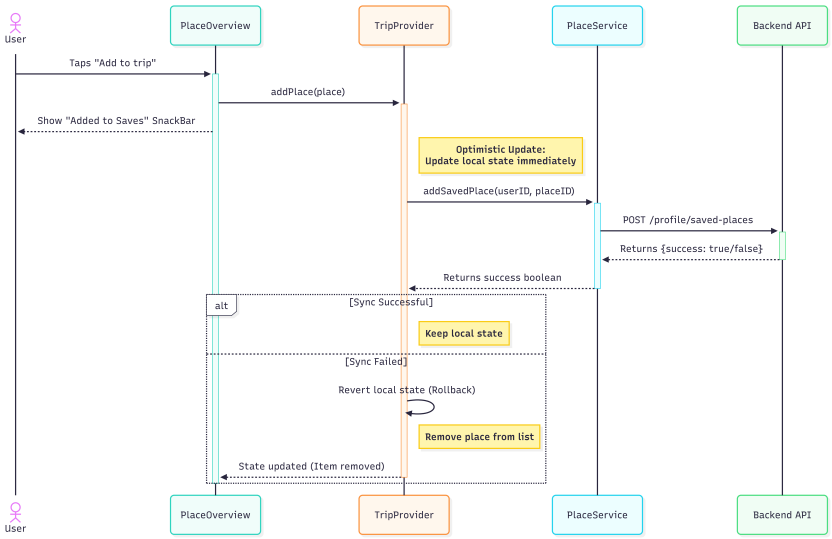


Figure 6: Save location sequence diagram

4.7 Map Feature

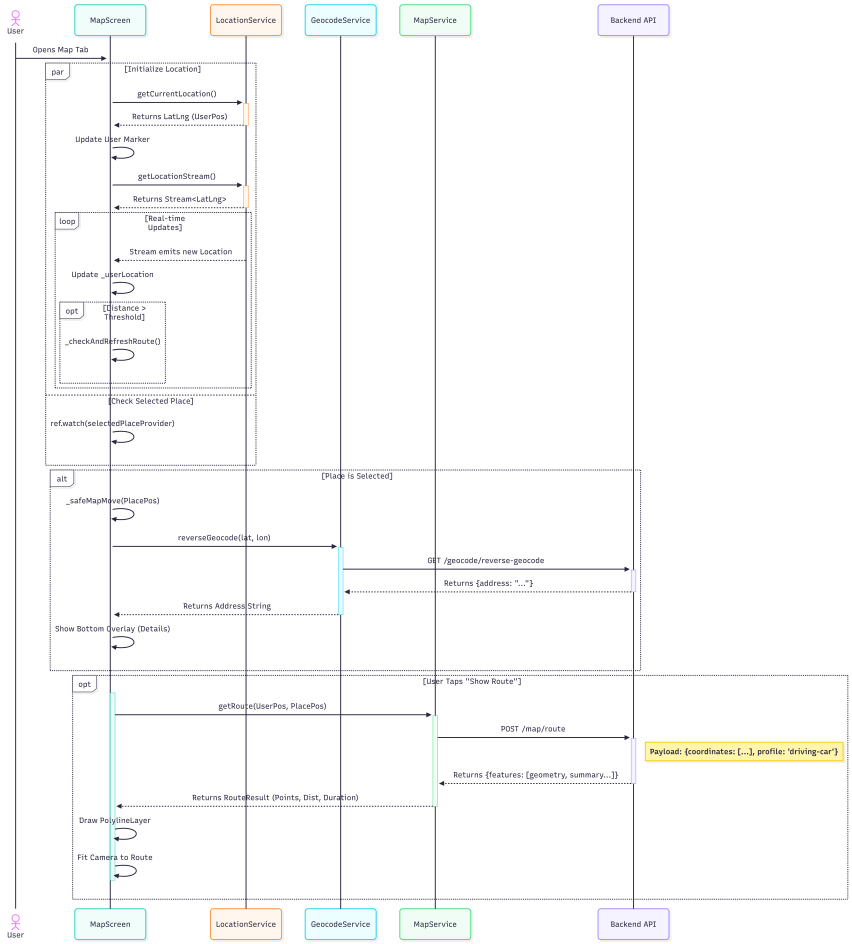


Figure 7: Map feature sequence diagram

5 Class Diagram

5.1 Architecture Overview

Our system's architecture employs a foursome design pattern, dividing the application into four main layers: the AI backend, the backend, the database, and the frontend. Each layer has distinct responsibilities and interacts with other layers to ensure seamless functionality.

5.2 Class Diagrams

5.2.1 Overall Architecture

Figure 8: Overall architecture.

The class diagram above illustrates the overall architecture of the system. It shows a Flutter frontend with UI screens (MyApp, IntroductionScreen, HomeScreen, SearchScreen, RegionOverview, PlaceOverview, MapScreen) connected by navigation edges. State is managed via Riverpod providers: UserSessionProvider (user state and preferences), SelectedPlaceProvider (current place), and NavigationIndexProvider (current tab). Service layer classes (PlaceService, ProfileService) wrap Dio HTTP calls to fetch places (by query, ID, or image) and user profiles. Models include Place, Region, User, and enum types FilterType and CategoryType. Screens read or set provider state: Home/Search/Region read UserSession; PlaceOverview sets SelectedPlace; MapScreen listens to SelectedPlace; Home sets NavigationIndex. Services supply data to UI and providers, and models represent the domain entities and filters used across the app.

5.2.2 AI Backend, Backend, and Database

Figure 9: Frontend architecture.

