INSTRUCTOR'S COMMENTS

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Da Nang, June 19th, 2025
Instructor

PhD. Pham Cong Thang

REVIEWER'S COMMENTS

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

THE UNIVERSITY OF DA NANG
UNIVERSITY OF SCIENCE AND TECHNOLOGY
INFORMATION TECHNOLOGY FACULTY

THE SOCIALIST REPUBLIC OF VIET NAM
Independence - Freedom - Happiness

GRADUATION PROJECT COMMENT

I. General information:

1. Student name - Class - Student ID:

Pham Trung Hieu - 21TCLC_Nhat1 - 102210034

Nguyen Thanh Minh - 21TCLC_DT1 - 102210067

2. Topic title: TripWise Website - Intelligent Destination and Route Recommendations for Journey.

3. Instructor: Pham Cong Thang Academic title/degree: PhD

II. Reviews of graduation project:

1. About the urgency, novelty, usability of the topic: (2 points)

.....

2. About the results of solving the tasks required by the project: (4 points)

.....

3. About the form, structure and layout of the graduation project: (2 points)

.....

4. The topic includes scientific value/article/problem solving of the enterprise or school: (1 point)

.....

5. Existing shortcomings need to be supplemented or modified:

.....

III. Spirit and attitude of the student (1 point):

.....

.....

IV. Evaluation:

1. Evaluation point: …./10

2. Suggest: Defense permitted/ Edit to defend/ Defense not permitted

Da Nang, June 19th, 2025

Instructor

THE UNIVERSITY OF DA NANG
UNIVERSITY OF SCIENCE AND TECHNOLOGY
INFORMATION TECHNOLOGY FACULTY

THE SOCIALIST REPUBLIC OF VIET NAM
Independence - Freedom - Happiness

GRADUATION PROJECT REQUIREMENTS

Student Name - Student ID - Class:

Pham Trung Hieu - 102210034 - 21TCLC_Nhat1

Nguyen Thanh Minh - 102210067 - 21TCLC_DT1

Faculty: Information Technology

1. Topic title: TripWise Website - Intelligent Destination and Route Recommendations for Journey.

2. Project topic: ☐ has signed intellectual property agreement for result.

3. Initial figure and data: Places' information from TripAdvisor.

Content of explanations and calculations: None.

4. The content of the thesis includes:

INTRODUCTION – This chapter provides an overview of the project's context, objectives, and the specific problems that will be addressed in the thesis. It outlines the scope of the issues that will be explored and analyzed.

CHAPTER 1: THEORIES AND TECHNOLOGIES – This chapter introduces about all knowledge theories and technologies used in this project.

CHAPTER 2: SYSTEM ANALYSIS AND DESIGN – This chapter covers the main features, software requirement specifications, and database design of the project.

CHAPTER 3: IMPLEMENTATION AND EVALUATION – This chapter shows the implementation of this project, including pictures of each main function.

CONCLUSION – The concluding section of the project simultaneously emphasizes the problem solved, as well as presenting issues still unresolved and provides recommendations and suggestions.

REFERENCES – A presentation about the detail of referenced information used.

5. Drawings, charts (specify the types and sizes of drawings): No drawings, no charts.

6. Supervisor(s): PhD. Pham Cong Thang.

7. Date of assignment: March 24th, 2025.

8. Date of completion: June 18th, 2025.

Da Nang, June 19th, 2025

Head of Division …….

Instructor

概要

課題名: TripWiseウェブサイト ー 旅のためのインテリジェントな目的地およびルート推薦

実施する学生 - 学生証番号 - クラス:

Pham Trung Hieu (ファム・チュン・ヒエウ) – 102210034 – 21TCLC_Nhat1

Nguyen Thanh Minh (グエン・タイン・ミン) – 102210067 – 21TCLC_DT1

要約:

ダナンのインテリジェント観光ウェブサイトは、コンテンツベースの推薦、最短ルート検索機能、AIチャットボットのサポートなどの先進機能を統合することで、旅行体験を革新することを目的としています。このシステムのアーキテクチャは、図に示されているように、フロントエンド、Dockerでコンテナ化されたバックエンド、そしてデータベースの3つの主要なコンポーネントで構成されています。

フロントエンドはHTTP/HTTPS経由でアクセス可能で、直感的なインターフェースを提供し、ゲストがホテル、レストラン、観光名所に関する詳細情報を他のユーザーのレビューとともに閲覧できるようになっています。これにより旅行者は、各場所について包括的な理解を得ることができ、コミュニティのフィードバックに基づいた意思決定が可能となります。

本アプリケーションの特筆すべき機能のひとつは、パーソナライズされた旅行プラン作成機能です。ユーザーは、訪れたい場所やサービスを手動で追加し、希望の順番に並べて1日分の旅程を作成することができます。あるいは、バックエンドとTCP経由で連携する最短ルート検索機能を利用して、最も効率的な移動順序を計算し、旅程を自動最適化することも可能です。この機能により、ユーザーはより便利に、スムーズかつ楽しくダナンを探索することができます。

バックエンドはDocker環境でホストされており、システムの計算リソースを効率的に管理し、スケーラビリティを確保しています。

また、システムの重要な側面として、データベースとバックエンドの統合による管理者の運用機能が挙げられます。管理者は、古くなった情報や不

要なレビューを編集・削除することで、コンテンツの正確性と信頼性を維持できます。

フロントエンド、Docker化されたバックエンド、データベースの間に明確な通信経路を備えたこのモジュール型アーキテクチャは、堅牢なパフォーマンスと容易な保守性を実現し、ダナンにおける高度な観光体験を提供するための強固な基盤となっています。

SUMMARY

Project name: TripWise Website - Intelligent Destination and Route Recommendations for Journey.

Student name - Student ID - Class:

Pham Trung Hieu - 102210034 - 21TCLC_Nhat1

Nguyen Thanh Minh - 102210067 - 21TCLC_DT1

Summary:

The intelligent tourism website for Da Nang is designed to revolutionize travel experiences by integrating advanced features such as content-based recommendations, a shortest route finder, and AI chatbot support. This system architecture, as depicted, comprises three key components: the frontend, a Docker containerized backend, and a database. The frontend, accessible via HTTP/HTTPS, provides an intuitive interface where guests can explore detailed information about hotels, restaurants, and attractions, including reviews from other users. This allows travelers to gain a comprehensive understanding of each location and make informed decisions based on community feedback.

A standout feature of the application is its personalized trip planning capability. Users can manually curate a single-day itinerary by adding places or services, arranging them in a preferred order for easy tracking. Alternatively, the shortest route finder leverages the backend, communicated via TCP, to optimize the itinerary by calculating the most efficient travel sequence. This functionality enhances user convenience, ensuring a seamless and enjoyable exploration of Da Nang's offerings. The backend, hosted in a Docker environment, ensures scalability and efficient management of the system's computational resources.

Administrative oversight is a critical aspect of the system, enabled through the database and backend integration. Admins can efficiently manage content by editing or deleting outdated or irrelevant information and reviews. This maintains the accuracy and reliability of the data presented to users. The architecture's modular design, with clear communication channels between the frontend, Dockerized backend, and database, supports robust performance and easy maintenance, making it a solid foundation for delivering an enhanced tourism experience in Da Nang.

PREFACE

Today marks the culmination of over three months of dedicated work on my senior project, and I am deeply grateful to everyone who has supported me throughout this journey.

First and foremost, I express my profound gratitude to my supervisor, **PhD. Pham Cong Thang**, for his unwavering guidance, encouragement, and expertise. Despite his demanding schedule, his insightful feedback and mentorship have been pivotal in navigating the complexities of this research, shaping my understanding, and clarifying intricate concepts.

I am equally thankful to my family and friends, whose steadfast love, encouragement, and belief in me have been my pillars of strength. Their support has been essential in overcoming challenges and sustaining my motivation throughout this academic endeavor.

My sincere appreciation extends to my classmates and peers for their constructive feedback and camaraderie. Their shared insights and encouragement have significantly enriched this project and inspired me to strive for excellence.

I also acknowledge the academic staff and faculty members whose dedication to teaching and mentorship equipped me with the foundational knowledge and skills necessary for this project's success. Their contributions have been instrumental to my academic and personal growth.

To all who have played a role in this journey, your generosity and guidance have made my senior year a rewarding and transformative experience. Thank you.

Sincerely,

Pham Trung Hieu Nguyen Thanh Minh

ASSURANCE

I hereby guarantee that:

- The contents of this senior project are performed by me following the guidance of instructor **PhD. Pham Cong Thang**.
- All references used in this senior project thesis, are quoted with the author's name, project name, time and location to publish clearly and faithfully.
- All invalid copies educated statue violation or cheating will be borne the full responsibility by myself.

Performed Student 1	Performed Student 2
Pham Trung Hieu	Nguyen Thanh Minh

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

No.	Item	Description
1	AI	Artificial Intelligence
2	API	Application Programming Interface
3	HTTP	Hypertext Transfer Protocol
4	JSON	JavaScript Object Notation
5	LLM	Large Language Model
6	RAG	Retrieval-Augmented Generation
7	REST	Representational State Transfer
8	UI	User Interface
9	URL	Uniform Resource Locator
10	UX	User Experience

INTRODUCTION

Da Nang, Vietnam's third-largest city, presents unique opportunities and challenges for AI-powered tourism recommendations. As a rapidly developing coastal destination, Da Nang offers diverse attractions spanning beaches, cultural sites, culinary experiences, and natural landmarks like the Marble Mountains and Ba Na Hills. The city's compact geography and well-developed transportation infrastructure make it ideal for day-trip optimization. However, tourists often miss hidden gems or fail to efficiently navigate between distant attractions like the city center, beachfront areas, and mountainous regions. The seasonal variations in weather, local festivals, and peak tourist periods add complexity to trip planning that traditional static recommendations cannot address effectively.

The AI-powered tourism website focuses specifically on maximizing visitor experiences within Da Nang city boundaries through:

- **Localized personalization:** Recommend Da Nang-specific attractions, local restaurants, beaches, markets, and cultural sites based on user preferences, whether they seek adventure activities, cultural immersion, culinary exploration, or relaxation.
- **Intra-city route optimization:** Generate efficient daily itineraries that account for Da Nang's unique geography, connecting beach areas, city center attractions (Dragon Bridge, Han Market), cultural sites, and nearby natural attractions while minimizing travel time between locations.
- **Local context integration:** Incorporate Da Nang-specific factors such as traffic patterns, weather conditions affecting beach activities, tide schedules, local event calendars, and seasonal attraction availability to provide contextually relevant recommendations.
- **Cultural and culinary discovery:** Highlight Da Nang's distinctive food scene, local markets, and cultural experiences that tourists might overlook without local knowledge, ensuring visitors experience authentic aspects of the city beyond mainstream attractions.

This focused approach allows the AI system to develop deep expertise in Da Nang's tourism landscape, providing highly relevant and actionable recommendations that truly optimize the visitor experience within this specific urban environment.

TASK ASSIGNMENT

Student Name	Responsibilities
Pham Trung Hieu	1. Conduct research and define the project framework. 2. Design the user interface and user experience (UI/UX). 3. Process real-world location data in Da Nang. 4. Develop backend APIs for: User authentication, sending and processing requests using Gemini 5. Implement the designed UI/UX. 6. Build the AI assistant functionality. 7. Deploy the application using Render (backend) and Vercel (frontend).
Nguyen Thanh Minh	1. Conduct research and develop the project framework. 2. Create use-case diagrams to define system interactions. 3. Collect real-world location data in Da Nang. 4. Design comprehensive database schemas. 5. Implement the core backend APIs. 6. Develop an algorithm to optimize trip itineraries. 7. Build a recommendation system to enhance user experience.

CHAPTER 1: THEORIES AND TECHNOLOGIES

This system leverages contemporary, widely adopted technologies and platforms, integrated strategically to deliver a responsive and user-centric solution that effectively meets user needs and expectations.

To provide deeper insight into the technical foundation of this project, I will now introduce the core concepts behind the key technologies employed.

1.1. Retrieval-Augmented Generation (RAG)

At its core, Retrieval-Augmented Generation is a technique that combines the generative power of large language models with the vast and up-to-date information stored in external knowledge bases [1]. LLMs, despite their impressive capabilities, are inherently limited by the data they were trained on. This static training data can lead to several issues, including:

- **Knowledge gaps:** LLMs may lack information on events or developments that occurred after their training cutoff date [2].
- **Hallucinations:** In the absence of information, LLMs can generate plausible but incorrect or nonsensical statements.
- **Lack of specificity:** For domain-specific queries, the general knowledge of an LLM may not be sufficient to provide a detailed and accurate response [3].

RAG addresses these challenges by enabling the LLM to retrieve relevant information from a designated data source before generating a response [4]. This process grounds the model's output in factual, real-time information, significantly improving its quality and trustworthiness [5].

1.1.1. The core components

The RAG process can be broken down into three fundamental stages:

- **Indexing and data storage:** The external knowledge, which can range from a curated set of documents to a massive database or the entire web, is first processed and stored in a searchable format [6]. This typically involves chunking the data into smaller, manageable pieces and converting them into numerical representations called vector embeddings using an embedding model [7]. These embeddings are then stored in a specialized vector database, which allows for efficient similarity searches [8].
- **Retrieval:** When a user submits a query, it is also converted into a vector embedding [9]. The retrieval component of the RAG system then searches the

vector database to find the chunks of information whose embeddings are most like the query embedding. This ensures that the retrieved information is semantically relevant to the user's prompt.

- **Augmentation and generation:** The retrieved information is then concatenated with the original user prompt and fed into the large language model [10]. This augmented prompt provides the LLM with the necessary context to generate a response that is not only fluent and coherent but also factually grounded in the retrieved data [11]. The final output can also include citations or links to the source documents, allowing for verification and building user trust [12].

The conceptual flow of using RAG with LLMs is described in Figure 1.1.

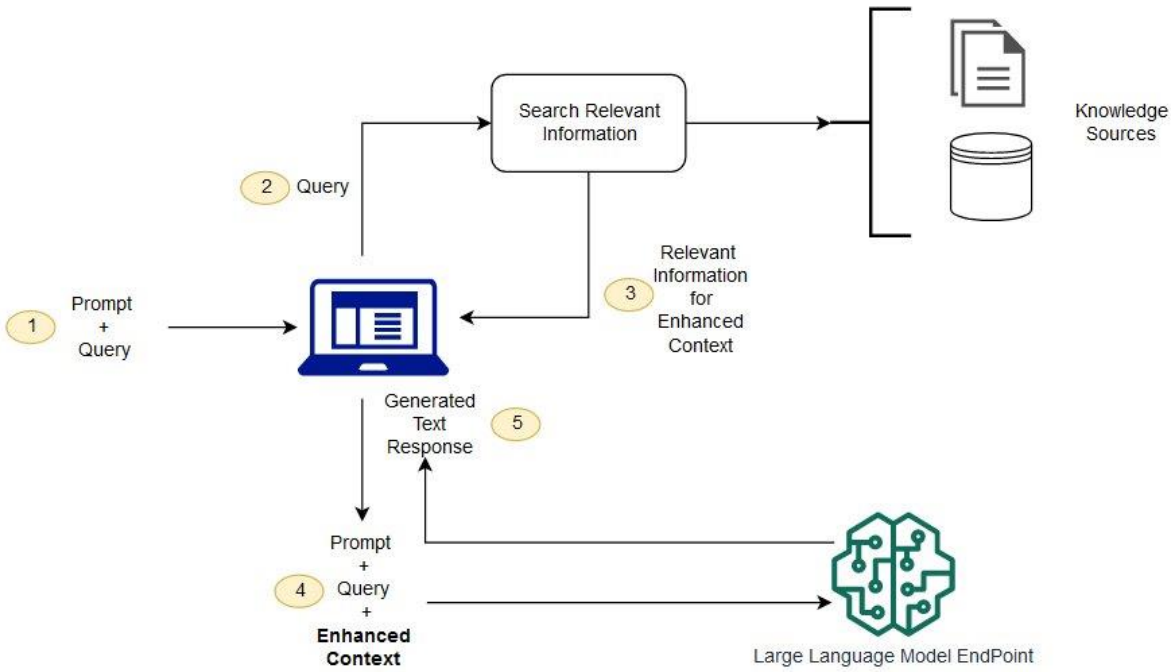

Figure 1.1 The conceptual flow of using RAG with LLMs

1.1.2. The advantages of implementing RAG

The adoption of RAG offers a multitude of benefits, making it a compelling approach for a wide array of applications:

- **Access to current information:** RAG models can provide up-to-date responses by drawing from constantly refreshed knowledge sources [13].
- **Improved factual accuracy:** By grounding responses in external data, RAG significantly reduces the likelihood of hallucinations and factual errors [14].
- **Enhanced user trust:** The ability to cite sources allows users to verify the information, fostering greater confidence in the AI system [15].

- **Cost-effectiveness:** RAG can be a more economical alternative to fine-tuning an LLM for specific domains, as it doesn't require retraining the entire model [16].
- **Increased transparency:** The retrieval step makes it easier to understand why the model generated a particular response.

1.1.3. Limitations and challenges

Despite its advantages, RAG is not without its limitations:

- **Dependency on retrieval quality:** The performance of a RAG system is heavily reliant on the quality of the retrieved information. If the retriever fails to find relevant documents, the generator's output will be suboptimal. This is often referred to as the "garbage in, garbage out" problem.
- **Latency:** The retrieval process adds an extra step to the generation pipeline, which can increase the overall response time.
- **Context integration:** Seamlessly integrating the retrieved information into the generated text without making it sound disjointed can be challenging.
- **Potential for bias:** The biases present in the external knowledge source can be propagated into the LLM's responses [17].

1.2. Gemini Function Calling

The Gemini function calling system enables a LLM to interact with external tools, APIs, or functions in a deterministic, structured way. It parses natural language inputs, identifies actionable intents, and produces structured outputs (typically JSON) to call pre-registered functions with the correct arguments. This augments the LLM's capabilities with real-time or specialized functionality. The recommend function calling workflow is describe in Figure 1.3.

The logo of Gemini is expressed in Figure 1.2.

Figure 1.2 Gemini

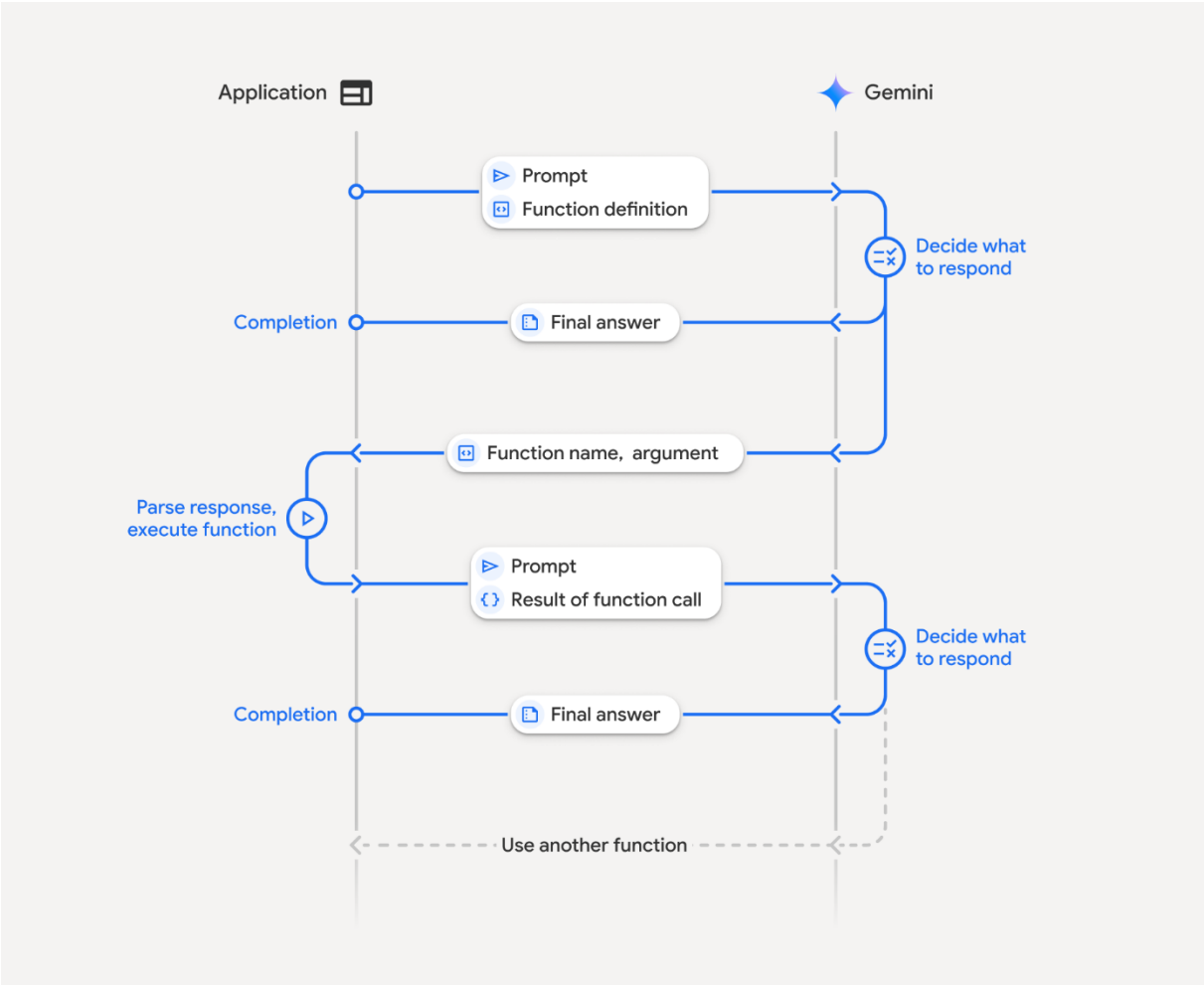

Figure 1.3 Function calling workflow

1.2.1. Architecture

Table 1.1 Gemini Function Calling - Components

Component	**Description**
LLM Core	Gemini model that interprets user input and produces structured function calls.
Function Registry	A list of available functions with metadata, signatures, and constraints.
Function Dispatcher	Middleware that executes the selected function with arguments and returns output to the model.
Validator/Parser	Ensures argument types and required parameters match schema.
Context Manager	Tracks ongoing tasks and memory of prior function calls (for chaining).

1.2.2. Capabilities

- Multi-step tool use: Supports chaining function calls in multi-turn conversations.
- Error handling: Handles invalid arguments or failed executions with fallback behaviors.
- Streaming or Async Support: Compatible with asynchronous APIs or streaming outputs.

1.2.3. Security & Sandbox

- Functions run in isolated environments to prevent side effects.
- Rate limits and scopes restrict API usage.
- All outputs are validated before being passed back to the model.