The Flutter frontend boots via ‘MyApp‘, routing first to ‘IntroductionScreen‘ and then into the main flow (Splash, Home, Map, Profile, Trip). UI screens read Riverpod providers for navigation (‘NavigationProvider‘), selection (‘SelectedPlaceProvider‘), user/session (‘UserInfoProvider‘), trips (‘TripProvider‘), and events. Screens invoke service classes for API calls: ‘PlaceService‘, ‘EventService‘, ‘GeocodeService‘, ‘MapService‘, ‘ReviewService‘, and ‘AuthService‘, with ‘SearchCacheService‘ caching queries locally. All HTTP calls share a Dio client; ‘ServiceHelpers‘ centralizes error handling and token refresh before updating session state through providers.

Figure 10: AI backend architecture.

The ai$_b$$ackendisaFastAPIappthatexposesarecommendationendpoint.Theentrypoint(Main)createsthe1$

Figure 11: Backend architecture.

The backend is an Express app that wires routes to controllers and services. The entry ‘App‘ holds the Express instance and registers ‘RecommendationRoutes‘, which map HTTP handlers to ‘RecommendationController‘. Controllers delegate to ‘RecommendationService‘ to call the Python AI backend via Axios and wrap responses in ‘ServiceResponse‘. Enmap-backed stores (‘UserDB‘, ‘LocationDB‘, ‘EventDB‘) provide simple persistence layers for user state, geocoded locations, and events. ‘RecommendationService‘ depends on these stores to enrich requests and cache results, while also calling the external ‘PythonRecAPI‘ for recommendations and feedback.

Figure 12: Database architecture.

The data layer loads CSVs into DataFrames, builds text+metadata docs, and keeps stable IDs (geo keys or MD5 fallback). BaseColInfo centralizes this; ArchColInfo and RestaurantColInfo tailor narratives for architecture and food. Outputs feed ChromaDB (vector) and BM25 (keyword) search, fused and reranked before the FastAPI /search endpoint returns scored items.

6 Test Cases and Scenarios

6.1 Stress Cases

We conducted stress tests to evaluate the system's performance limits and latency under heavy load.

- **High Volume User Load:** We flooded the user base with 1 million simulated users to check query response times. The system remained stable, with response times consistently remaining **under 5ms**.
- **Concurrent Search Capacity:** Stress testing revealed a specific throughput limit for search operations; the system can currently handle 87 searches simultaneously.
- **Authentication Scalability:** We sent multiple concurrent sign-in/sign-up requests to test scalability. The response time was found to scale linearly with the volume of requests. At a load of 200 concurrent requests, the response time was recorded at 1000ms.

6.2 Edge Cases

We evaluated specific edge cases to check the routing logic and search interpretation capabilities.

- **Remote Routing:** We tested the routing algorithms against remote geographic locations. The result showed that coverage is not yet universal, as some remote places could not be successfully routed.
- **Sophisticated Search Prompts:** We input niche and complex search prompts to test the search engine's semantic understanding. These prompts were processed successfully, indicating the system handles semantic search intent correctly.

7 Completed Feature

Throughout the 10 weeks of development, the application has undergone multiple stages and underhauls. Throughout every development iteration, we aim to pack in as much features as possible. These features include

- A functional profiling system, including secure registration and login, user data, and a stable URI that every user has pointing to their user data.
- Map display and highlights for locations and routes.
- AI-powered tools that fetches a user's possible interested places for recommendation.
- Events creation and participation
- Geocoding, reverse-geocoding, route finding and nearby location search.
- AI-powered location searching to ensure accuracy in semantics and syntax.

8 Planning

8.1 Phase 1: Minimum Viable Product (MVP)

The initial phase focuses on establishing the foundational features that differentiate Cultour from conventional travel applications, emphasizing intelligent search capabilities and community-driven event discovery.

8.1.1 Granular Search System

- **Multi-modal Search Implementation**: Deploy advanced search functionality supporting text queries, category-based filters, and image-based location recognition.
- **Technical Implementation**: Integration of fuzzy search algorithms and semantic search powered by Python backend services (`/search` and `/search-by-image` endpoints).
- **AI-Powered Query Processing**: Leverage natural language processing to interpret user intent and return contextually relevant results.

- **Filter Architecture**: Implement comprehensive filtering system including tags, location proximity, operating hours, and venue age/historical significance.

8.1.2 Intelligent Recommendation System

- **Personalization Engine**: Develop machine learning models to analyze user preferences, search history, and interaction patterns.
- **Collaborative Filtering**: Implement algorithms to suggest locations and events based on behavior of similar user profiles.
- **Preference Management**: Create user preference storage and updating mechanisms integrated with user session management.
- **Dynamic Suggestions**: Real-time recommendation updates based on user location, time of day, and seasonal factors.

8.1.3 Event Services Platform

- **Event Discovery**: Comprehensive event browsing interface with categorization and filtering capabilities.
- **Event Creation**: Full-featured event creation workflow allowing users to establish public or private events with detailed metadata (name, location, description, start/end times, participant limits).
- **Event Participation**: User subscription system enabling participants to join/leave events with real-time participant tracking.
- **Event Management**: CRUD operations for event lifecycle management through `EventService` API.

8.2 Phase 2: Scaling & Community Development

This phase transitions Cultour from a functional application to a comprehensive travel community platform capable of handling significant user traffic and fostering user-generated content.

8.2.1 Social Hub Development

- **User Review System**: Implement five-star rating system with text reviews, photo attachments, and review verification mechanisms.
- **User-Generated Itineraries**: Enable users to create, share, and publish custom travel itineraries with day-by-day breakdowns and location recommendations.
- **Travel Journals**: Develop journaling functionality allowing users to document trips with multimedia content, timestamps, and geolocation data.

8.2.2 Infrastructure Scaling

- **Load Balancer Implementation**: Deploy horizontal scaling architecture using containerized services (Docker infrastructure already established in codebase).
- **Database Optimization**: Implement database sharding, indexing strategies, and caching layers to manage high-volume queries.
- **CDN Integration**: Utilize Content Delivery Networks for image and static asset distribution to reduce latency.
- **API Gateway**: Establish API gateway with rate limiting, request routing, and microservices orchestration.