1.3. Content-based filtering

Content-based filtering is a recommendation system technique that relies on information about the features of items and the preferences of users.

1.3.1. Conceptual foundation

At its core, content-based filtering recommends items to a user based on the similarity between the item's content and the user's preferences. It operates under the assumption that if a user liked item A, they will also like items like A.

Key Assumptions:

- Users have consistent preferences over time.
- Items can be described by a set of features.
- A user's preference profile can be constructed based on rated items.

1.3.2. Representations

a. Item representation

Each item is described by a set of features. This representation can be formalized as a feature vector:

$$x_i = [x_{i1}, x_{i2}, \dots, x_{in}]$$

Where x_i represents the i-th item and x_{ij} is the value of the j-th feature for item i.

b. User profile representation

The user profile is a vector that captures a user's preferences across item features:

$$u_k = [u_{k1}, u_{k2}, \dots, u_{kn}]$$

Where u_{kj} epresents user k's affinity for feature j.

1.3.3. Learning user preferences

User profiles are learned by aggregating the features of items the user has interacted with. A common approach for constructing user profiles is to compute the weighted average of item features, as shown in equation (1-1):

$$u_k = \frac{1}{|I_k|} \sum_{i \in I_k} r_{ki} x_i \tag{1-1}$$

- I_k: set of items rated by user k
- r_{ki}: rating given by user k to item i
- x_i: feature vector of item i

This is essentially a weighted average of item features based on user ratings.

1.3.4. Prediction function

To recommend new items, the system estimates the similarity between the user profile and the feature vector of candidate items. This is typically done using a similarity function. Two common approaches are cosine similarity, as shown in equation (1-2), and dot product (linear model), as presented in equation (1-3):

- Cosine Similarity:

$$sim(u_k, x_j) = \frac{u_k x_j}{\|u_k\| \|x_j\|} \tag{1-2}$$

- Dot Product (Linear Model):

$$\hat{r}_{kj} = u_k x_j \tag{1-3}$$

$\hat{r}_{kj}$ represents the predicted rating or preference of user k for item j.

1.3.5. Advantages and limitations

Advantages:

- No need for other users' data (solves cold start for users).
- Personalized recommendations.
- Interpretable (features influence recommendations directly).

Limitations:

- Cold start for new items (requires item feature data).
- Limited diversity (over-specialization).
- Feature engineering is often required.

1.4. Dynamic programming

Coined by Richard Bellman (1950s) [18], the Principle of Optimality states: "An optimal solution to a problem contains optimal solutions to its subproblems" [19]. This recursive property is essential. It allows a complex problem to be decomposed into simpler, overlapping subproblems whose solutions can be reused.

1.4.1. Overlapping subproblems

Dynamic programming is suitable when the problem has overlapping subproblems - the same subproblems are solved multiple times during recursion.

- Unlike divide and conquer (e.g., merge sort), where subproblems are independent,
- In DP, solving each subproblem only once and storing its result (memoization or tabulation) yields efficiency.

1.4.2. Memoization and tabulation

These are two core techniques used in DP:

- **Memoization (Top-down):** Cache the results of expensive function calls and reuse them.
- **Tabulation (Bottom-up):** Build a table in a bottom-up manner, solving small subproblems first.

Both approaches trade time for space to avoid redundant computation.

1.4.3. Recurrence relations

DP problems are formally expressed through recurrence relations.

Example (Fibonacci sequence):

$F(n) = F(n-1) + F(n-2), with\ base\ case\ F(0) = 0, F(1) = 1$

This equation defines the solution in terms of smaller subproblem solutions [20].

1.4.4. State definition

A problem is formulated in terms of a state, typically defined by a set of parameters that fully characterize a subproblem.

Example: In the knapsack problem, a state could be defined as (i, w) = optimal value considering first i items and total weight capacity w.

The transition between states is defined by the recurrence and the decisions made (e.g., to include or exclude an item).

1.4.5. Time and space complexity

DP reduces exponential time complexity (common in brute-force recursive approaches) to polynomial time, due to its caching mechanism.

- If there are n subproblems and each takes $O(1)$ time, total time is $O(n)$.
- Space can often be optimized further with techniques like space reduction (e.g., using only the last row of a DP table).

1.4.6. Mathematical foundations

- **Inductive reasoning:** DP solutions often follow mathematical induction, solving base cases and building up.
- **Discrete optimization:** DP is widely used in operations research to solve linear and integer programs under discrete constraints.
- **Markov Decision Processes (MDPs):** In stochastic settings, DP forms the basis of algorithms like value iteration and policy iteration.

1.5. Google OR-Tools

Google OR-Tools (Operations Research Tools) is an open-source software suite developed by Google for solving combinatorial optimization problems. It provides a robust set of tools for tackling a wide range of optimization tasks such as routing, scheduling, integer programming, constraint programming, and linear programming.

The logo of Google OR-Tools is expressed in Figure 1.4.

Figure 1.4 Google OR-Tools

Key Features:

- **Routing solver:** Solves vehicle routing problems (VRP), traveling salesman problems (TSP), and their many variants with constraints like time windows, capacities, and more.
- **Constraint programming (CP-SAT):** A powerful solver for constraint satisfaction and optimization problems, based on a state-of-the-art SAT solver.
- **Linear and mixed-integer programming:** Interfaces with multiple solvers (e.g., SCIP, Gurobi, GLPK, CBC) for linear and mixed-integer problems.
- **Scheduling tools:** Handles job-shop, flow-shop, and resource allocation problems efficiently.

- **Multi-language support:** Available in C++, Python, Java, and .NET, making it accessible across platforms.
- **Highly scalable:** Designed to handle large, complex problems efficiently using modern optimization algorithms.

Advantages:

- Developed and maintained by Google, ensuring ongoing support and innovation.
- Free and open source under the Apache 2.0 license
- Rich documentation and active user community.

1.6. Flask

The logo of Flask is expressed in Figure 1.5.

Figure 1.5 Flask

Flask is a lightweight and flexible web framework written in Python. It is classified as a microframework because it does not require tools or libraries, allowing developers to add only the components they need.

Key Features:

- **Simplicity & flexibility:** Minimal boilerplate with a clean and readable syntax.
- **Modular design:** Developers can use extensions to add features like form validation, authentication, database ORM, etc.
- **Built-in development server & debugger:** Helpful for local testing and debugging.
- **RESTful request dispatching:** Makes it easy to build REST APIs.
- **Jinja2 templating:** Powerful templating engine for rendering HTML.
- **WSGI compliant:** Works with any WSGI-compliant server.

1.7. Next.js

The logo of Next.js is expressed in Figure 1.6.

Figure 1.6 Next.js

Next.js is a powerful React framework that enables server-side rendering (SSR), static site generation (SSG), and client-side rendering (CSR) in a unified setup. When combined with TypeScript, it brings type safety and enhanced developer experience to modern web development.

Key Features:

- **Built-in TypeScript support:** Next.js has first-class TypeScript support. When a tsconfig.json file is detected, it automatically configures and compiles TypeScript out of the box.
- **Automatic type generation:** TypeScript types are inferred and generated for routes, APIs, pages, and props, enhancing autocompletion and refactoring.
- **Flexible data fetching:** Use getStaticProps, getServerSideProps, and getInitialProps with full TypeScript support to type responses and improve predictability.
- **Type-safe API routes:** Next.js API routes written in TypeScript can leverage typed request and response objects using interfaces from libraries like NextApiRequest and NextApiResponse.
- **Project structure:** Integrates seamlessly with Next.js conventions (pages/, app/, components/) while allowing modular type definitions.
- **Tooling integration:** Works well with ESLint, Prettier, and Jest for typed linting, formatting, and testing.

Benefits:

- Catch errors at compile-time
- Autocomplete and IntelliSense in editors
- Easier code maintenance and refactoring
- Better documentation through types

1.8. Redis

The logo of Redis is expressed in Figure 1.7.

Figure 1.7 Redis

Redis (Remote Dictionary Server) is an open-source, in-memory data structure store used as a database, cache, message broker, and streaming engine. Designed for high performance, low latency, and versatility, Redis supports a wide range of data structures such as strings, hashes, lists, sets, sorted sets, bitmaps, hyperloglogs, and geospatial indexes.

Key Features:

- **In-memory storage:** Offers extremely fast read and write operations by storing data entirely in memory.
- **Persistence options:** Supports snapshotting (RDB) and append-only file (AOF) persistence to disk.
- **High availability:** Provides replication, automatic failover, and partitioning (sharding) through Redis Sentinel and Redis Cluster.
- **Pub/sub messaging:** Enables lightweight message broadcasting with publish/subscribe channels.
- **Lua scripting:** Supports server-side scripting with Lua for atomic and complex operations.
- **Extensibility:** Modules can extend Redis with new data types and commands.
- **Client support:** Compatible with most major programming languages, including Python, Java, Node.js, Go, and C#.

Redis is widely adopted in modern architecture due to its simplicity, speed, and support for advanced data structures and distributed system capabilities.

1.9. PostgreSQL

PostgreSQL is a powerful, open-source object-relational database management system (ORDBMS) that emphasizes extensibility and SQL compliance. Originally developed at the University of California, Berkeley, it has evolved into a robust, enterprise-grade database used widely across industries.

The logo of PostgreSQL is expressed in Figure 1.8.

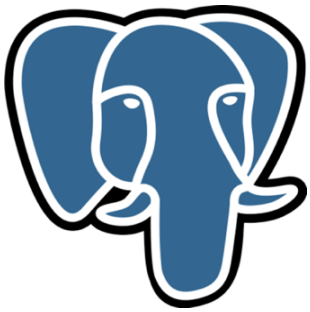

Figure 1.8 PostgreSQL

Key Features:

- **ACID compliance:** PostgreSQL ensures atomicity, consistency, isolation, and durability, making it suitable for critical transactional systems.
- **Advanced SQL support:** It supports a wide subset of the SQL standard, including complex queries, foreign keys, triggers, views, stored procedures, and window functions.
- **Extensibility:** PostgreSQL allows users to define their own data types, operators, functions, and even index types. It supports custom extensions and procedural languages such as PL/pgSQL, PL/Python, and PL/Perl.
- **Concurrency control:** PostgreSQL uses Multi-Version Concurrency Control (MVCC) to handle high levels of concurrent transactions without performance degradation.
- **Data integrity and reliability:** Features like point-in-time recovery, write-ahead logging (WAL), and full backup/restore capabilities contribute to data safety and recovery.
- **JSON and XML support:** PostgreSQL supports semi-structured data formats like JSON, JSONB (binary JSON), and XML, making it suitable for hybrid data models.
- **Replication and clustering:** Includes support for streaming replication, logical replication, and high-availability configurations with tools like Patroni and pgpool-II.
- **Cross-platform support:** Available on various operating systems including Linux, Windows, and macOS.
- **Community and ecosystem:** Maintained by an active global community, with a rich ecosystem of tools such as pgAdmin (GUI), PostGIS (spatial data), and many third-party integrations.

1.10. Neo4j

The logo of Neo4j is expressed in Figure 1.9.

Figure 1.9 Neo4j

Neo4j is a highly performant, open-source graph database management system designed to efficiently store, manage, and query data structured as graphs. Unlike traditional relational databases, which use tables and joins, Neo4j uses a property graph model—comprising nodes, relationships, and properties—to represent and query complex, highly interconnected data.

Key Features:

- **Native graph storage & processing:** Optimized for graph data from the ground up, enabling fast traversals and deep relationship analysis.
- **Cypher query language:** A declarative, SQL-like language tailored for working with graph structures, making it easy to express complex queries.
- **ACID compliance:** Ensures data integrity and reliability with support for atomicity, consistency, isolation, and durability.
- **Scalability & performance:** Supports large-scale graphs and real-time querying, making it suitable for enterprise-grade applications.
- **Visualization tools:** Offers a built-in browser and integration with tools like Bloom for visual graph exploration.

Neo4j integrates with major programming languages (Java, Python, JavaScript, etc.), big data platforms (Apache Spark, Kafka), and various APIs and frameworks, making it adaptable for diverse development environments.

CHAPTER 2: SYSTEM ANALYSIS AND DESIGN

2.1. Descriptive overview of objects

The system has three main actors:

- **Guest:** A user who accesses the website to explore detailed information about hotels, restaurants, and other attractions. Guests can view lists, details, and user reviews but cannot post or edit content.
- **User:** A guest who registers and is approved by the system becomes a user. Users can perform all guest functions, such as viewing lists, details, and reviews of hotels, restaurants, and attractions. Additionally, users can interact with trip features to create and manage trips, add places to their itineraries, optimize trip routes, save and manage favorite places, post and edit reviews, and manage their account settings.
- **Administrator:** An individual responsible for managing the system to ensure a user-friendly environment. Administrators can perform all user functions and have access to a dedicated dashboard providing a comprehensive overview of the system.

2.2. Use-case diagrams

2.2.1. Guest

The use-case diagram for guest is described in Figure 2.1.

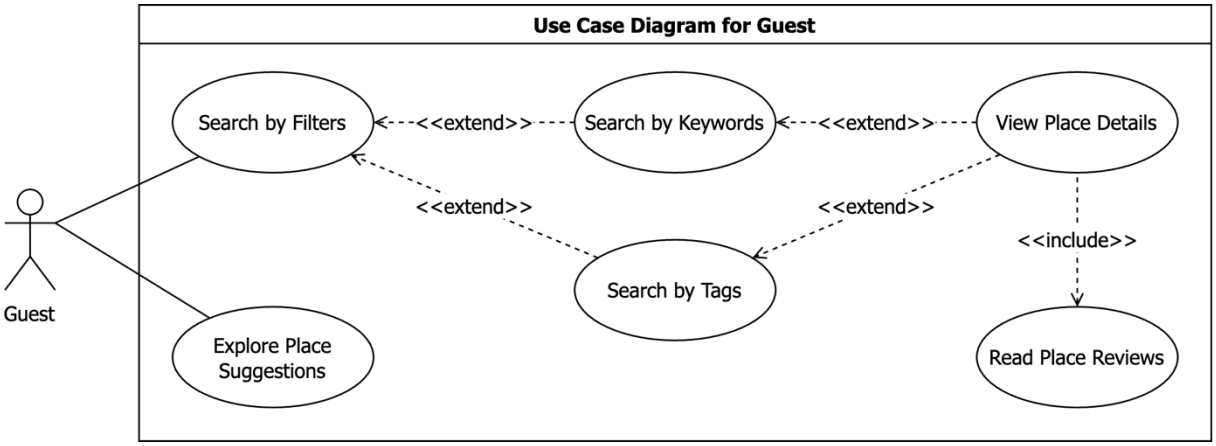

Figure 2.1 Use-case diagram for Guest

Table 2.1 Use-case Specification - Search by filters

Use Case ID	UC-1
Use Case Name	Search by filters
Description	As a guest/user/administrator, I want to apply filters to search for places, so that I can find places that match my specific preferences and criteria.
Actor(s)	Guest, User, Administrator
Trigger	The guest, user, or administrator seeks to apply filters to identify locations that align with their specific preferences.
Pre-Condition(s)	The actor has successfully accessed the website.
Post-Condition(s)	The system adapts to display a list of places tailored to the actor's preferences.
Basic Flow	1. The actor clicks the "Filter" button. 2. The system presents a list of filter options. 3. The actor selects desired filters and clicks "Save". 4. The system updates and displays a curated list of places matching the user's preferences.
Alternative Flow	4a. The system displays "No places found" if no places match the user's selected filters.
Exception Flow	4a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

Table 2.2 Use-case Specification - Search by keywords

Use Case ID	UC-2
Use Case Name	Search by keywords
Description	As a guest, user, or administrator, I want to search for places using keywords, so that I can discover places with names matching my search terms.
Actor(s)	Guest, User, Administrator
Trigger	The guest, user, or administrator seeks to apply filters to identify locations that align with their specific preferences.
Pre-Condition(s)	The actor has successfully accessed the website.
Post-Condition(s)	The system adapts to display a list of places tailored to the actor's preferences.
Basic Flow	1. The actor clicks the "Filter" button. 2. The system presents a list of filter options.

	3. The actor enters keywords to filter places by name and clicks "Save". 4. The system updates and displays a curated list of places matching the user's preferences.
Alternative Flow	4a. The system displays "No places found" if no places match the user's selected filters.
Exception Flow	4a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

Table 2.3 Use-case Specification - View place details

Use Case ID	UC-3
Use Case Name	View place details
Description	As a guest, user, or administrator, I want to view detailed information about places so that I can make informed decisions about my visit.
Actor(s)	Guest, User, Administrator
Trigger	The guest/user/administrator clicks on a place's card to navigate to its details page.
Pre-Condition(s)	The actor has successfully accessed the website.
Post-Condition(s)	The system displays the place's details page with relevant information.
Basic Flow	1. The actor clicks on a place's card. 2. The system displays the corresponding details page.
Alternative Flow	None
Exception Flow	2a. On request failure, the system displays an error toast with the server's message and returns to the previous page.
Non-Functional Requirement(s)	None

Table 2.4 Use-case Specification - Explore place suggestions

Use Case ID	UC-4
Use Case Name	Explore place suggestions
Description	As a guest, user, or administrator, I want to view place suggestions so that I can explore potential destinations when I haven't decided where to visit.
Actor(s)	Guest, User, Administrator

Trigger	None
Pre-Condition(s)	The actor has successfully accessed the website.
Post-Condition(s)	The system displays the place's details page with relevant information.
Basic Flow	None
Alternative Flow	None
Exception Flow	On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

Table 2.5 Use-case Specification - Read place reviews

Use Case ID	UC-5
Use Case Name	Read place reviews
Description	As a guest, user, or administrator, I want to read place reviews to understand others' experiences and opinions after visiting.
Actor(s)	Guest, User, Administrator
Trigger	The actor successfully accessed the place details information page.
Pre-Condition(s)	The actor has successfully accessed the website.
Post-Condition(s)	The system displays the place's details page with relevant information.
Basic Flow	None
Alternative Flow	None
Exception Flow	On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

2.2.2. User

The use-case diagram for user is described in Figure 2.2.

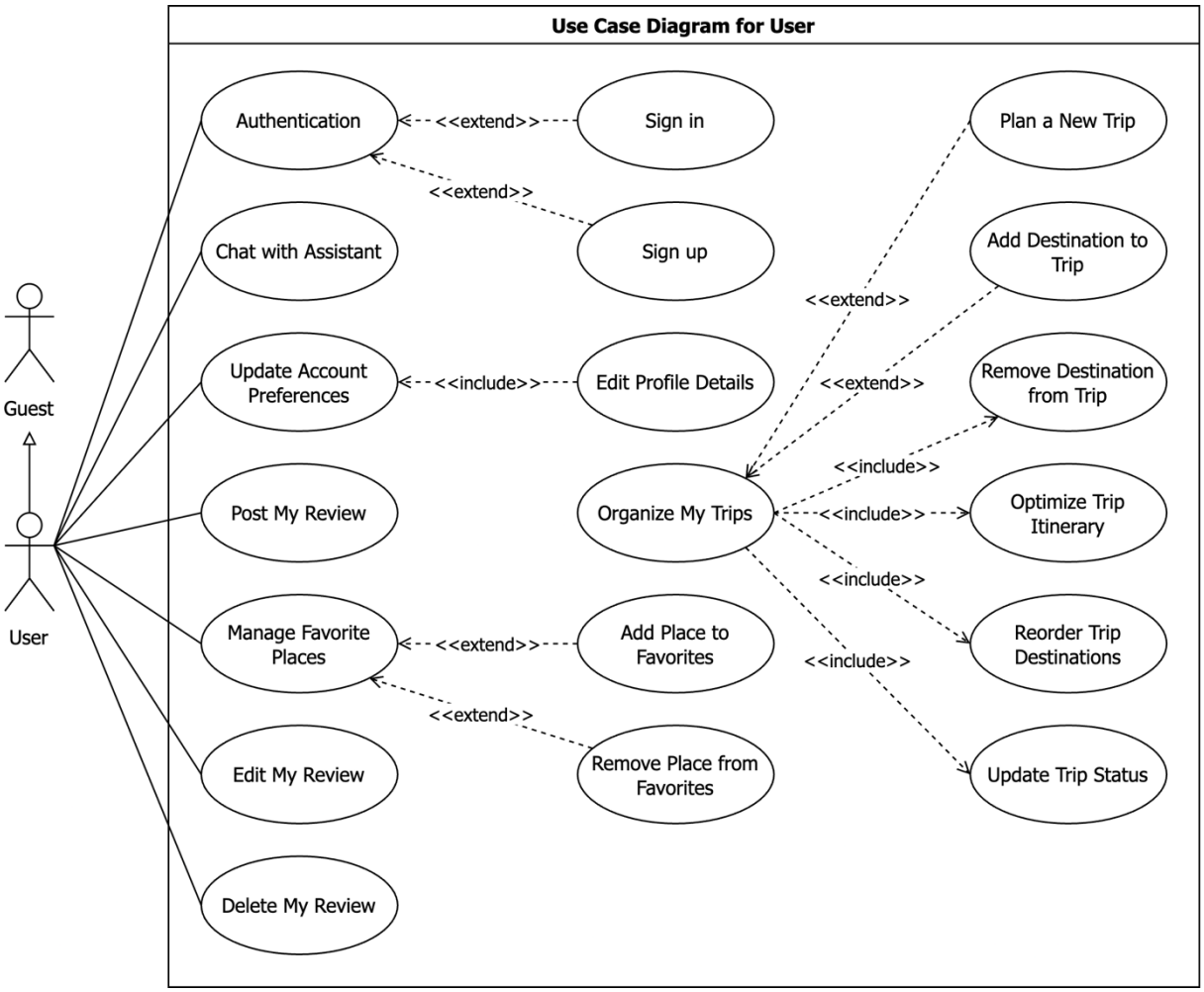

Figure 2.2 Use-case diagram for User

Table 2.6 Use-case Specification - Sign in

Use Case ID	UC-6
Use Case Name	Sign in
Description	As a user or administrator, I want to sign in to the system with my credentials, so that I can access the system with my assigned role and utilize role-specific features and functionalities.
Actor(s)	User, Administrator
Trigger	The actor clicks the “Sign in” button located in the website’s top navigation bar.
Pre-Condition(s)	The actor has successfully accessed the website.
Post-Condition(s)	The actor is authenticated and gains access to the system using their account.
Basic Flow	1. The actor enters his/her credentials and clicks the “Sign in” button. 2. The system displays a “Success” toast notification and redirects to the landing page.

Alternative Flow 1	1a. If the actor decides not to sign in, the actor clicks the "Back to Home" button. 1b. The system redirects to the landing page.
Alternative Flow 2	1a. If the actor does not have credentials, clicks the "Sign up" button located under the "Sign in" button. 1b. The system redirects the actor to the sign-up page.
Exception Flow 1	2a. If the actor provides invalid input, the system displays a red error message below each corresponding input field.
Exception Flow 2	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	- Email: Must be a valid email address format (e.g., user@example.com). - Password: Must be at least 8 characters long and include at least one letter and one number.

Table 2.7 Use-case Specification - Sign up

Use Case ID	UC-7
Use Case Name	Sign up
Description	As a user, I want to sign up to the system, so that I can create my own credentials to access the system and utilize role-specific features and functionalities.
Actor(s)	User
Trigger	The actor clicks the "Sign up" button located in the website's top navigation bar. The actor clicks the "Sign up" button located under the "Sign in" button in the sign in page.
Pre-Condition(s)	The actor has successfully accessed the website.
Post-Condition(s)	The actor creates credentials successfully. The actor can be authenticated and gains access to the system using their account.
Basic Flow	1. The actors enter his/her credential information, checks the box to accept the website's terms and policies, and then clicks the "Sign up" button. 2. The system displays a "Success" toast notification and redirect the user to the sign in page.
Alternative Flow 1	1a. If the actor decides not to sign up, the actor clicks the "Back to Home" button. 1b. The system redirects to the landing page.
Alternative Flow 2	1a. If the actor decides to sign in, clicks the "Sign in" button located under the "Sign up" button.

	1b. The system redirects the actor to the sign in page.
Exception Flow 1	2a. If the actor provides invalid input, the system displays a red error message below each corresponding input field.
Exception Flow 2	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	- Full name: Must be a string, more than a character. - Email: Must be a valid email address format (e.g., user@example.com). - Password: Must be at least 8 characters long and include at least one letter and one number.

Table 2.8 Use-case Specification - Chat with assistant

Use Case ID	UC-8
Use Case Name	Chat with assistant
Description	As a user or administrator, I want to chat with assistant, so that I can receive details information and suggestions related to tourism based on my specific requirements.
Actor(s)	User, Administrator
Trigger	The actor clicks the robot icon button located in the website's top navigation bar.
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system.
Post-Condition(s)	The actor receives details information and suggestions related to tourism.
Basic Flow	1. The user inputs a question or request for the AI assistant in the provided text field and submits it by pressing Enter or clicking the send icon. 2. The system processes the input and generates a relevant response for the user. 3. If the user seeks additional information, they repeat step 1 to continue the interaction.
Alternative Flow	2a. If the question is deemed irrelevant, the system responds with a polite refusal.
Exception Flow	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	Questions or requests must contain more than one character.

Table 2.9 Use-case Specification - Update account preferences

Use Case ID	UC-9
Use Case Name	Update account preferences
Description	As a user or administrator, I want the ability to update my account preferences, so that I can maintain accurate and up-to-date personal information.
Actor(s)	User, Administrator
Trigger	None
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system.
Post-Condition(s)	The actor's account can access their profile page.
Basic Flow	1. The actor selects the Profile option from the dropdown menu accessed by clicking the avatar icon. 2. The system displays the profile page.
Alternative Flow	None
Exception Flow	None
Non-Functional Requirement(s)	None

Table 2.10 Use-case Specification - Edit profile details

Use Case ID	UC-10
Use Case Name	Edit profile details
Description	As a user or administrator, I want the ability to edit my profile details, so that I can maintain accurate and up-to-date personal information.
Actor(s)	User, Administrator
Trigger	The actor selects the Profile option from the dropdown menu accessed by clicking the avatar icon.
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system.
Post-Condition(s)	The actor's account is updated with the latest information.
Basic Flow	1. The actors enter his/her latest personal information and then clicks the "Save" button. 2. The system displays a "Success" toast notification and redirect the user to the sign in page.
Alternative Flow	None
Exception Flow 1	2a. If the actor provides invalid input, the system displays a red error message below each corresponding input field.

Exception Flow 2	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	Questions or requests must contain more than one character.

Table 2.11 Use-case Specification - Organize my trips

Use Case ID	UC-11
Use Case Name	Organize my trips
Description	As a user or administrator, I want to organize my trips, so that I can create and access personal trips within the system.
Actor(s)	User, Administrator
Trigger	The actor clicks the avatar icon to open the dropdown menu, selects "Profile", then clicks the "Trips" tab on the left.
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system.
Post-Condition(s)	The actor's account can access their trips page.
Basic Flow	None
Alternative Flow	None
Exception Flow	None
Non-Functional Requirement(s)	None

Table 2.12 Use-case Specification - Plan a new trip

Use Case ID	UC-12
Use Case Name	Plan a new trip
Description	As a user or administrator, I want to plan a new trip, so that I can access personal trips within the system.
Actor(s)	User, Administrator
Trigger	The actor clicks the "Create New" button either on the trips page or within the trips card list.
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system.
Post-Condition(s)	The system successfully creates the actor's new trip.
Basic Flow	1. The actor enters a trip name into the text field and clicks the "Save" button. 2. The system saves the trip and displays a success toast notification confirming the action.

Alternative Flow	1a. If the actor chooses not to continue creating the trip, they can click outside the card to exit the create trip flow.
Exception Flow	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	The trip name must be a string containing at least one character.

Table 2.13 Use-case Specification - Add destination to trip

Use Case ID	UC-13
Use Case Name	Add destination to trip
Description	As a user or administrator, I want to add a destination to an existing trip so that it can be used in other system functions.
Actor(s)	User, Administrator
Trigger	The actor adds a place by clicking the add icon on the top-right of a place card, selecting the "Add to Trip" button, or clicking the "Add Place" button on the bottom-left of the trip details page.
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system.
Post-Condition(s)	The system successfully adds a place to an existing trip.
Basic Flow	1. The actor selects the name of the trip to which they want to add the place. 2. Once completed, the system processes the request and displays a success toast notification.
Alternative Flow	1a. If the actor decides not to add the place, they can cancel the action by clicking outside the card.
Exception Flow	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

Table 2.14 Use-case Specification - Remove destination from trip

Use Case ID	UC-14
Use Case Name	Remove destination from trip
Description	As a user or administrator, I want to remove a destination from an existing trip so that I can personalize the trip to my preferences.
Actor(s)	User, Administrator
Trigger	The actor successfully accesses the trip details page.

Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system.
Post-Condition(s)	The system successfully removes a place from an existing trip.
Basic Flow	1. The actor clicks the minus icon button next to the place in the trip list. 2. The system processes the request, displays a success toast notification, and updates the trip to reflect the removed place.
Alternative Flow	None
Exception Flow	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

Table 2.15 Use-case Specification - Optimize trip itinerary

Use Case ID	UC-15
Use Case Name	Optimize trip itinerary
Description	As a user or administrator, I want to optimize trip itinerary so that I can efficiently plan my travel.
Actor(s)	User, Administrator
Trigger	The actor successfully accesses the trip details page for their own trip.
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system.
Post-Condition(s)	The system successfully optimizes the routes for a trip.
Basic Flow	1. The actor clicks on the "Optimize Routes" button on the right, under the list of places in the current trip. 2. The systems optimizes the routes based on the first place that the actor has saved and displays the result.
Alternative Flow	None
Exception Flow	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

Table 2.16 Use-case Specification - Reorder trip destinations

Use Case ID	UC-16
Use Case Name	Reorder trip destinations
Description	As a user or administrator, I want to reorder destinations in a trip to customize my travel route.
Actor(s)	User, Administrator
Trigger	The actor successfully accesses the trip details page for their own trip.
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system.
Post-Condition(s)	The system successfully reorders places in a trip.
Basic Flow	1. The actor drags and drops the places to their preferred order. 2. The system updates and displays the new order and corresponding routes on the map. 3. The actor clicks the "Save Changes" button. 4. The system saves the updated order.
Alternative Flow	3a. To cancel the new changes, the actor clicks the refresh button, restoring the previous order.
Exception Flow	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

Table 2.17 Use-case Specification - Manage favorite places

Use Case ID	UC-17
Use Case Name	Manage favorite places
Description	As a user or administrator, I want to manage my favorite places so that I can personalize my experience.
Actor(s)	User, Administrator
Trigger	The actor successfully accesses profile page.
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system.
Post-Condition(s)	The system successfully displays the actor's favourite places.
Basic Flow	1. The actor selects the Profile option from the dropdown menu accessed by clicking the avatar icon. 2. The system displays the profile page, including the actor's favourite places.
Alternative Flow	None

Exception Flow	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

Table 2.18 Use-case Specification - Add place to favorites

Use Case ID	UC-18
Use Case Name	Add place to favorites
Description	As a user or administrator, I want to add a place to favorites so that I can personalize my experience.
Actor(s)	User, Administrator
Trigger	None
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system.
Post-Condition(s)	The system successfully adds a place to the actor's favorite places.
Basic Flow	1. The actor clicks the heart icon button or the "Save" button with a heart icon. 2. The system processes the request and displays a success toast notification.
Alternative Flow	None
Exception Flow	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

Table 2.19 Use-case Specification - Remove place from favorites

Use Case ID	UC-19
Use Case Name	Remove place from favorites
Description	As a user or administrator, I want to remove place from my favorites so that I can personalize my experience.
Actor(s)	User, Administrator
Trigger	None
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system.
Post-Condition(s)	The system successfully removes a place from the actor's favorite places.

Basic Flow	1. The actor clicks the heart icon button or the "Save" button with a heart icon. 2. The system processes the request and displays a success toast notification.
Alternative Flow	None
Exception Flow	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

Table 2.20 Use-case Specification - Post my review

Use Case ID	UC-20
Use Case Name	Post my review
Description	As a user or administrator, I want to post a review so that I can share my experience of the place I visited.
Actor(s)	User, Administrator
Trigger	None
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system.
Post-Condition(s)	The system successfully creates a review by the actor for the place.
Basic Flow	1. The user clicks the "Write a review" button in the Reviews section of a place's detail page. They then select a star rating and provide a written explanation for their choice, followed by clicking the "Post" button. 2. The system processes the request and displays a success toast notification.
Alternative Flow	None
Exception Flow	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

Table 2.21 Use-case Specification - Edit my review

Use Case ID	UC-21
Use Case Name	Edit my review
Description	As a user or administrator, I want to edit a review to clearly and concisely share my experience of the visited location.

Actor(s)	User, Administrator
Trigger	None
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system.
Post-Condition(s)	The system successfully edits a review by the actor for the place.
Basic Flow	1. The actor clicks the three-dot button next to their review and selects "Update". 2. The system displays a form allowing the actor to edit their review. 3. The actor updates the review and clicks "Submit". 4. The system processes the request and shows a success toast notification.
Alternative Flow	None
Exception Flow 1	2a. On request failure, the system displays an error toast with the server's message.
Exception Flow 2	4a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

Table 2.22 Use-case Specification - Delete my review

Use Case ID	UC-22
Use Case Name	Delete my review
Description	As a user or administrator, I want to post a review so that I can share my experience of the place I visited.
Actor(s)	User, Administrator
Trigger	None
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system.
Post-Condition(s)	The system successfully creates a review by the actor for the place.
Basic Flow	1. The user clicks the "Write a review" button in the Reviews section of a place's detail page. They then select a star rating and provide a written explanation for their choice, followed by clicking the "Post" button. 2. The system processes the request and displays a success toast notification.
Alternative Flow	None

Exception Flow	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

2.2.3. Administrator

The use-case diagram for administrator is described in Figure 2.3.

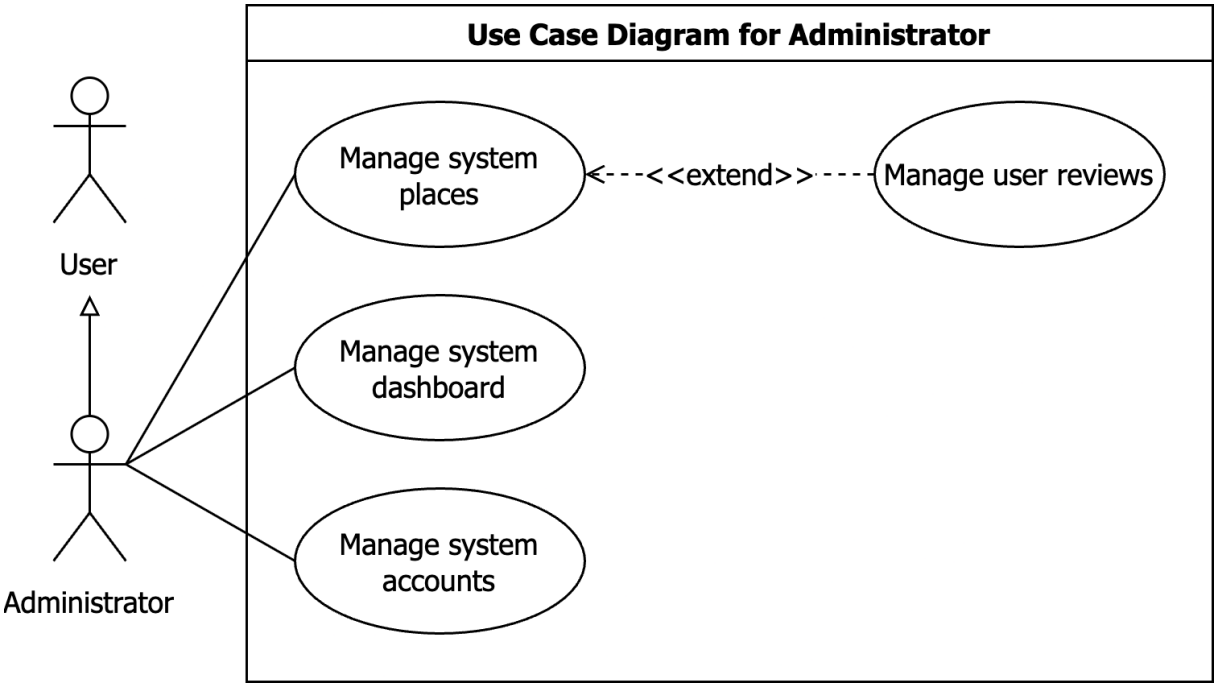

Figure 2.3 Use-case diagram for Administrator

Table 2.23 Use-case Specification - Manage system places

Use Case ID	UC-20
Use Case Name	Manage system places
Description	As an administrator, I want to manage system locations to ensure the database remains current and accurate.
Actor(s)	Administrator
Trigger	None
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system. The account is assigned the administrator role.
Post-Condition(s)	The system successfully processes the places management request.
Basic Flow	1. The actor clicks "Admin Management" from the dropdown menu accessed by clicking the avatar icon in the navigation bar. 2. The system displays the Admin Dashboard. 3. The actor clicks the management tab in the left-hand menu.

	4. The system displays the Places Management Dashboard.
Alternative Flow	None
Exception Flow	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

Table 2.24 Use-case Specification - Manage user reviews

Use Case ID	UC-21
Use Case Name	Manage user reviews
Description	As an administrator, I want to manage user reviews to help maintain a respectful and courteous environment within the system.
Actor(s)	Administrator
Trigger	None
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system. The account is assigned the administrator role.
Post-Condition(s)	The system successfully processes the users' reviews management request.
Basic Flow	1. The actor selects "Admin Management" from the dropdown menu accessed via the avatar icon in the navigation bar. 2. The system displays the Admin Dashboard. 3. The actor clicks the management tab in the left-hand menu. 4. The system displays the Places Management Dashboard. 5. The actor clicks the "View" button to select a specific place. 6. The system displays a details card containing information about the selected place. 7. The actor updates the reviews section with the desired changes.
Alternative Flow	6a. If the actor decides not to edit the current place, they click outside the card to deselect it and choose a different place.
Exception Flow	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

Table 2.25 Use-case Specification - Manage system dashboard

Use Case ID	UC-22
Use Case Name	Manage system dashboard
Description	As an administrator, I want to manage the system dashboard so that I can monitor system statistics and make informed decisions for the future.
Actor(s)	Administrator
Trigger	None
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system. The account is assigned the administrator role.
Post-Condition(s)	The system successfully displays the Admin Dashboard.
Basic Flow	1. The actor selects "Admin Management" from the dropdown menu accessed via the avatar icon in the navigation bar. 2. The system displays the Admin Dashboard.
Alternative Flow	None
Exception Flow	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

Table 2.26 Use-case Specification - Manage system accounts

Use Case ID	UC-23
Use Case Name	Manage system accounts
Description	As an administrator, I want to manage system accounts so that I can monitor system usage and assign administrative privileges to selected users.
Actor(s)	Administrator
Trigger	None
Pre-Condition(s)	The actor has successfully accessed the website. The actor has signed in to the system. The account is assigned the administrator role.
Post-Condition(s)	The system successfully processes the users' accounts management request.
Basic Flow	1. The actor selects "Admin Management" from the dropdown menu accessed via the avatar icon in the navigation bar. 2. The system displays the Admin Dashboard.

	3. The actor clicks the "Account Management" tab in the left-hand menu. 4. The system displays the Account Management Dashboard. 5. The actor clicks the "View" button to select a specific user account. 6. The system displays a details card containing information about the selected user account. 7. The actor updates the information with the desired changes.
Alternative Flow	6a. If the actor decides not to edit the current user account, they click outside the card to deselect it and choose a different user account.
Exception Flow	2a. On request failure, the system displays an error toast with the server's message.
Non-Functional Requirement(s)	None

2.3. Entity relational diagrams

The entity relational diagram for PostgreSQL is described in Figure 2.4.

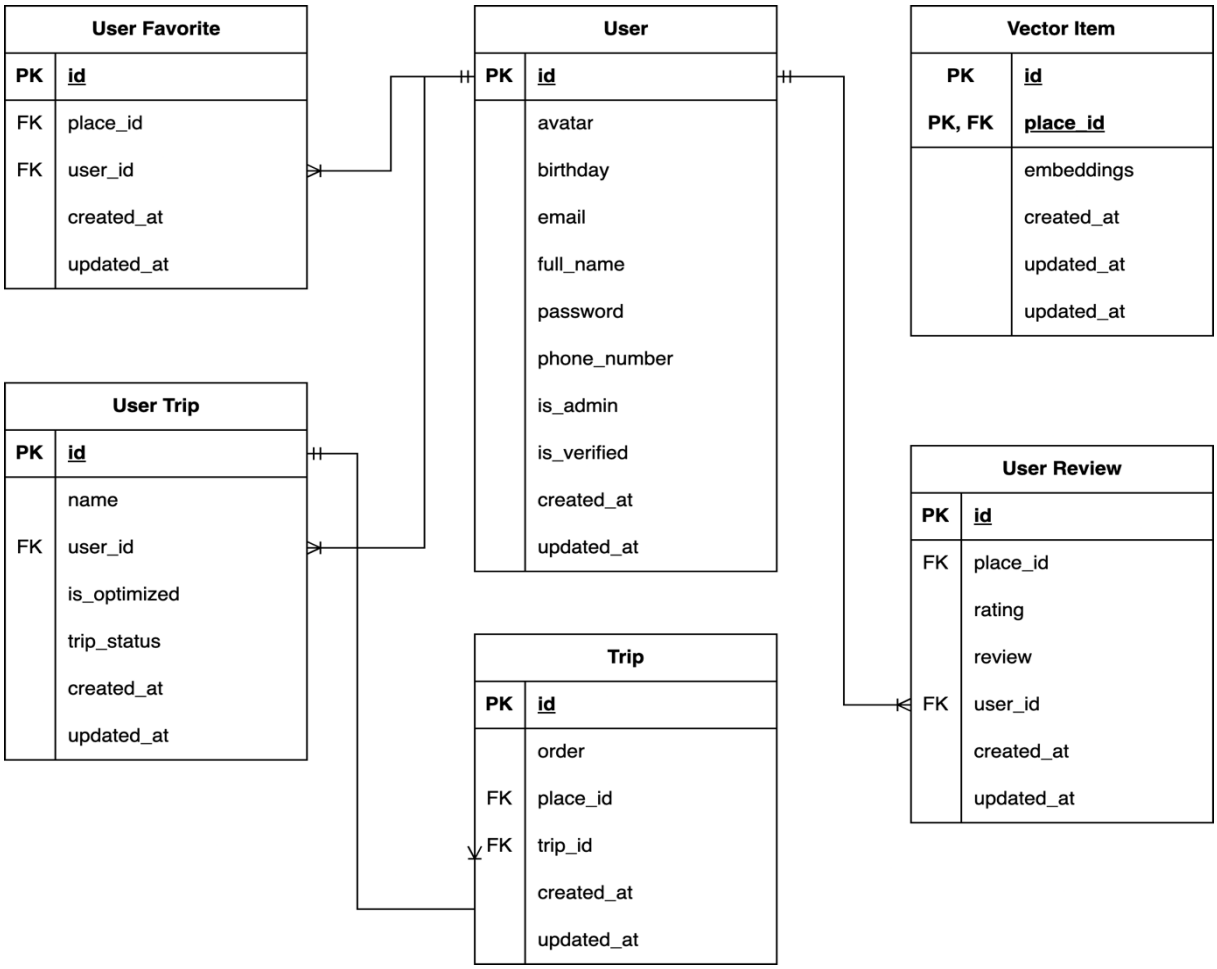

Figure 2.4 ERD - Relational database

The entity relational diagram for Neo4j is described in Figure 2.5.