8.2.3 Geographic Expansion

- **Regional Data Collection**: Systematic gathering of location data from underrepresented regions throughout Vietnam.
- **Data Source Integration**: Partnership with local tourism boards, cultural organizations, and government databases.
- **Crowdsourced Data**: Enable verified users to contribute location information, photos, and metadata with quality control processes.
- **Multi-language Support**: Implement internationalization for Vietnamese, English, and additional languages.

8.3 Phase 3: Monetization Strategy

Upon establishing a stable user base and comprehensive content library, Phase 3 introduces revenue generation mechanisms while preserving platform accessibility.

8.3.1 Premium Event Listings

- **Organizational Accounts**: Create business-tier accounts for tourism organizations, hotels, and event companies.
- **Featured Event Placement**: Offer prominent positioning in search results and recommendation feeds for a service fee.
- **Analytics Dashboard**: Provide event organizers with participant demographics, engagement metrics, and conversion analytics.
- **Tiered Pricing Model**: Implement flexible pricing based on event visibility duration, geographic reach, and promotional features.

8.3.2 Affiliate Marketing Integration

- **Partnership Establishment**: Develop affiliate relationships with major booking platforms (Agoda, Klook, Booking.com).
- **Deep Linking Implementation**: Seamless integration enabling users to book accommodations and activities directly through Cultour.

8.4 Partner Verification & Quality Assurance

To maintain platform credibility and ensure user safety, a comprehensive partner verification system will be implemented.

8.4.1 Verification Process Architecture

- Multi-Tier Verification System.

8.4.2 Verification Requirements

- Documentation Standards.
- Review Process.

8.4.3 Badge & Trust System

- Verification Badges.
- Graduated Privileges.
- Ongoing Monitoring.

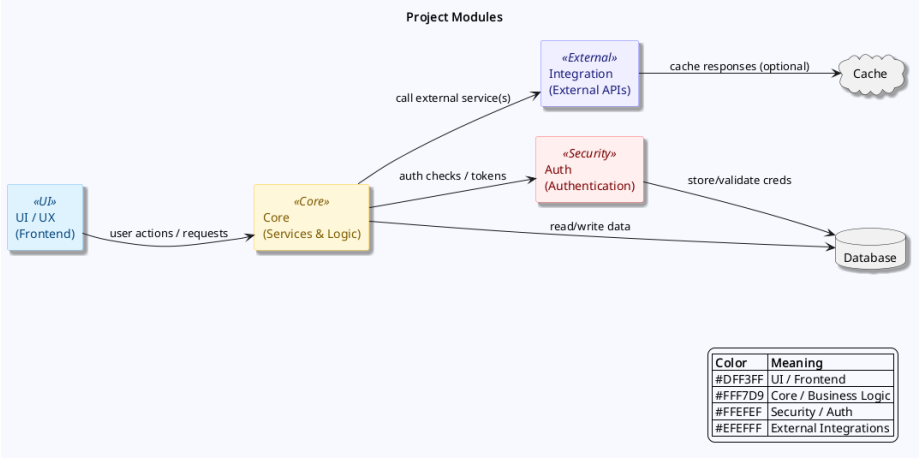
9 Technical Solution and Architecture

The project, in its core, is a Flutter mobile application that utilizes a NodeJS backend with assistance from Python backend servers that form into a single clean API that the frontend can use. The project's overall architecture can be split into three main architecture styles: the system-level architecture, the client architecture and the backend architecture.

System level architecture. The interaction between the frontend and the backend is a classic three-tier client-server architecture. More specifically, the Flutter client, which we call the *presentation*, utilizes the Node and Python API (the *application*). The backend then utilizes the database and external services to parse upstream data and pass it to the downstream clients. This is the simplest architecture for an MVP achievable within a reasonable time frame.