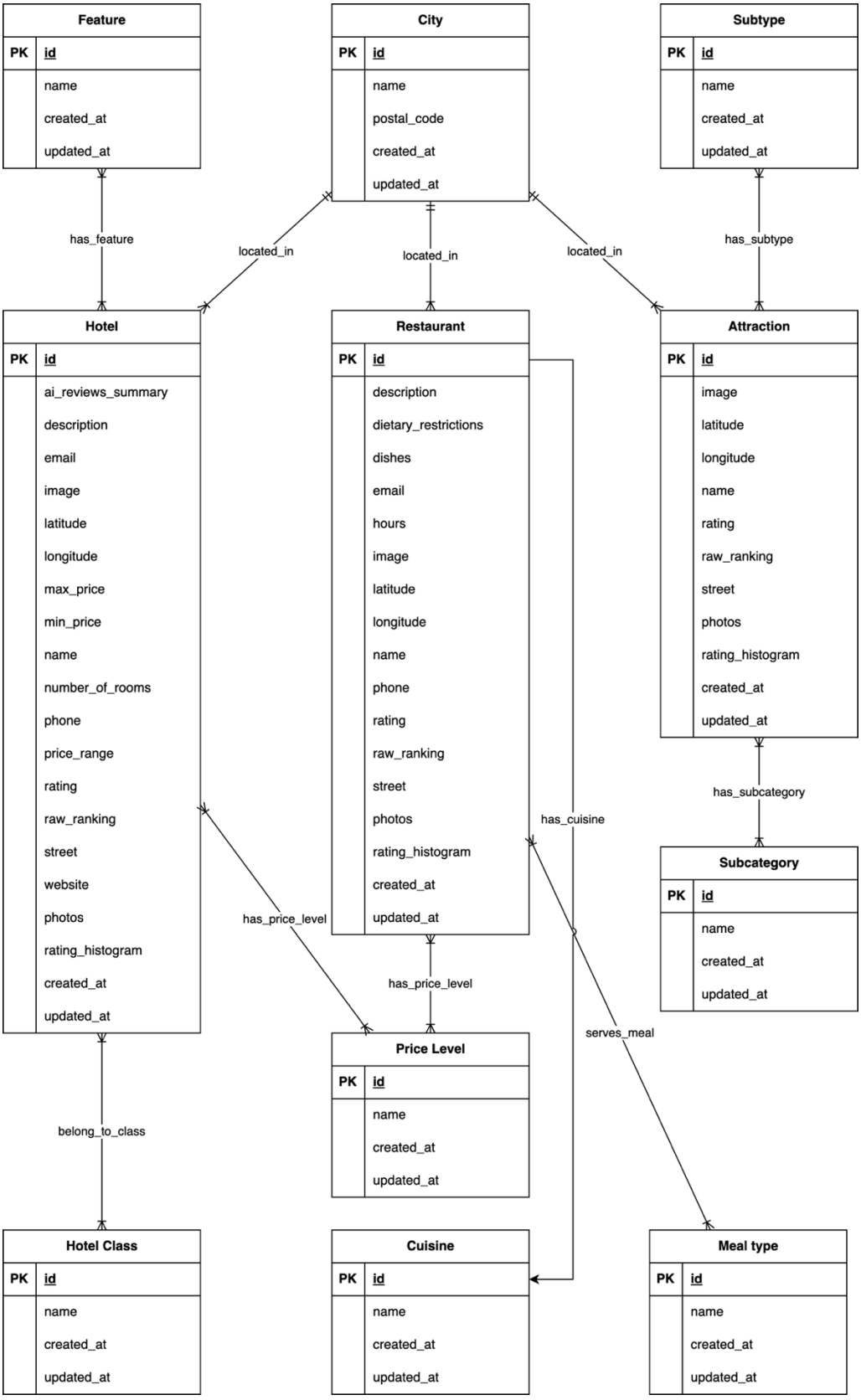

Figure 2.5 ERD - Graph database

Table 2.27 Database Specification - Users table

No.	Attribute	Data type	PK	NN	FK	Default
1	id	uuid	x	x		
2	avatar	varchar				
3	birthday	datetime				
4	email	varchar		x		
5	full_name	varchar		x		
6	password	varchar		x		
7	phone_number	varchar				
8	is_admin	boolean		x		false
9	is_verified	boolean		x		false
10	created_at	datetime		x		now()
11	updated_at	datetime		x		now()

Table 2.28 Database Specification - User favorites table

No.	Attribute	Data type	PK	NN	FK	Default
1	id	uuid	x	x		
2	place_id	varchar		x	hotels::id restaurants::id attractions::id	
3	user_id	uuid		x	users::id	
4	created_at	datetime		x		now()
5	updated_at	datetime		x		now()

Table 2.29 Database Specification - User trips table

No.	Attribute	Data type	PK	NN	FK	Default
1	id	uuid	x	x		
2	name	varchar		x		
3	user_id	uuid		x	users::id	
4	is_optimized	boolean		x		false
5	trip_status	boolean		x		false
6	created_at	datetime		x		now()
7	updated_at	datetime		x		now()

Table 2.30 Database Specification - Trips table

No.	Attribute	Data type	PK	NN	FK	Default
1	id	uuid	x	x		
2	order	integer		x		
3	place_id	varchar		x	hotels::id restaurants::id attractions::id	
4	trip_id	uuid		x	user_trips::id	
5	created_at	datetime		x		now()
6	updated_at	datetime		x		now()

Table 2.31 Database Specification - User reviews table

No.	Attribute	Data type	PK	NN	FK	Default
1	id	uuid	x	x		
2	place_id	varchar			hotels::id restaurants::id attractions::id	
3	rating	integer		x		
4	review	varchar		x		
5	user_id	uuid		x	users::id	
6	created_at	datetime		x		now()
7	updated_at	datetime		x		now()

Table 2.32 Database Specification - Vector places table

No.	Attribute	Data type	PK	NN	FK	Default
1	id	uuid	x	x		
2	embedding	jsonb		x		
3	place_id	varchar		x		
4	created_at	datetime		x		now()
5	updated_at	datetime		x		now()

Table 2.33 Database Specification - Cities node

No.	Attribute	Data type	PK	NN	FK	Default
1	id	uuid	x	x		
2	name	varchar		x		

3	postal_code	varchar		x		
4	created_at	datetime		x		now()
5	updated_at	datetime		x		now()

Table 2.34 Database Specification - Hotels node

No.	Attribute	Data type	PK	NN	FK	Default
1	id	uuid	x	x		
2	ai_reviews_summary	varchar				
3	description	varchar				
4	email	varchar				
5	image	varchar				
6	latitude	decimal		x		
7	longitude	decimal		x		
8	max_price	decimal				
9	min_price	decimal				
10	name	varchar		x		
11	number_of_rooms	integer				
12	phone	varchar				
14	price_range	varchar				
15	rating	decimal		x		
16	raw_ranking	decimal		x		
17	street	varchar		x		
18	website	varchar				
13	photos	jsonb				[]
19	rating_histogram	jsonb		x		[0, 0, 0, 0, 0]
20	created_at	datetime		x		now()
21	updated_at	datetime		x		now()

Table 2.35 Database Specification - Restaurants node

No.	Attribute	Data type	PK	NN	FK	Default
1	id	uuid	x	x		
2	description	varchar				

3	dietary_restrictions	jsonb				
4	dishes	jsonb				
5	email	varchar				
6	hours	json				
7	image	varchar				
8	latitude	decimal		x		
9	longitude	decimal		x		
10	name	varchar		x		
11	phone	varchar				
13	rating	decimal		x		
14	raw_ranking	decimal		x		
15	street	varchar		x		
12	photos	jsonb				[]
16	rating_histogram	jsonb				[0, 0, 0, 0, 0]
17	created_at	datetime		x		now()
18	updated_at	datetime		x		now()

Table 2.36 Database Specification - Attractions node

No.	**Attribute**	**Data type**	**PK**	**NN**	**FK**	**Default**
1	id	uuid	x	x		
2	image	varchar				
3	latitude	decimal		x		
4	longitude	decimal		x		
5	name	varchar		x		
6	rating	decimal		x		
7	raw_ranking	decimal		x		
8	street	varchar		x		
9	photos	jsonb				[]
10	rating_histogram	jsonb				[0, 0, 0, 0, 0]
11	created_at	datetime		x		now()
12	updated_at	datetime		x		now()

Table 2.37 Database Specification - Cuisines node

No.	Attribute	Data type	PK	NN	FK	Default
1	id	uuid	x	x		
2	name	varchar		x		
3	created_at	datetime		x		now()
4	updated_at	datetime		x		now()

Table 2.38 Database Specification - Features node

No.	Attribute	Data type	PK	NN	FK	Default
1	id	uuid	x	x		
2	name	varchar		x		
3	created_at	datetime		x		now()
4	updated_at	datetime		x		now()

Table 2.39 Database Specification - Hotel classes node

No.	Attribute	Data type	PK	NN	FK	Default
1	id	uuid	x	x		
2	name	varchar		x		
3	created_at	datetime		x		now()
4	updated_at	datetime		x		now()

Table 2.40 Database Specification - Meal types node

No.	Attribute	Data type	PK	NN	FK	Default
1	id	uuid	x	x		
2	name	varchar		x		
3	created_at	datetime		x		now()
4	updated_at	datetime		x		now()

Table 2.41 Database Specification - Price levels node

No.	Attribute	Data type	PK	NN	FK	Default
1	id	uuid	x	x		
2	name	varchar		x		
3	created_at	datetime		x		now()
4	updated_at	datetime		x		now()

Table 2.42 Database Specification - Subcategories node

No.	Attribute	Data type	PK	NN	FK	Default
1	id	uuid	x	x		
2	name	varchar		x		
3	created_at	datetime		x		now()
4	updated_at	datetime		x		now()

Table 2.43 Database Specification - Subtypes node

No.	Attribute	Data type	PK	NN	FK	Default
1	id	uuid	x	x		
2	name	varchar		x		
3	created_at	datetime		x		now()
4	updated_at	datetime		x		now()

2.4. Sequence diagrams

Figure 2.6 is a sequence diagram showing the process of handling a user-submitted question through the endpoint, where the server forwards the prompt to the Gemini service, optionally queries the database for additional system data, and returns a response to the UI - or an error message if something fails.

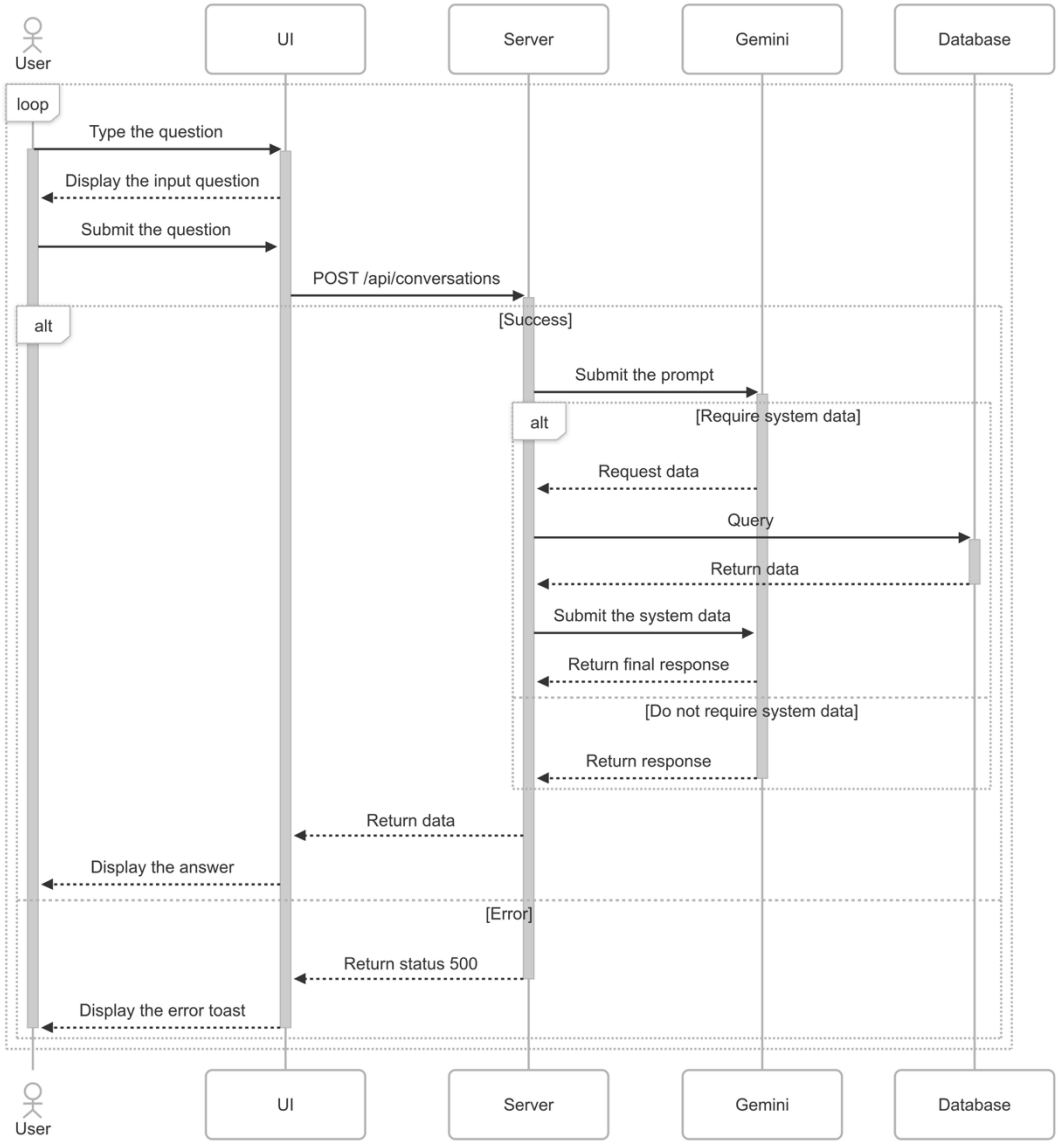

Figure 2.6 Sequence diagram - Chat with assistant

Figure 2.7 is a sequence diagram that depicts the parallel processing of multiple API requests, demonstrating how the server fetches or computes data using the cache or database as needed, and returns the results to the UI for display.

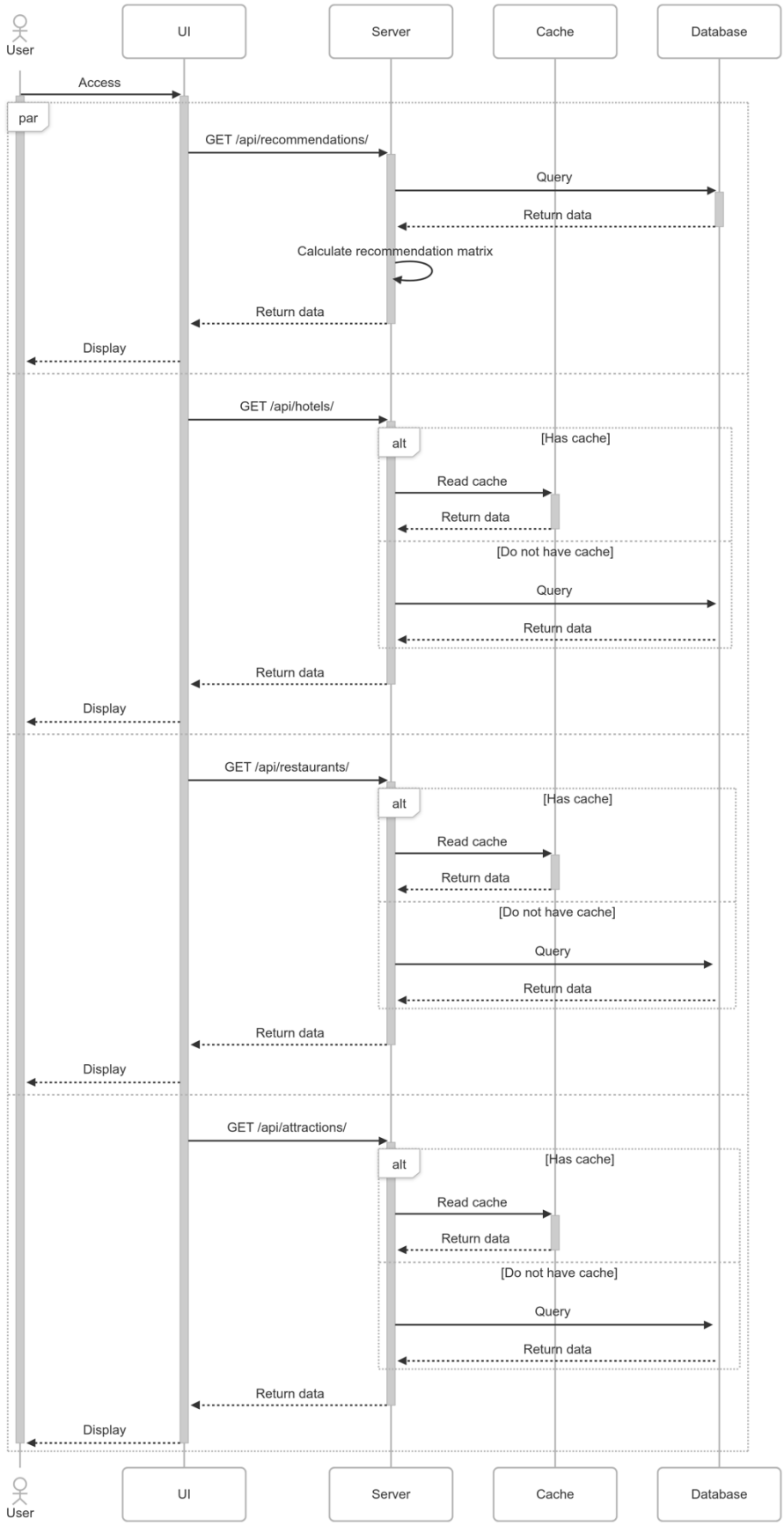

Figure 2.7 Sequence diagram - Actor accesses Home Screen

Figure 2.8 is a sequence diagram describing the process of adding a trip via a request, where the server validates the user session and place data, interacts with the database, manages cache invalidation, and returns appropriate status codes (200, 400, 403, or 500) that are reflected in the UI through corresponding success or error toasts.

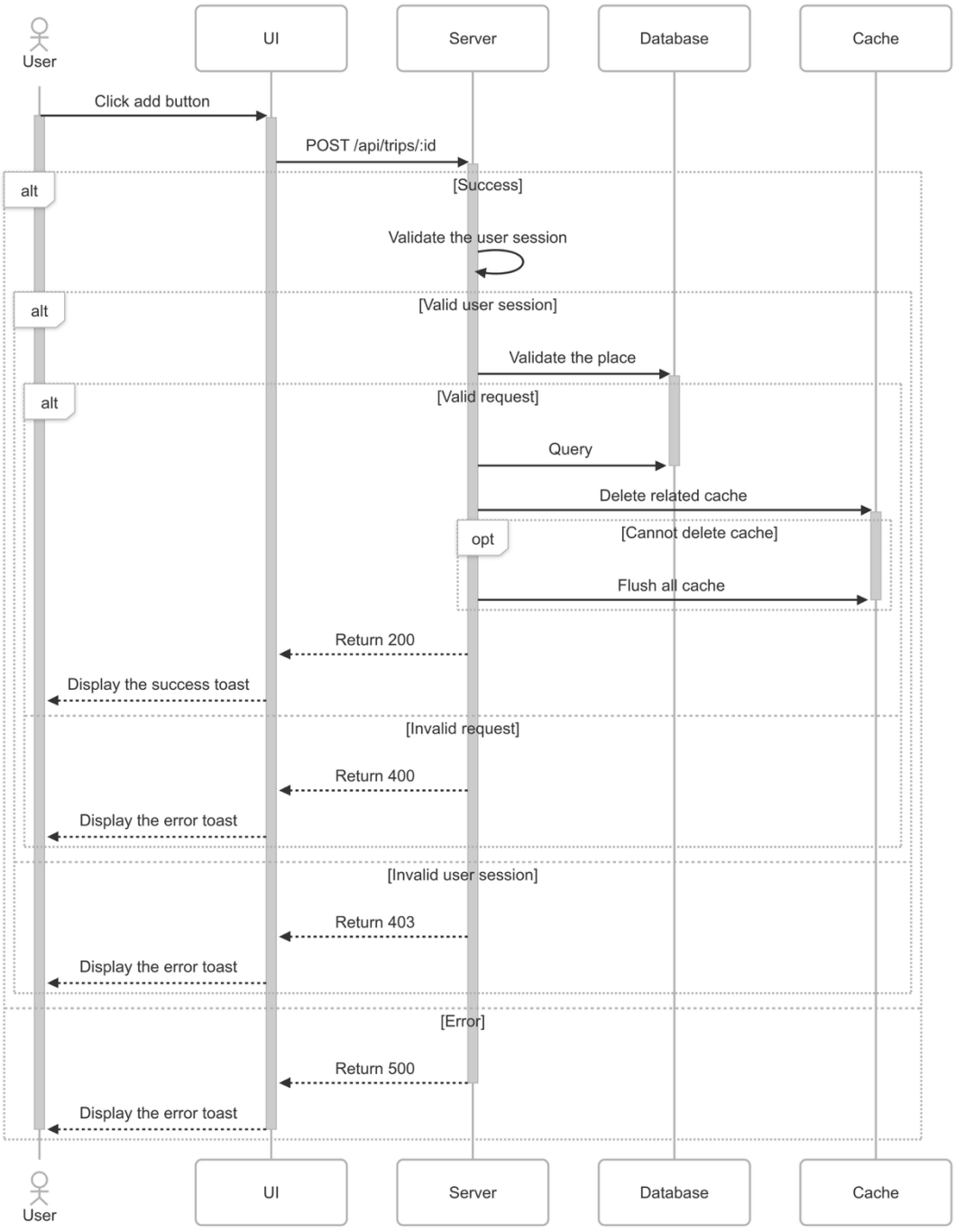

Figure 2.8 Sequence diagram - Add destination to trip

This image is a sequence diagram that illustrates the process flow when a user clicks the “love” button, triggering a request, including session validation, database

query, cache handling, and the corresponding success or error responses returned to the UI.

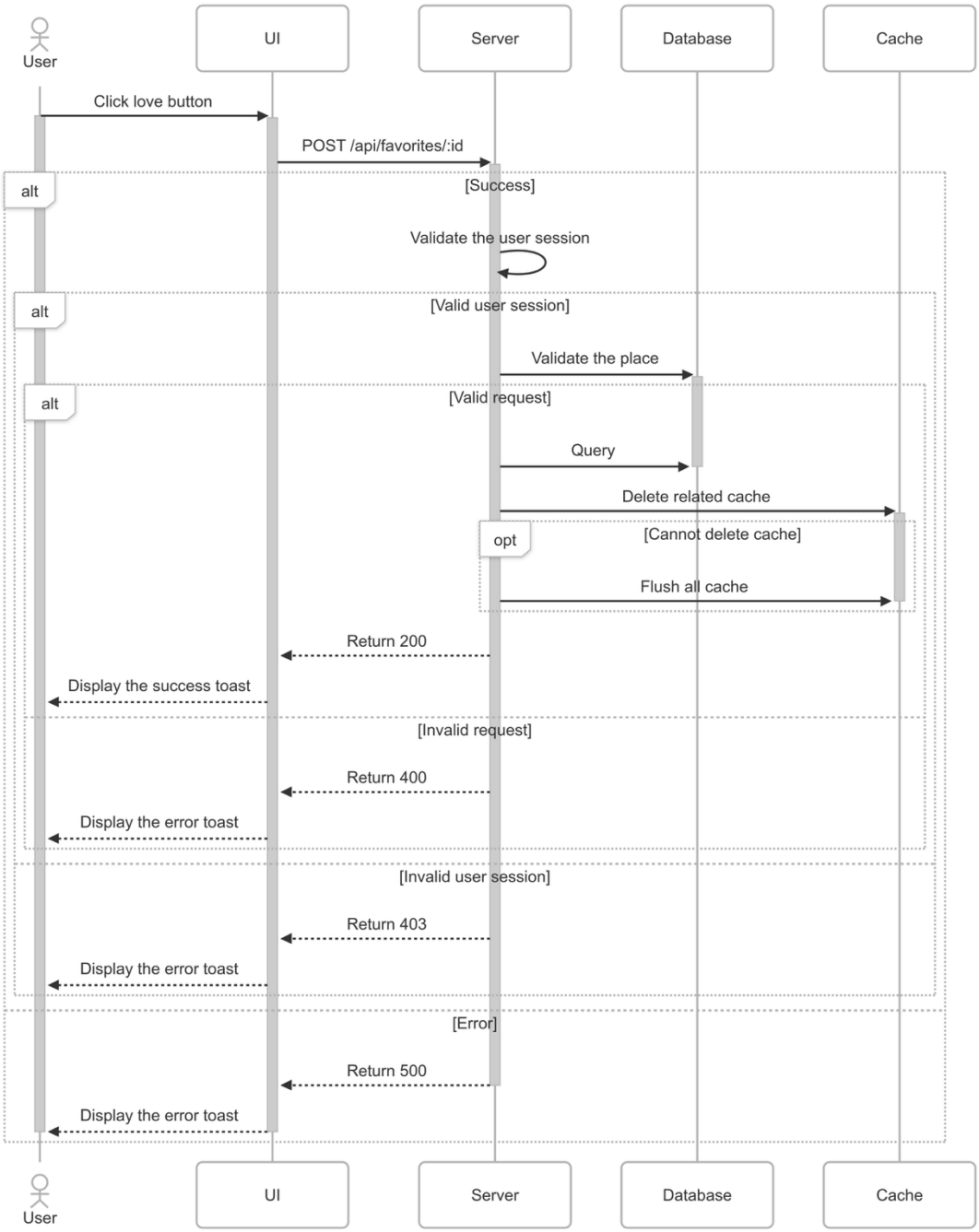

Figure 2.9 Sequence diagram - Add place to favorites

Figure 2.10 is a sequence diagram that depicts the workflow triggered when a user clicks “Save changes,” sending a `PATCH /api/trips/:id` request, followed by user session and trip validation, database operations, cache management, and various server responses that determine whether a success or error message is displayed in the UI.

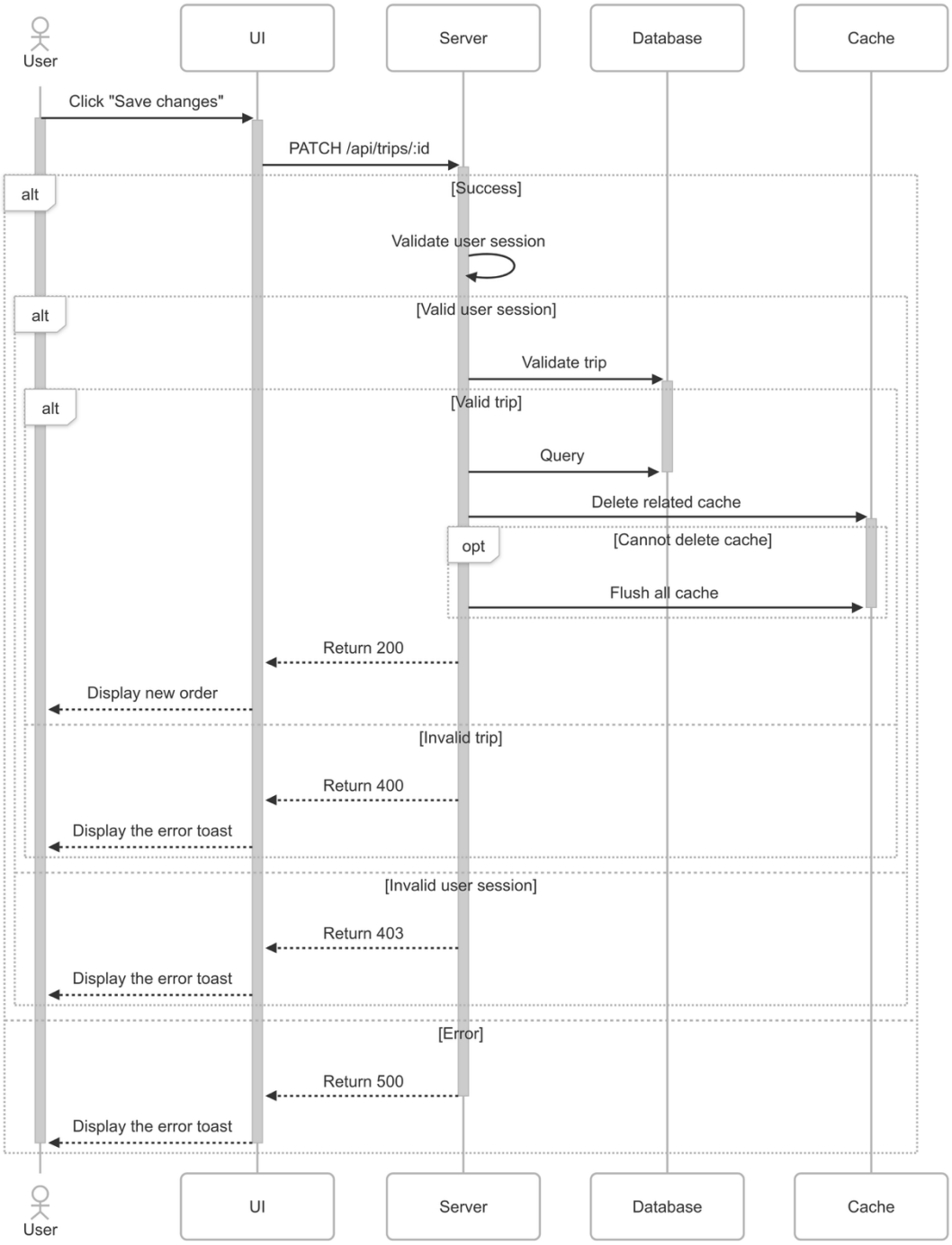

Figure 2.10 Sequence diagram - Reorder trip destinations

This image is a sequence diagram showing the process initiated when a user clicks “Create New,” sending a request, which includes validating the user session, querying the database, managing the cache, and returning the appropriate response (success or error) to update the UI accordingly.

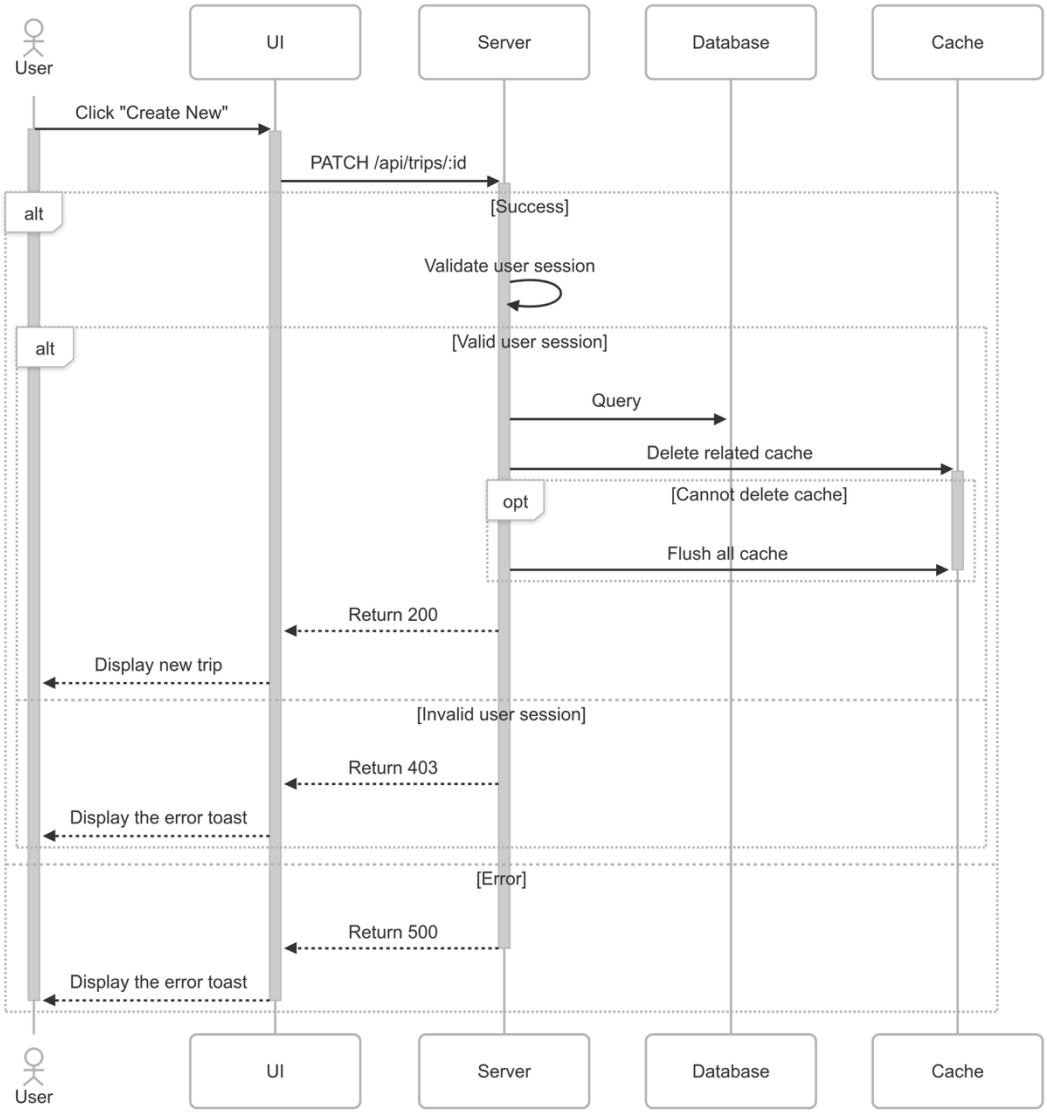

Figure 2.11 Sequence diagram - Plan a new trip

This image is a sequence diagram illustrating the process flow triggered when a user clicks “Submit”, issuing a request. It includes steps for validating the user session, querying the database, managing cache invalidation, and returning a success or error response that updates the UI accordingly.

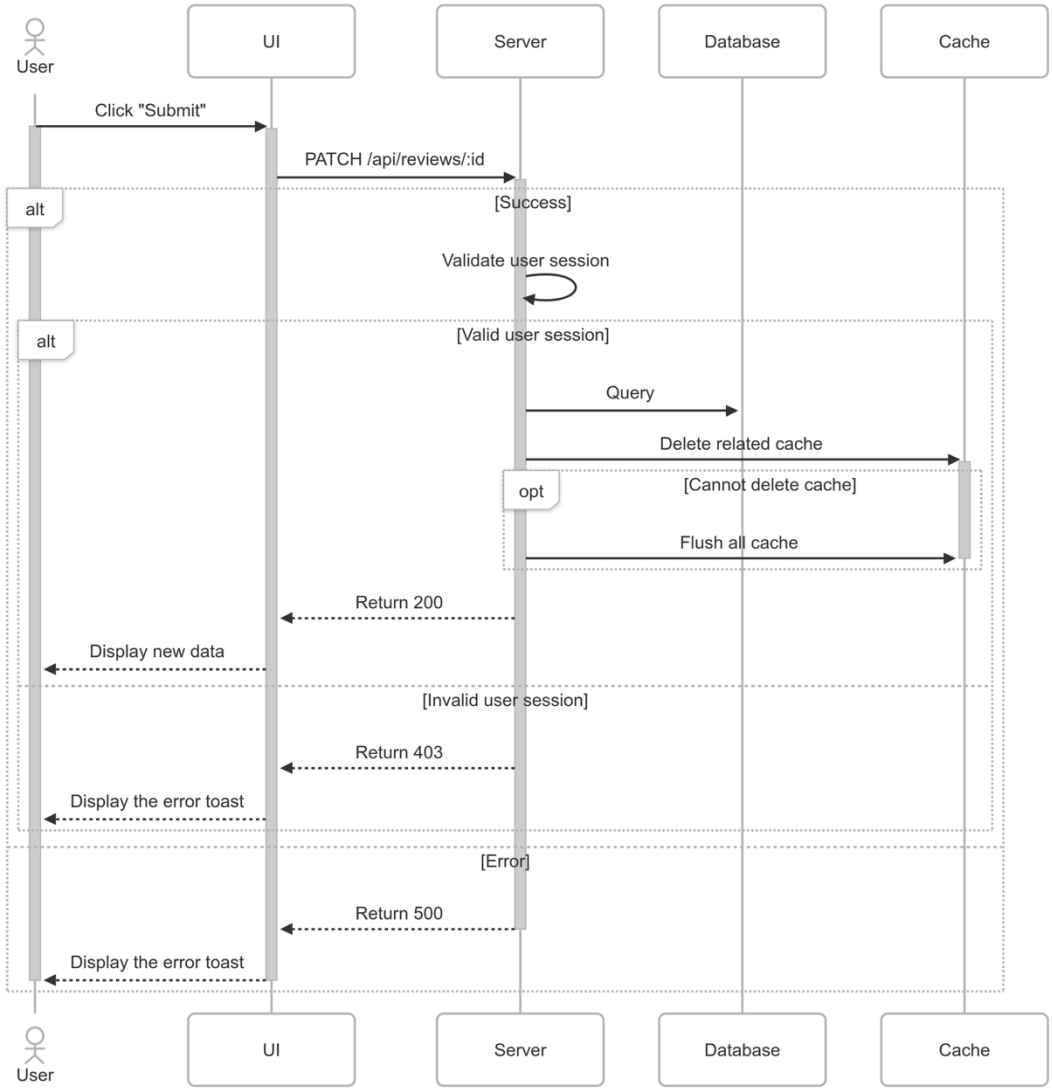

Figure 2.12 Sequence diagram - Edit my review

This image is a sequence diagram showing the process when a user accesses their favorites via a request. It includes validating the user session, attempting to retrieve data from the cache, querying the database if the cache misses, optionally saving the cache, and returning the data or appropriate error responses to the UI.

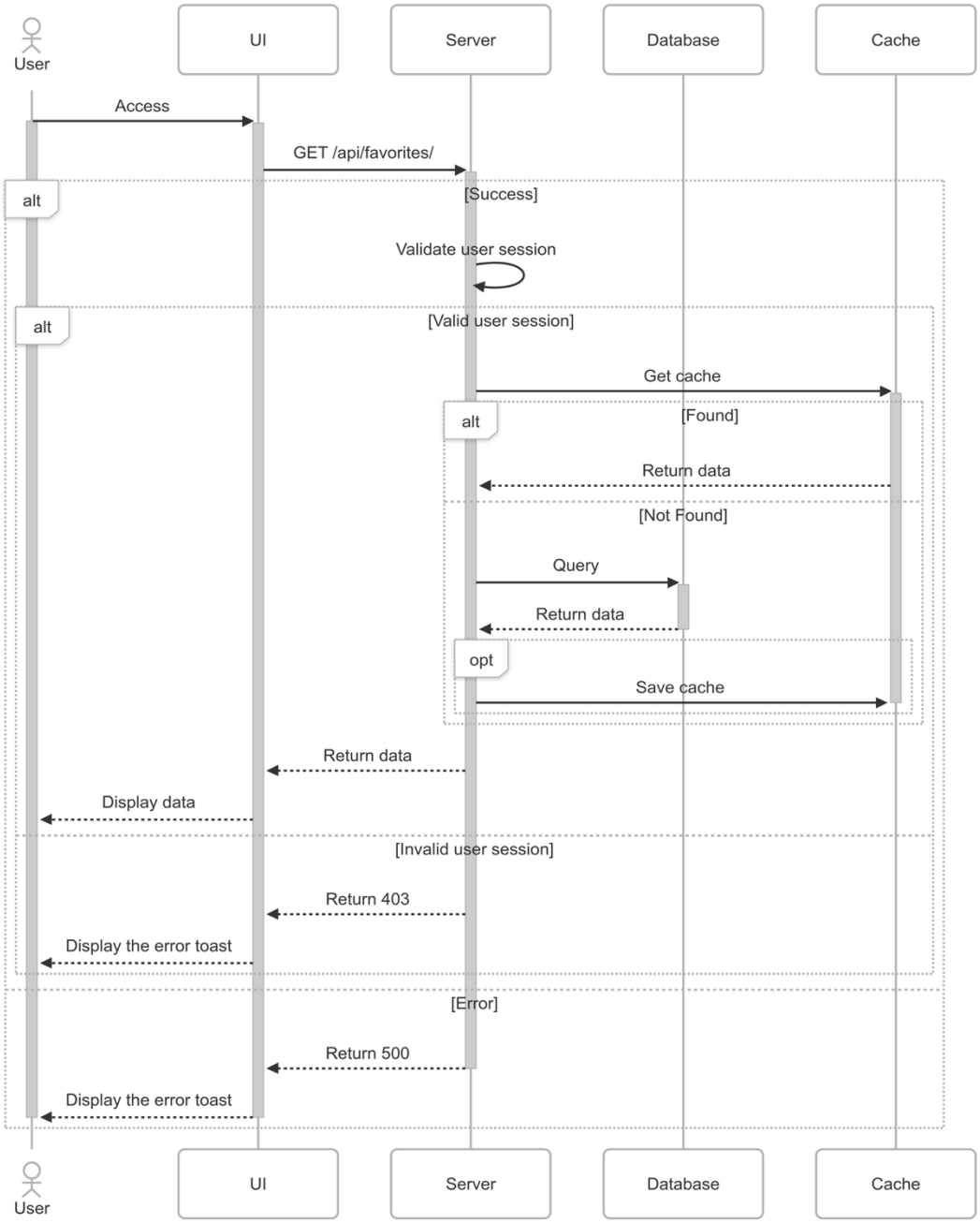

Figure 2.13 Sequence diagram - Get my favorite places

Figure 2.14 is a sequence diagram illustrating the process of a user accessing a trip API endpoint, including user session validation, cache retrieval, database querying, and error handling with appropriate HTTP responses.

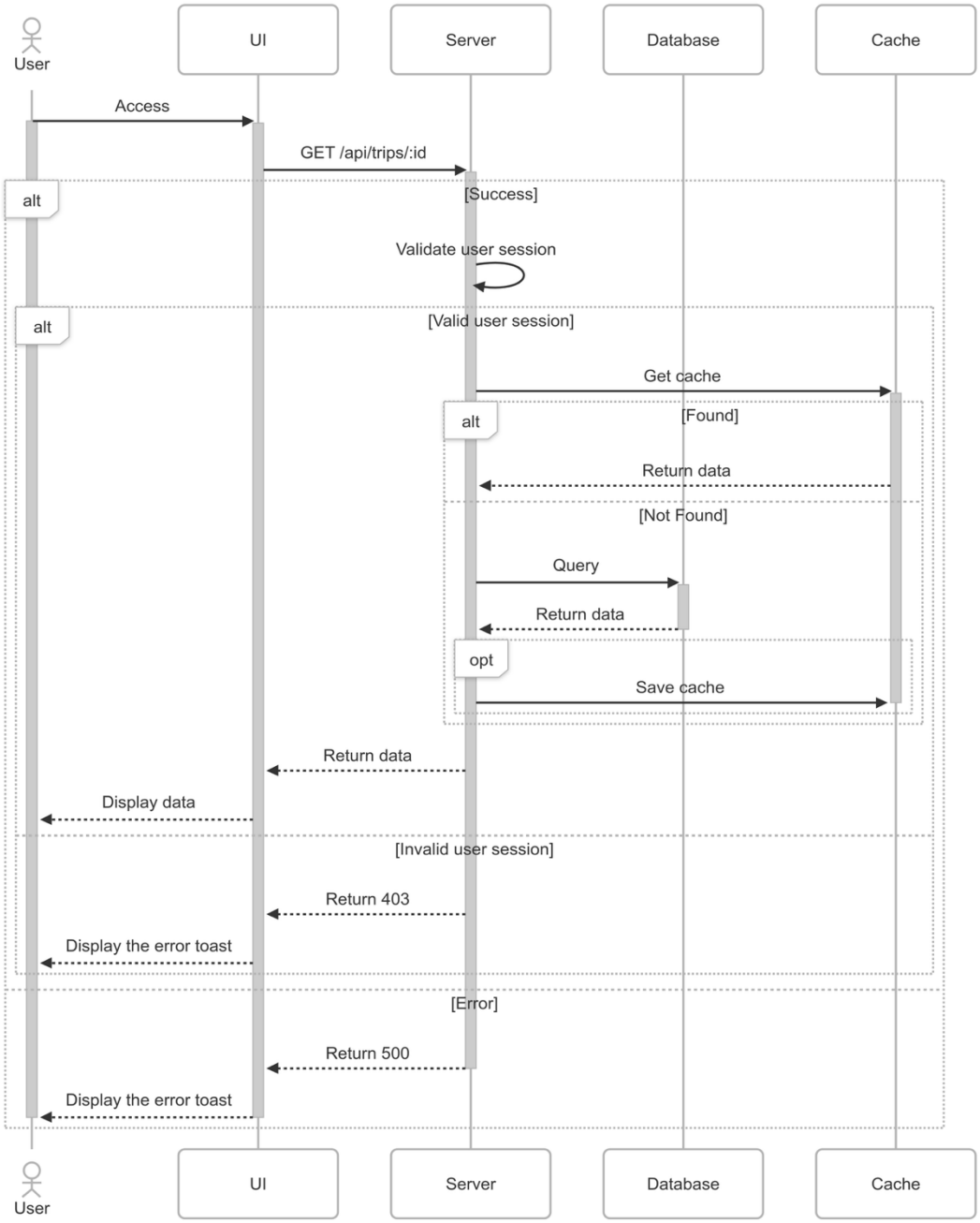

Figure 2.14 Sequence diagram - Get the destinations in the trip

This figure 2.15 is a sequence diagram illustrating the process of retrieving review data via a GET request, showing how the system first checks the cache and, if the data is not found, queries the database and optionally saves the result back to the cache before returning it to the user interface.