Client architecture. The frontend is primarily developed vertically via a feature-first architecture. The codebase can be grouped by separate features (including but not limited to login, map screen, trips, saved place, location searching), each has its own logic and UI implementation. We choose this architecture because it is a simple and intuitive architecture primarily derived from the user flow. This ensures ease of development and understanding from the team.

Backend architecture. The backend is a NodeJS express app comprising of multiple endpoints grouped into categories callable from the frontend via dio. In its core, it is an MVC architecture. An endpoint is structured by the core logic's within its service function, express handling and parameters fetching within its controller functions, and routing via an express's Router object. Middlewares for authorization and checking are also included, allowing decoupling and reducing code duplication. Other minor backend servers are also made for training AIs and AI-based searching, and the main backend calls these servers via Cloudflare tunnels. This allows for separation of duties and dependency decoupling.



10 Conclusion and Future Development

10.1 Future Development

While the current MVP is restricted to the geographic scope of Ho Chi Minh City, the system architecture is designed for scalability. Our roadmap focuses on overcoming two primary challenges: **Data Density** (populating the map) and **Event Verification** (ensuring trust).

10.1.1 Scalability Strategy

To handle the projected growth from the MVP phase to mass adoption, the backend will evolve through staged upgrades:

- **Infrastructure Migration:** Transitioning from basic hosting to a Virtual Private Server (VPS) to manage increased traffic.
- **Load Balancing:** Upon reaching a user base of 10,000+, we will implement load balancers to distribute traffic and support up to 100,000 concurrent searches, ensuring latency remains low during high-demand queries.

10.1.2 Expansion via AI-Driven Data Extraction

Expansion to other regions (e.g., Hanoi, Da Nang) relies on solving the **Data Density** problem without requiring manual entry for every site. We propose a semi-supervised learning pipeline:

1. **Manual Labeling:** Creating a high-quality, manually labeled dataset of cultural sites in Ho Chi Minh City.
2. **Model Training:** Using this data to train an Image Tag Extraction model capable of identifying architectural and cultural attributes (e.g., distinguishing "Sino-Vietnamese" from "Khmer" architecture).
3. **Automated Expansion:** Leveraging the Google Maps API to fetch images from new regions and passing them through our trained model. This allows us to rapidly populate the database with "semantic tags" for locations across Vietnam with a target accuracy of 90%.

10.1.3 Community Governance and Verification

As the platform introduces **User-Organized Events**, maintaining the platform's core value proposition—authenticity—is critical.

- **Current State (Manual Moderation):** To prevent spam and ensure safety, all user-submitted events currently undergo manual review.
- **Future State (Community Voting):** As the active user base grows (targeting 1,000 monthly active users), we will transition to a decentralized trust model. A "Community Voting" system will allow trusted users to verify events, reducing the bottleneck of central moderation.

10.1.4 Roadmap & Monetization

The long-term sustainability of CulTour relies on a three-phase evolution:

- **Phase 1 (MVP):** establishing the Granular Search and Recommendation System.

- **Phase 2 (Social Hub):** introducing user-generated itineraries, travel journals, and Q&A forums to build a sticky community.
- **Phase 3 (Monetization):** implementing revenue streams via small fees for organization-led events and affiliate marketing integrations with major aggregators like Agoda or Klook.

10.2 Conclusion

The tourism industry's current reliance on generic, location-based data has created a "Staged Authenticity" trap, where travelers are funneled into curated performances rather than genuine experiences.

CulTour addresses this by providing the necessary data layer to decouple culture from mere geography. By creating a tool that understands the semantic nuances of architecture, religion, and history, we empower travelers to bypass the "front-stage" and discover the "back-stage" reality of Vietnam.

Our ultimate vision is not just to build a better search engine, but to foster a deeper appreciation of Vietnamese heritage. We hope that by reducing the friction of discovery, more travelers will confidently explore local culture, moving from passive observation to active, first-hand experience.