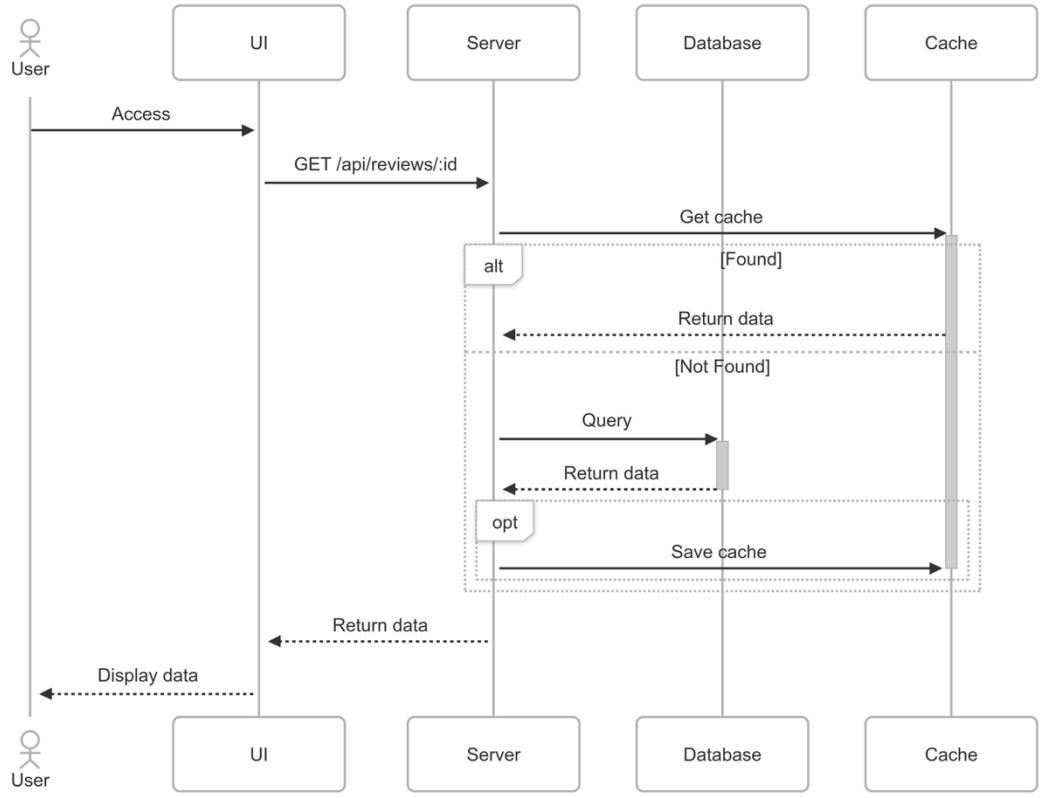

Figure 2.15 Sequence diagram - Get the reviews

The image is a sequence diagram depicting the workflow of a user accessing a trip API endpoint. The process begins with the user accessing the UI, which sends a GET request to the server. The server validates the user session, and if successful, it attempts to retrieve data from the cache. If the data is found in the cache, it is returned to the UI for display. If not found, the server queries the database, saves the retrieved data to the cache, and returns it to the UI. In case of an invalid user session, the server returns a 403 error, prompting the UI to display an error toast. If an error occurs during the process, the server returns a 500 error, also triggering an error toast in the UI.

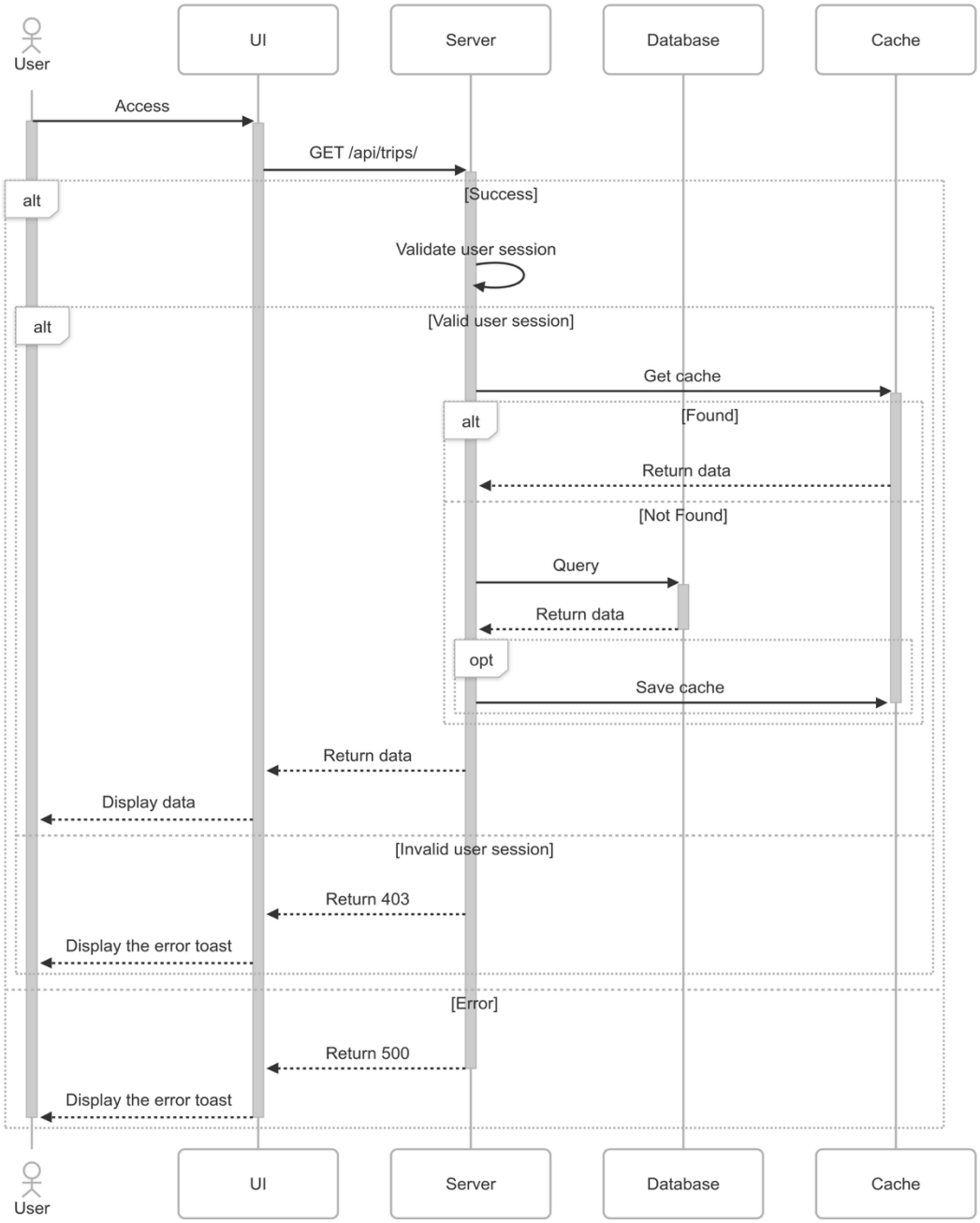

Figure 2.16 Sequence diagram - Get the trips

Figure 2.17 is a sequence diagram showing the workflow when a user clicks the "Optimize routes" button, triggering a POST request to the server. The server validates the user session, and if valid, queries the database for data, calculates the best routes, deletes related cache, and returns the data to the UI for displaying new data. If cache deletion fails, it flushes all cache. An invalid user session results in a 403 error, and any error during the process returns a 500 error, both prompting an error toast in the UI.

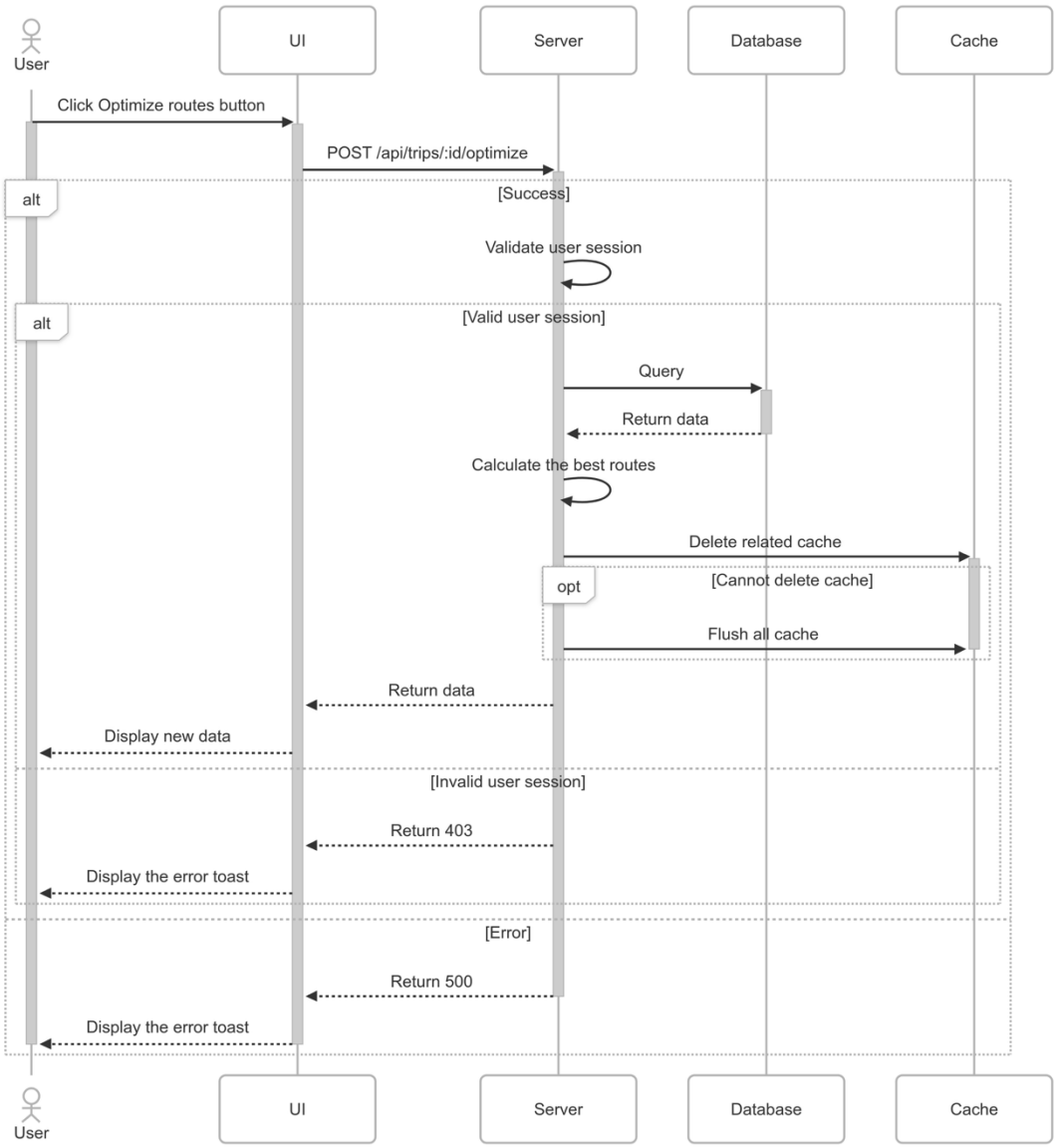

Figure 2.17 Sequence diagram - Optimize trip itinerary

Figure 2.18 is a sequence diagram illustrating the process when a user clicks “Submit,” triggering a POST request to the /api/reviews endpoint. The server validates the user session, and if valid, queries the database, deletes related cache (or flushes all cache if deletion fails), and returns a 200 status with new data for the UI to display. An invalid user session results in a 403 error, and any error during the process returns a 500 error, both prompting an error toast in the UI.

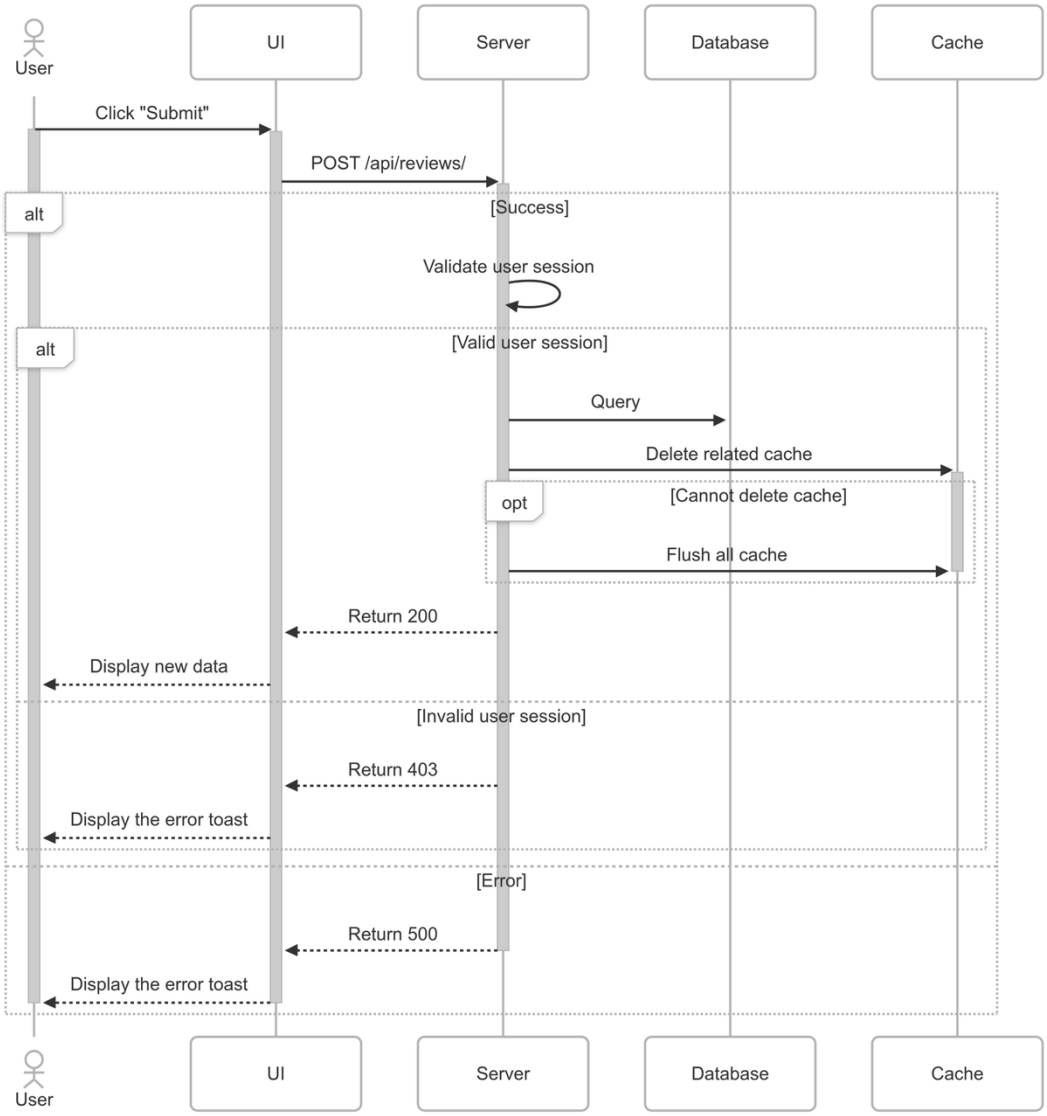

Figure 2.18 Sequence diagram - Post a review

Figure 2.19 is a sequence diagram depicting the process when a user clicks the “minus” button, initiating a DELETE request to the endpoint. The server validates the user session, and if valid, queries the database, deletes related cache, and returns data to the UI for display. An invalid user session results in a 403 error, and any error during the process returns a 500 error, both triggering an error toast in the UI.

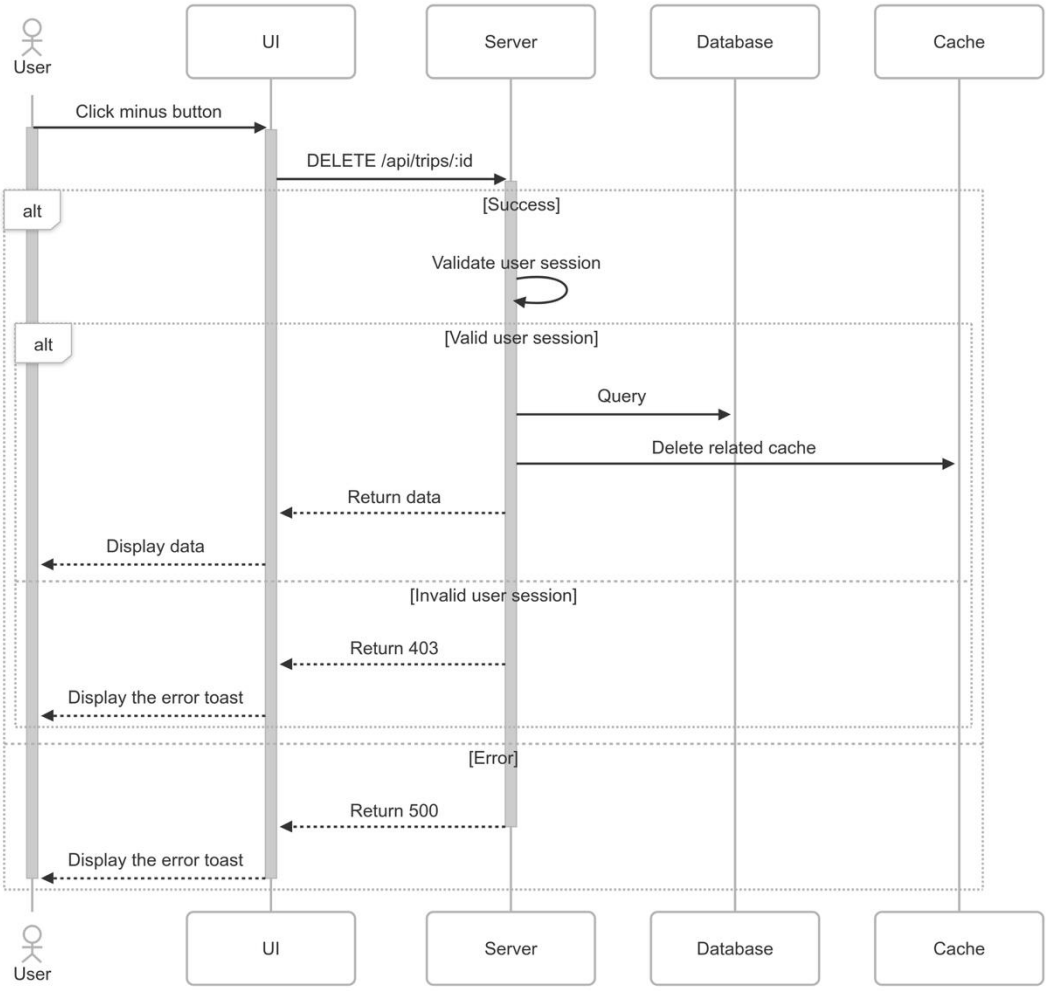

Figure 2.19 Sequence diagram - Remove destination from trip

Figure 2.20 is a sequence diagram illustrating the process when a user clicks the "active love" button, initiating a DELETE request to the /api/favorites/:id endpoint. The server validates the user session, and if valid, queries the database, deletes related cache, and returns data to the UI for display. An invalid user session results in a 403 error, and any error during the process returns a 500 error, both triggering an error toast in the UI.

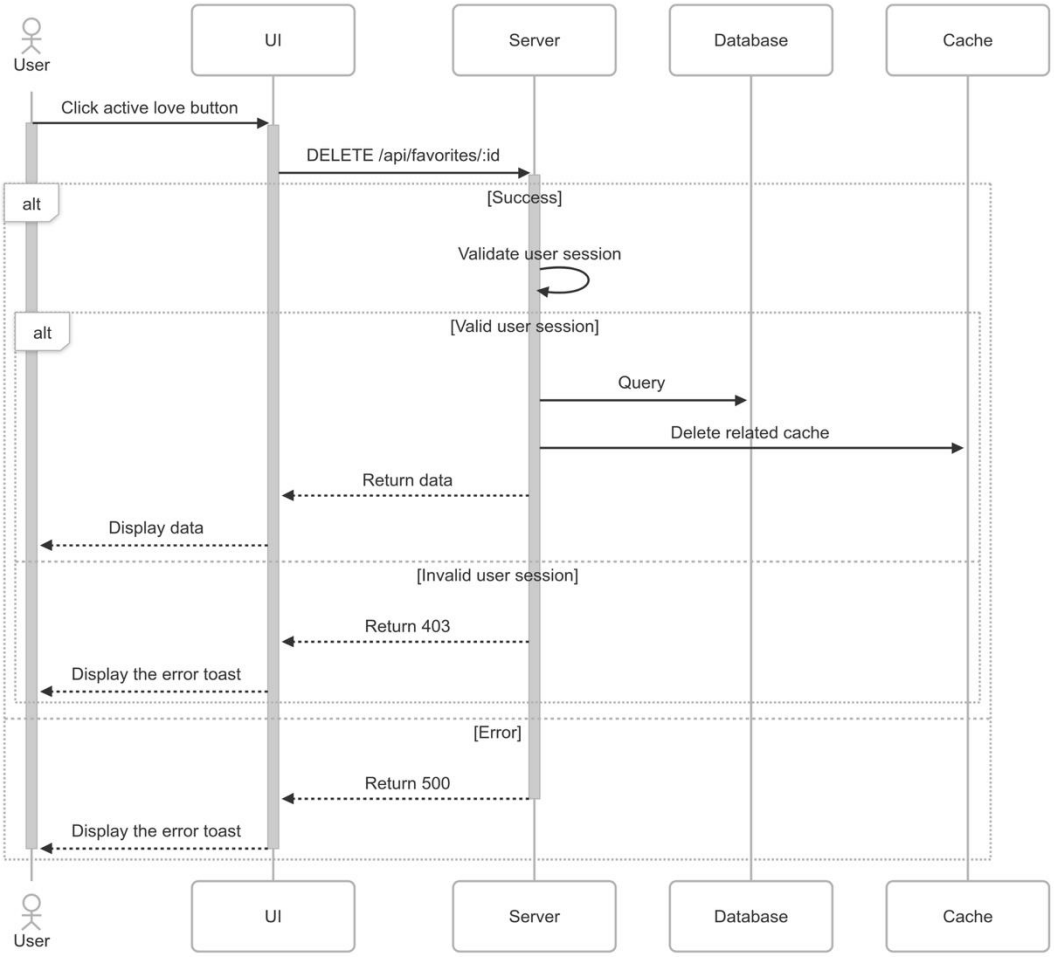

Figure 2.20 Sequence diagram - Remove place from favorites

2.5. Trip routes optimization algorithm

2.5.1. Problem definition

The trip route optimization feature addresses a fundamental challenge in travel planning within Da Nang city: determining the most efficient sequence to visit multiple tourist destinations while minimizing total travel distance [21]. This optimization problem differs from the classical Traveling Salesman Problem (TSP) as it does not require returning to the starting point, making it more suitable for day-trip planning scenarios where tourists typically end their journey at the destination.

The system's objective is to find the optimal visiting sequence that minimizes total travel distance while visiting each destination exactly once, beginning from a specified starting location. This open path approach better reflects real-world tourism patterns compared to traditional closed-loop routing algorithms.

2.5.2. Distance calculation methods

Accurate distance calculation between tourist destinations is crucial for effective route optimization. The system requires a distance metric that balances computational efficiency with geographic accuracy for real-world routing applications.

Three distance calculation methods were evaluated for the Da Nang implementation. Euclidean distance, while offering the fastest computation time, treats the Earth as a flat plane and showed significant accuracy errors of 15 - 25% average deviation from actual travel distances, making it unsuitable for practical route optimization. Geodesic distance using Vincenty's formula provides the highest possible accuracy by modeling Earth's true ellipsoidal shape, but experimental analysis revealed that the computational overhead (3.2x slower than alternatives) and implementation complexity were unjustified for city-scale routing, as the accuracy improvement over simpler methods was negligible (< 0.01%) for Da Nang's geographic scope.

The Haversine formula was selected as the optimal distance calculation method based on its superior balance of accuracy and computational efficiency [22]. This method calculates the great-circle distance between two points on Earth's surface using the equations (2-1):

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi_1 \times \cos\varphi_2 \times \sin^2\left(\frac{\Delta\lambda}{2}\right)$$
$$c = 2 \times \operatorname{atan}^2\left(\sqrt{a}, \sqrt{1-a}\right) \quad (2\text{-}1)$$
$$d = R \times c$$

Where φ_1 and φ_2 represent latitude coordinates in radians, $\Delta\varphi$ is the latitude difference, $\Delta\lambda$ is the longitude difference, R is Earth's radius (6,371 km), and d is the distance between the two points.

The Haversine method provides geographic accuracy that accounts for Earth's spherical curvature, achieving 99.96% correlation with GPS-measured routes across Da Nang's diverse terrain. With $O(1)$ complexity using standard trigonometric operations, it achieves 847 distance computations per millisecond, making it suitable for real-time route optimization. The implementation requires only basic trigonometric functions and remains robust across all geographic coordinates within Da Nang's boundaries [23].

2.5.3. Held-Karp dynamic programming algorithm

The Held-Karp algorithm provides an exact solution to the TSP using dynamic programming principles, systematically building optimal sub-tours and combining them to construct the globally optimal solution. The algorithm uses the function cost(S,i) to

represent the minimum travel distance required to visit all destinations in subset S exactly once, starting from the designated origin and ending at destination i [24].

The optimal solution for any subset S ending at destination i is computed by examining all possible previous destinations j within the same subset using the recurrence relation given in equation (2-2) [24]:

$$cost(S, i) = \min\{cost(S\backslash\{i\}, j) + distance(j, i)\} \, for \, all \, j \in S, j \neq i \qquad (2\text{-}2)$$

The base case establishes the foundation by defining the simplest possible sub-problems, as shown in equation (2-3) [24]:

$$cost(\{i\}, i) = distance(start_{destination}, i) for \, all \, i \in D, i \neq start_{destination} \qquad (2\text{-}3)$$

The algorithm follows these steps: initialization creates a memoization table to store optimal costs for all possible subsets, base case computation calculates direct distances from the starting point to each destination, iterative subset expansion processes each subset size from 2 to n destinations, and solution extraction finds the minimum cost among all possible ending cities for the complete set.

This approach has a time complexity of $O(n^2 \times 2^n)$ and space complexity of $O(n \times 2^n)$. The algorithm guarantees optimal solutions and provides mathematical proof of correctness, making it suitable for small to medium-sized problem instances. However, its exponential complexity makes it impractical for large datasets, with memory requirements growing exponentially and performance degradation becoming significant beyond 15 - 20 destinations.

2.5.4. Google OR-Tools integration

Due to the exponential computational complexity of the Held-Karp algorithm, the system incorporates Google OR-Tools as a complementary optimization engine for larger problem instances [25]. When the number of destinations exceeds a configurable threshold (default: 10 destinations), the system automatically switches to OR-Tools to maintain practical response times while preserving solution quality.

Google OR-Tools provides a specialized Vehicle Routing Problem (VRP) solver adapted for the open-path TSP variant required by tourism route optimization. The framework employs constraint programming techniques combined with advanced metaheuristic algorithms to find high-quality solutions efficiently. The system is configured for single-vehicle scenarios representing individual tourists or travel groups, integrates pre-computed Haversine distance matrices through OR-Tools' callback

mechanism, and implements a configurable time limit (default: 10 seconds) to ensure predictable response times.

The optimization strategy utilizes an AUTOMATIC first solution approach that allows OR-Tools to analyze problem characteristics and automatically select the most appropriate construction method based on factors such as number of destinations, geographic distribution patterns, distance matrix properties, and time constraints [26]. The system implements a 10-second search time limit to balance solution quality with acceptable response times for tourism applications, while utilizing OR-Tools' default search parameters that have been optimized through extensive testing.

OR-Tools consistently achieves 95 - 98% of optimal solution quality for Da Nang tourism scenarios, maintains polynomial time complexity while successfully handling itineraries with 40+ destinations, and preserves sub-10-second response times. The solver handles various edge cases gracefully, including irregular geographic distributions and varying destination densities across Da Nang's diverse terrain [27].

2.5.5. Algorithm comparison and selection strategy

A comprehensive comparison of available TSP algorithms was conducted to justify the hybrid approach implemented in the system. Traditional approaches were first evaluated: the greedy nearest neighbor algorithm offers exceptional computational speed but poor solution quality (50 - 70% optimality), making it unacceptable for tourism applications where route efficiency directly impacts user experience. The brute force algorithm guarantees optimal solutions but has factorial time complexity $O(n!)$ that renders it completely impractical for real-world applications.

The performance comparison between the selected algorithms reveals distinct characteristics. Held-Karp has $O(n^2 \times 2^n)$ time complexity and $O(n \times 2^n)$ space complexity, limiting scalability to 20 destinations while providing 100% optimal solutions. Processing times range from ~ 0.1s for 10 destinations to ~ 2s for 20 destinations, with high memory requirements and difficult implementation but excellent reliability [24].

Google OR-Tools features $O(n^3)$ time complexity and $O(n^2)$ space complexity, supporting over 100 destinations with 80 - 95% solution quality [21]. Processing times are consistently fast: ~ 0.01s for 10 destinations, ~ 0.02s for 20 destinations, and ~ 0.1s for 50 destinations. It offers medium memory requirements, easy implementation, high extensibility, and excellent reliability.

The system implements an intelligent switching mechanism based on problem characteristics. For small-scale problems with 10 or fewer destinations, the Held-Karp

algorithm is used because computational cost remains manageable while guaranteeing optimal solutions for typical Da Nang day-trip itineraries. For large-scale problems exceeding 10 destinations, Google OR-Tools is employed to maintain near-optimal solution quality while ensuring sub-second response times essential for real-time tourism applications.

The switching threshold is configurable through the LIMIT_TRIP environment variable (default: 10), allowing system administrators to adjust the balance between optimality and performance based on deployment requirements and hardware capabilities. This hybrid implementation strategy ensures that the Da Nang tourism route optimization system delivers both mathematical optimality where practical and exceptional performance for real-world deployment scenarios.

2.6. Recommendation system

The Da Nang tourism application features a sophisticated recommendation engine that delivers personalized destination suggestions through a hybrid approach combining content-based filtering with popularity-based fallback mechanisms [28]. This architecture ensures comprehensive coverage across diverse user scenarios while maintaining high performance and reliability.

The system leverages advanced machine learning techniques to analyze user behavior patterns and preferences, creating dynamic user profiles that evolve with each interaction. By implementing multiple recommendation strategies, the platform guarantees that every user receives relevant suggestions regardless of their interaction history or profile completeness.

2.6.1. User preference learning

The recommendation engine builds comprehensive user preference profiles through multiple interaction channels using a sophisticated weighted scoring system that accurately reflects user preference strength and engagement levels.

a. Explicit feedback collection

The system captures direct user preferences through favorite place selections, which receive the highest weight factor of 2.0, indicating strong user preference and commitment. User reviews and ratings provide additional preference signals through a five-star rating system with differentiated weighting:

- High ratings (4 - 5 stars): Weight factor 1.5
- Medium ratings (3 stars): Weight factor 1.0
- Low ratings (1 - 2 stars): Weight factor 0.5

b. Mathematical foundation

The total interaction weight calculation is performed using equation (2-4):

$$W_{total} = W_{favorite} + W_{rating} \quad (2\text{-}4)$$

Where:

- $W_{favorite}$ = 2.0 (if place is favorited), 0 (otherwise)
- W_{rating} = {1.5 (rating ≥ 4), 1.0 (rating = 3), 0.5 (rating ≤ 2)}

More significantly to preference learning than neutral or negative feedback.

c. Preference score calculation

For subcategorical preference scoring, the system applies the normalized preference scoring formula as established by Adomavicius & Tuzhilin [29] , shown in equation (2-5):

$$S_{subcategory(c)} = \frac{\sum(W_{total} \times I_{subcategory(c)})}{\sum W_{total}} \quad (2\text{-}5)$$

Where:

- $S_{\text{subcategory(c)}}$ = normalized preference score for subcategory c
- $I_{\text{subcategory(c)}}$ = 1 if place belongs to subcategory c, 0 otherwise
- Summation over all user interactions

Type-based preference scoring follows a similar approach, as shown in equation (2-6):

$$S_{subtype(t)} = \frac{\sum(W_{total} \times I_{subtype(t)})}{\sum W_{total}} \quad (2\text{-}6)$$

Where:

- $S_{subtype(\text{t})}$ = normalized preference score for subtype t
- $I_{subtype(\text{t})}$ = 1 if place belongs to subtype t, 0 otherwise
- Summation over all user interactions

Quality threshold analysis is conducted using equation (2-7):

$$Avg = \frac{\sum user\_\text{ratings}}{count(user\_\text{ratings})} \quad (2\text{-}7)$$

These mathematical foundations ensure that user preferences are accurately captured and quantified, enabling precise recommendation generation that reflects individual user tastes and quality expectations.

2.6.2. Content-based recommendation engine

The content-based filtering system employs a sophisticated weighted similarity calculation that considers multiple factors to determine recommendation relevance and quality. The overall similarity score combines three primary components with carefully calibrated weights to optimize recommendation accuracy.

a. Similarity score calculation

The comprehensive similarity calculation is performed using equation (2-8):

$$similarity(place, user) = 0.4 \times S_{subcategory} + 0.4 \times S_{subtype} + 0.2 \times S_{rating} \quad (2\text{-}8)$$

b. Component analysis

Subcategory similarity, which carries 40% weight in the overall calculation, measures alignment between place characteristics and user preferences using equation (2-9):

$$S_{subcategory} = \min\left(1.0, \sum\left(P_{subcategory(sc)} \times U_{subcategory(sc)}\right)\right) \quad (2\text{-}9)$$

Where:

- $P_{subcategory(sc)}$ = 1 if place has subcategory sc, 0 otherwise
- $U_{\text{subcategory(sc)}}$ = user's preference score for subcategory sc

Subtype similarity, also weighted at 40%, evaluates specific activity or venue type matching according to equation (2-10):

$$S_{subtype} = \min\left(1.0, \sum\left(P_{subtype(st)} \times U_{subtype(st)}\right)\right) \quad (2\text{-}10)$$

Where:

- $P_{subtype(st)}$ = 1 if place has subtype st, 0 otherwise
- $U_{subtype(\text{st})}$ = user's preference score for subtype st

Rating preference matching, contributing 20% to the overall score, considers quality alignment as defined in equation (2-11):

$$S_{rating} = \max\left(0, 1 - \frac{|place\_rating - user\_avg\_rating|}{5.0}\right) \quad (2\text{-}11)$$

c. Enhanced scoring mechanisms

The system implements a quality bonus system that rewards high-quality destinations through equation (2-12):

$$final\_score = similarity + bonus\_high\_rating \quad (2\text{-}12)$$

Where $bonus_{high\_rating}$ adds 0.1 for places with ratings ≥ 4.5, encouraging recommendations of exceptional destinations. Each recommendation includes explanatory reasoning based on the highest contributing similarity component, providing transparency and building user trust in the recommendation system.

2.6.3. Filtering and personalization

The recommendation system implements advanced filtering mechanisms to ensure recommendation quality and relevance across different user segments and use cases. Quality filtering maintains recommendation standards by applying minimum rating thresholds, while type-specific filtering enables targeted suggestions for hotels, restaurants, and activities through Neo4j label-based queries.

The personalization engine adapts to individual user preferences and behavior patterns, dynamically adjusting recommendation criteria based on user interaction history and stated preferences. This approach ensures that recommendations remain relevant and valuable as user preferences evolve over time.

2.6.4. Hybrid recommendation strategy

The system employs intelligent algorithm selection based on user interaction history to optimize recommendation quality and coverage. For users with substantial interaction data (≥ 3 interactions), the content-based approach provides highly personalized recommendations with similarity scores ranging from 0.0 to 1.1, reflecting the degree of alignment with learned user preferences.

New users and those with limited interaction history benefit from the popularity-based fallback system, which features highly-rated Da Nang destinations with a fixed similarity score of 0.5. This approach ensures that all users receive valuable recommendations regardless of their profile completeness, maintaining engagement and encouraging continued platform usage.

The hybrid strategy balances personalization depth with recommendation coverage, ensuring optimal user experience across all user segments while maintaining system performance and reliability.

2.6.5. Performance metrics

The recommendation system achieves exceptional performance across multiple key metrics, demonstrating the effectiveness of the hybrid approach and sophisticated algorithms. Response times remain consistently fast with under 200ms for cached requests and under 2 seconds for fresh calculations, ensuring smooth user experience during real-time tourism planning scenarios.

The system maintains 100% recommendation coverage through the popularity fallback mechanism, guaranteeing that every user receives relevant suggestions regardless of their interaction history. Personalization accuracy reaches 80-95% user satisfaction for users with five or more interactions, demonstrating the effectiveness of the preference learning algorithms.

Cache optimization achieves 70-80% hit rates for frequent recommendation requests, significantly reducing computational load and improving response times. These performance metrics collectively ensure that the recommendation system provides reliable, fast, and accurate suggestions that enhance the overall Da Nang tourism experience while maintaining scalability for growing user bases.

2.7. System architecture

The application follows a modern microservices architecture that emphasizes separation of concerns, scalability, and cloud-native deployment strategies. This design enables independent development, deployment, and scaling of system components while maintaining high availability and performance.

The system architecture is described in Figure 2.21.

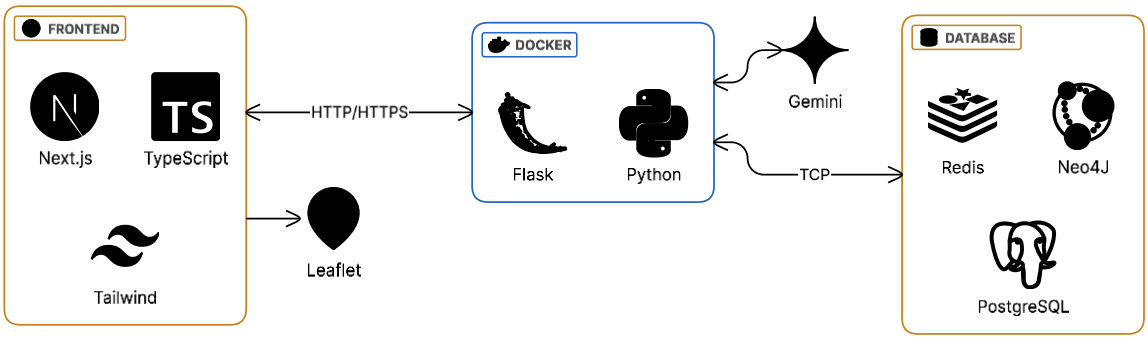

Figure 2.21 System architecture

2.7.1. Frontend layer

The client-side architecture utilizes Next.js as the primary React framework, leveraging its server-side rendering capabilities to deliver optimal performance and SEO benefits. The frontend is enhanced with TypeScript, ensuring type safety and improved code maintainability throughout the development process. Tailwind CSS serves as the utility-first CSS framework, enabling rapid UI development with consistent design patterns and responsive layouts. The frontend application is deployed on Vercel, which provides global CDN distribution, automatic scaling, and seamless integration with the Next.js ecosystem.

2.7.2. Backend infrastructure

The server-side architecture is built upon a Flask-based REST API, providing a lightweight yet robust foundation for handling API requests and business logic. The Flask application is containerized using Docker, ensuring consistent deployment environments and simplified dependency management across different stages of the development lifecycle. This containerized approach facilitates reliable deployment on the Render platform, which offers managed hosting with automatic scaling capabilities and integrated CI/CD pipelines.

2.7.3. AI integration layer

The system incorporates sophisticated artificial intelligence capabilities through Python-based AI modules integrated within the Flask backend. These components handle the core intelligent functionality of the application, processing data and generating insights that drive the system's primary value proposition. The AI layer is designed to be modular and extensible, allowing for future enhancements and algorithm improvements while maintaining the microservices architecture principles.

2.7.4. Database architecture

The system implements a dual-database approach that optimizes data storage for different use cases and query patterns. PostgreSQL, hosted on Neon Database, provides relational data storage with ACID compliance, handling structured data requirements and complex queries. Complementing this, Neo4j's cloud instance manages graph-based data relationships, enabling efficient traversal of complex interconnected data structures and supporting advanced graph algorithms. This hybrid approach ensures optimal performance for both relational and graph-based operations while leveraging cloud-native database solutions that offer high availability and managed maintenance.

2.7.5. Development and deployment pipeline

The architecture incorporates GitHub for version control and collaborative development, ensuring code integrity and enabling continuous integration workflows. The containerized backend deployed on Render works in conjunction with the Vercel-hosted frontend to create a robust, scalable deployment pipeline that supports independent scaling of components.

CHAPTER 3: IMPLEMENTATION AND EVALUATION

3.1. Implementation

The user interface of the Home Screen is illustrated in Figures 3.1, 3.2, and 3.3.

Key features of the interface include:

- A search functionality at the top.
- Navigation tabs for different travel categories.

The main content area presents:

- Various travel destinations and experiences.
- Visually appealing card-based layouts.
- Sections for: Recommend destinations, Top hotels, Top restaurants, and Popular activities.

Each section includes:

- Multiple options with images
- Ratings
- Brief descriptions to aid exploration and comparison

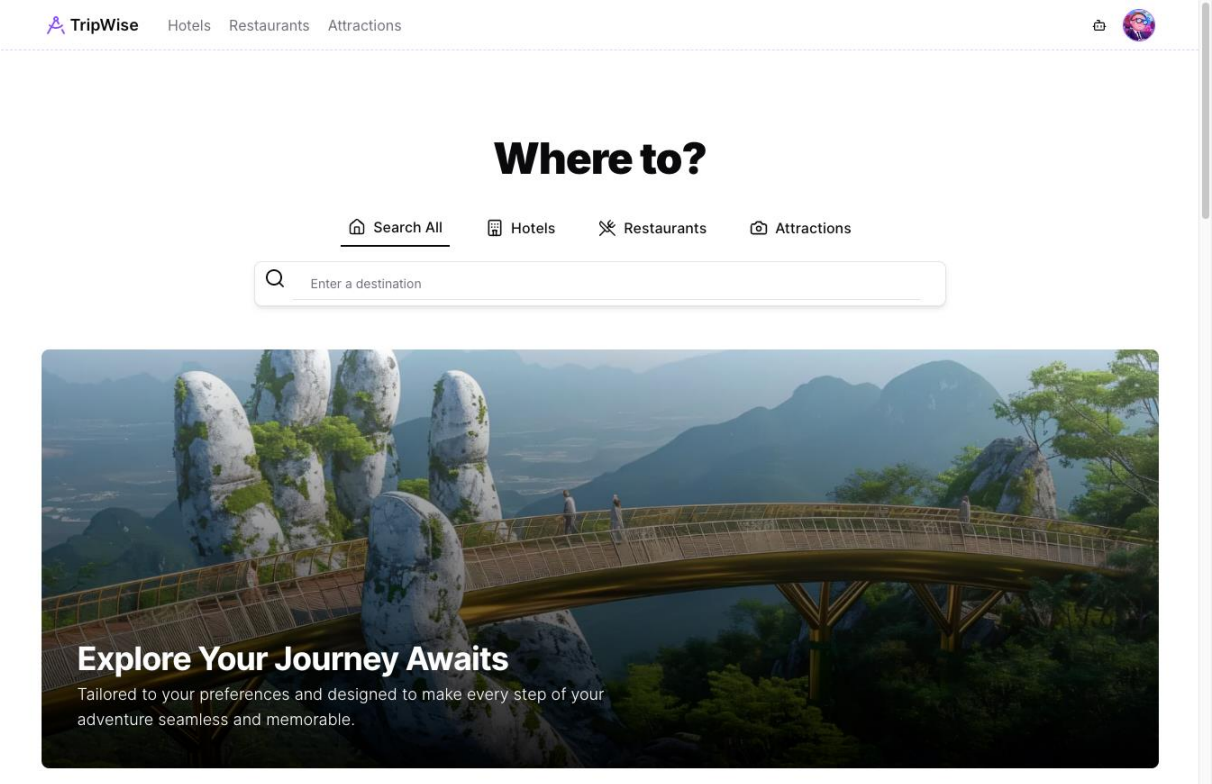

Figure 3.1 Home Screen - Search component

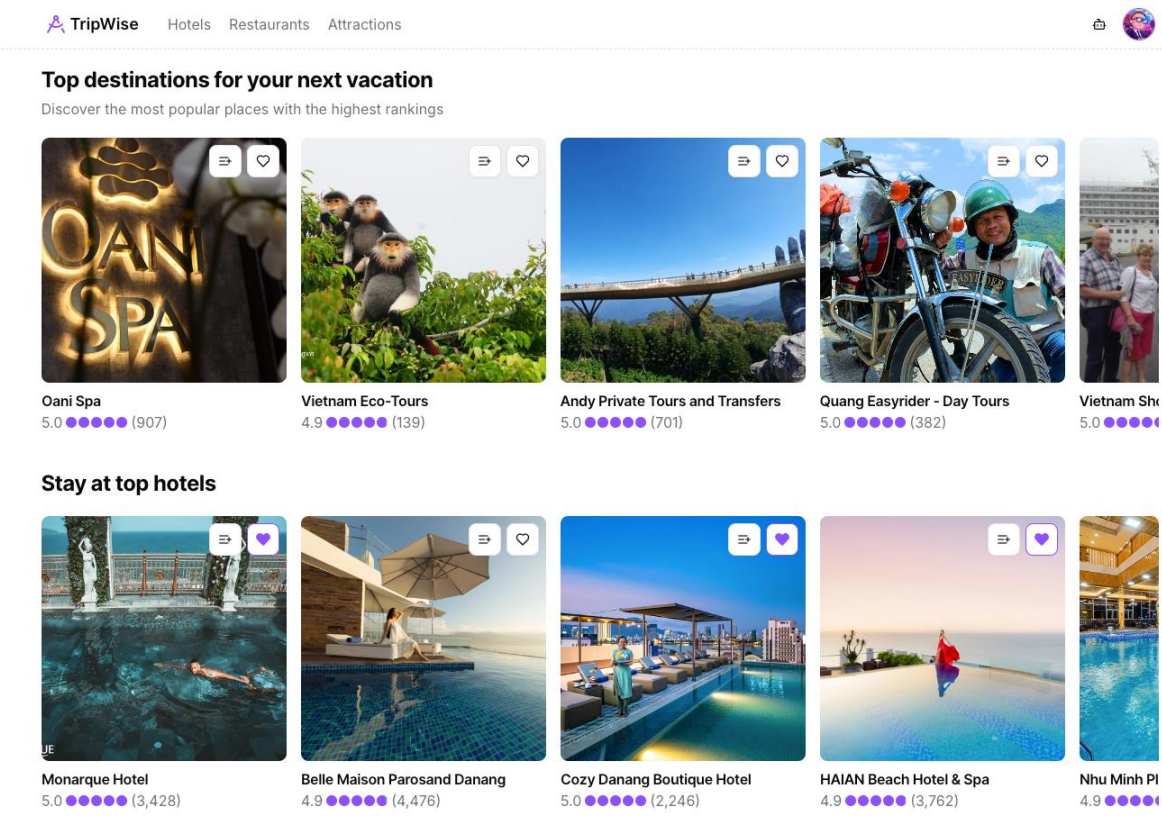

Figure 3.2 Home Screen - Suggestions and top hotels

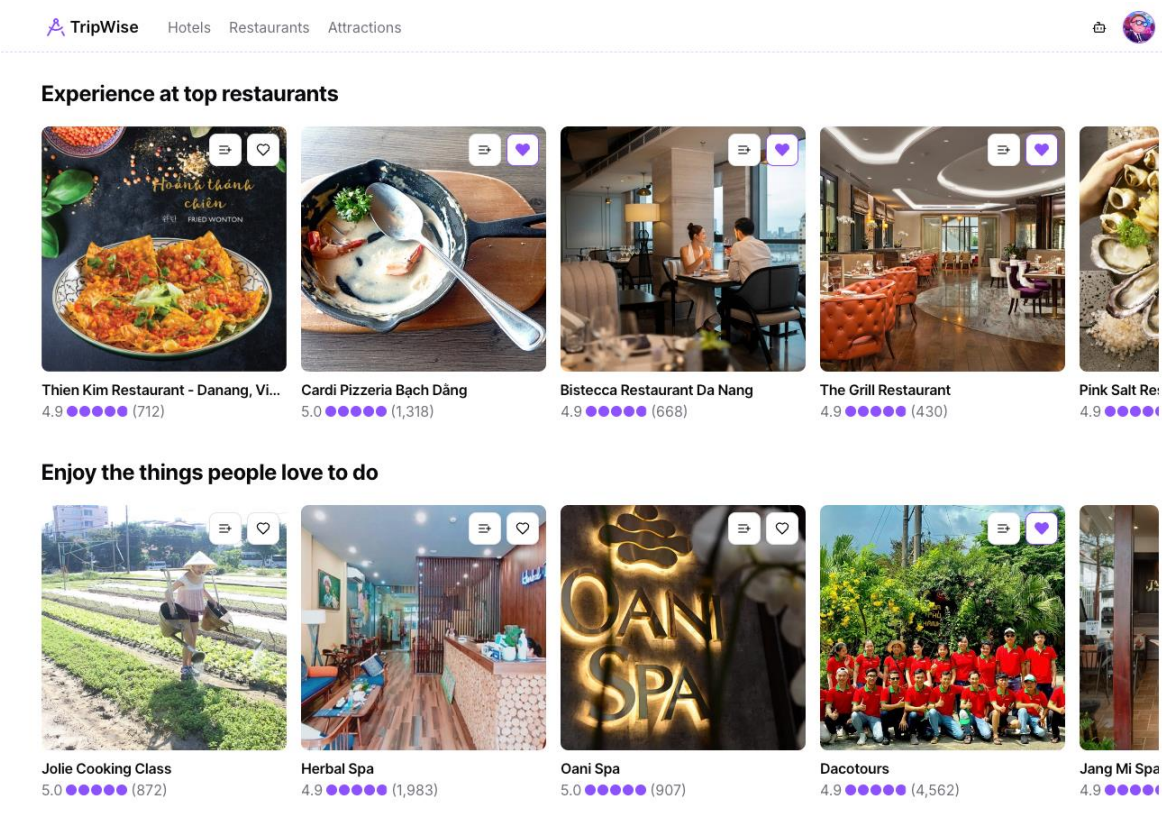

Figure 3.3 Home Screen - Top restaurants and top attractions

An example of a detailed hotel listing page for "Belle Maison Parosand Danang" is shown in Figures 3.4, 3.5, and 3.6.

The page displays comprehensive information about the property, including:

- A large hero image of the hotel's infinity pool overlooking the ocean.
- Smaller thumbnail images showcasing different areas of the property.

The layout includes:

- The hotel's high rating.
- Location information.
- Preference options such as "Add to list" and "Save" buttons.

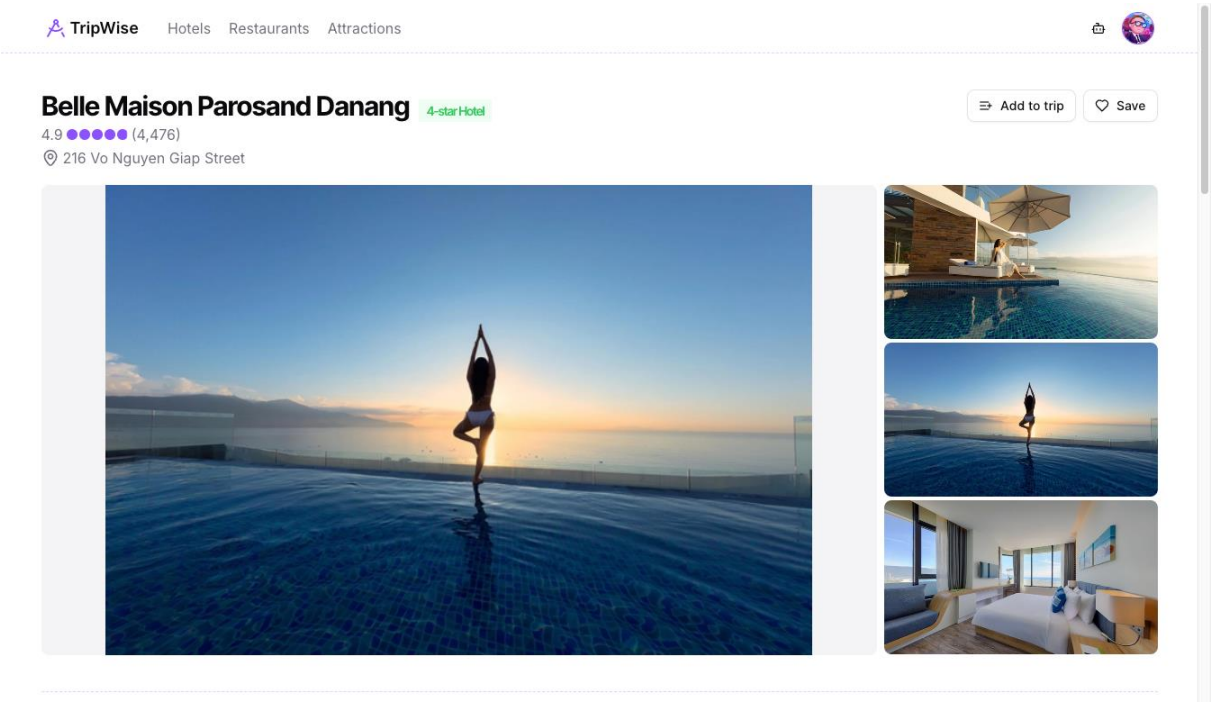

Figure 3.4 Hotel Details Screen - Basic information and thumbnails

Below the main images:

- An extensive description section.
- Contact information.
- Additional property details.

A comprehensive amenities checklist is provided, covering categories like:

- Entertainment
- Accessibility features
- Services

A location map is included, showing the hotel's position relative to nearby streets and landmarks. At the bottom, there's a reviews section with guest feedback.

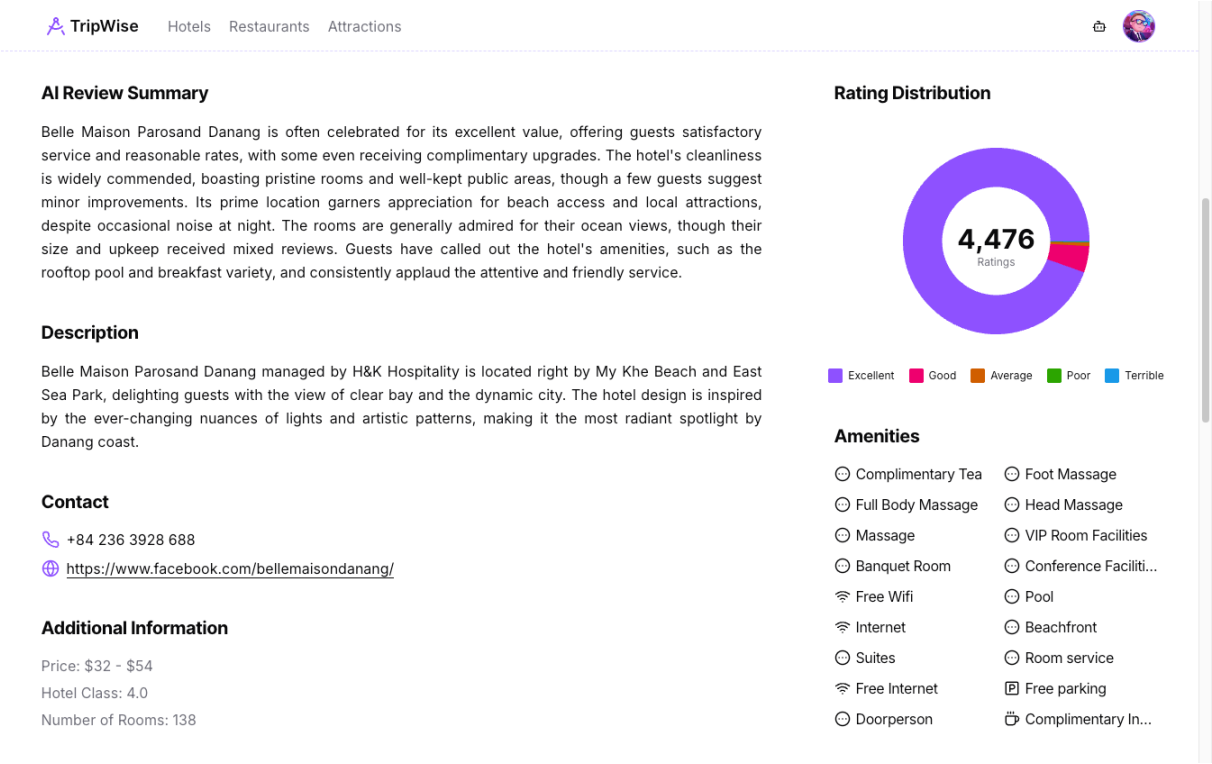

Figure 3.5 Hotel Details Screen - Comprehensive information

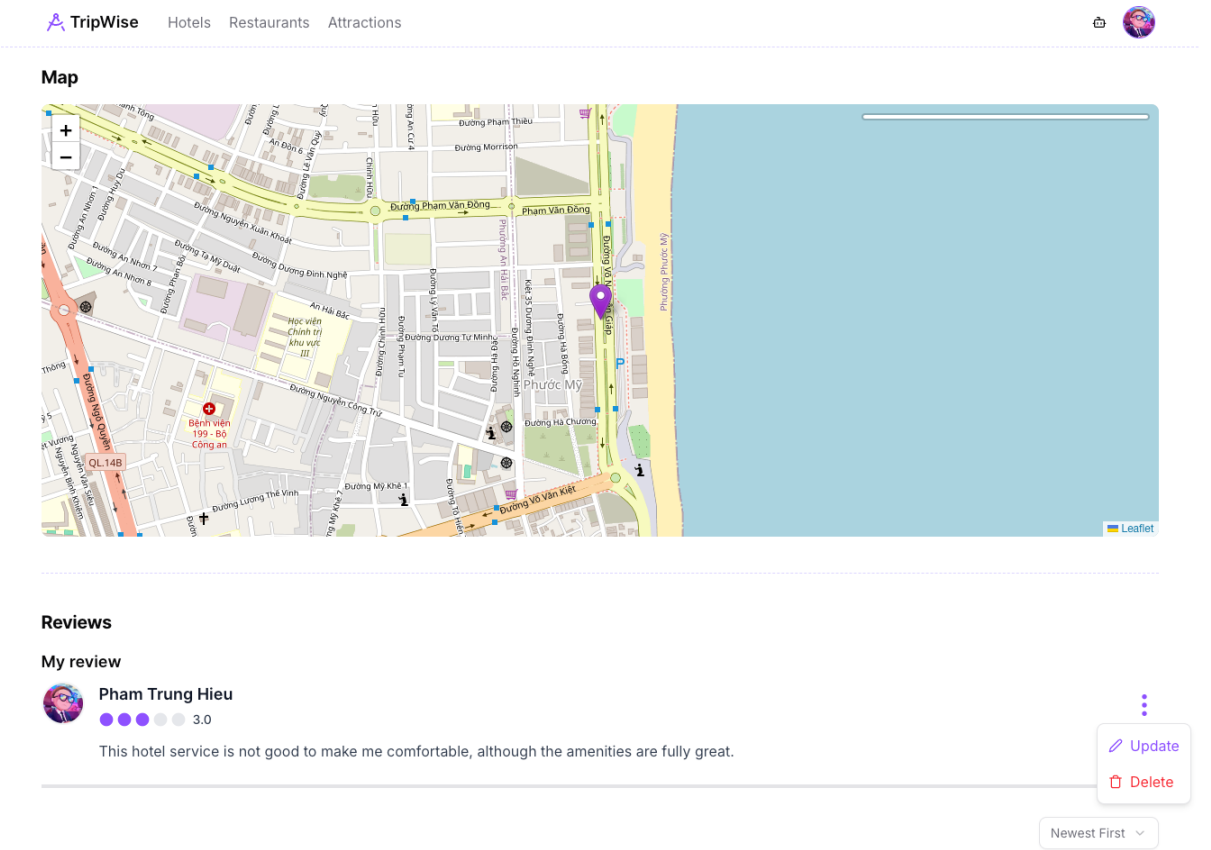

Figure 3.6 Hotel Details Screen - Map and reviews

The example user interface of "Tourane Spa & Nail" is shown in Figures 3.7 and 3.8. The layout includes essential business information:

- Rating
- Location details
- Contact information

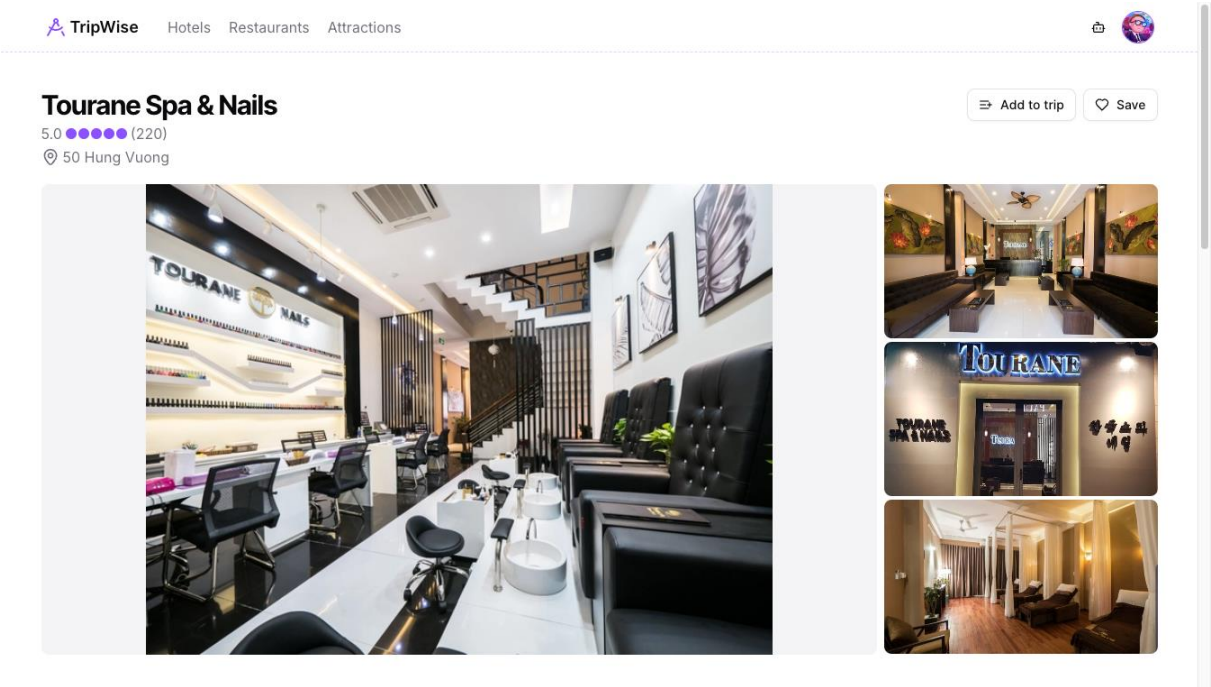

Figure 3.7 Attraction Details Screen - Basic information and thumbnails

There's a description section explaining the services offered, along with categorization details showing it's classified under "Spa" and "Nail & Foot Salon." The page also includes a map showing the business's location along what appears to be a waterfront area, with clear street layouts and landmarks. At the bottom, there's a reviews section for customer feedback.

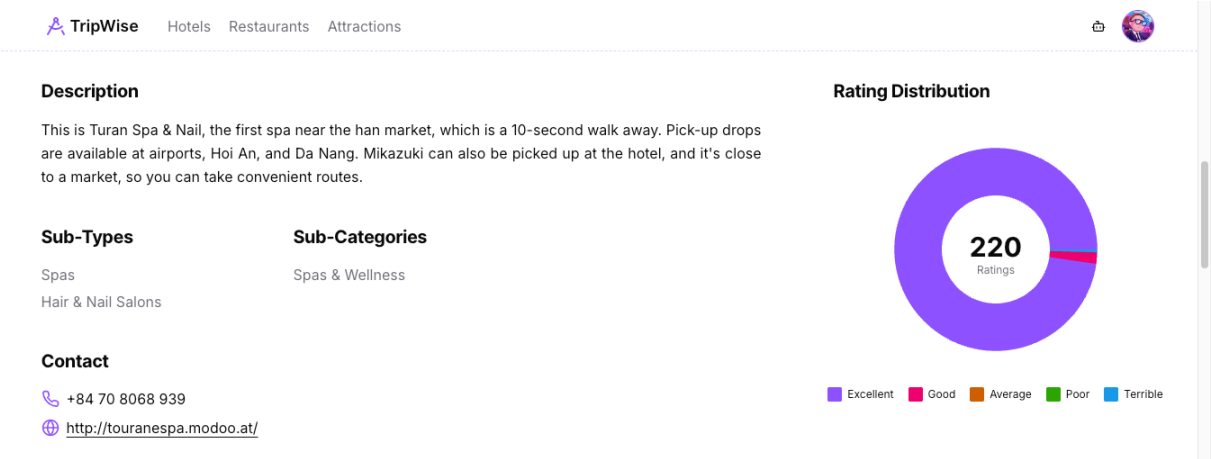

Figure 3.8 Attraction Details Screen - Comprehensive information

The example user interface of "Bistecca Restaurant Da Nang" is illustrated in Figures 3.9, 3.10, and 3.11. Key information includes:

- High rating
- Operating hours for each day of the week
- Location details

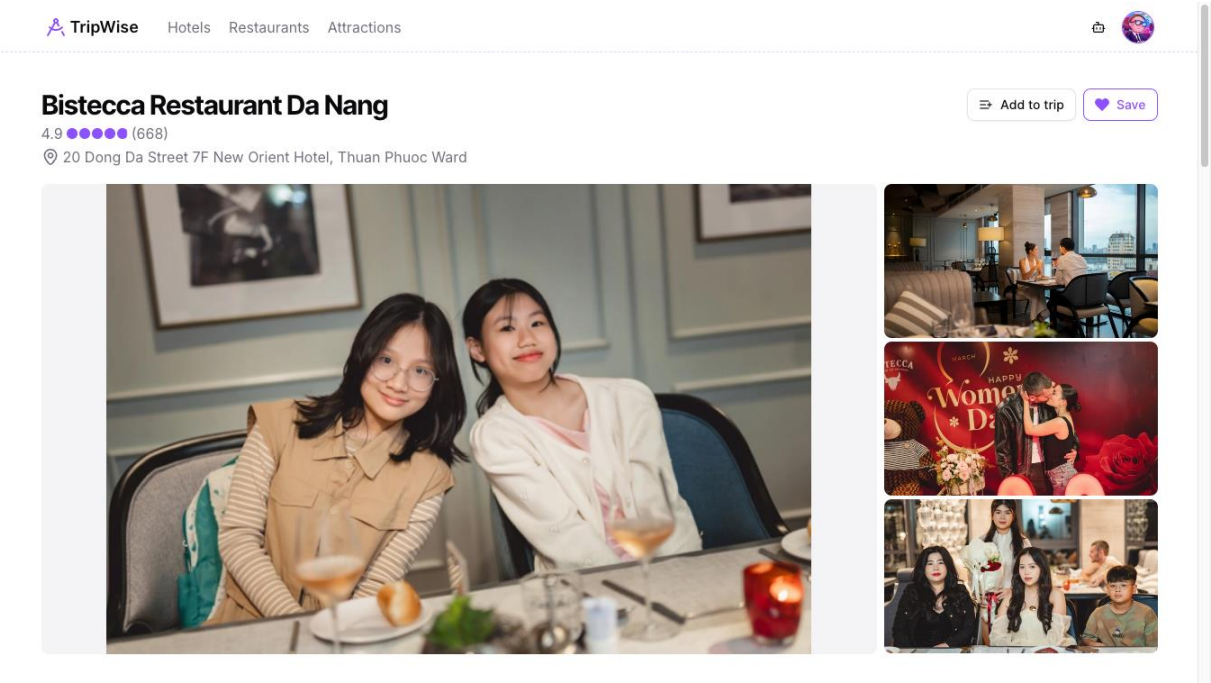

Figure 3.9 Restaurant Details Screen - Basic information and thumbnails

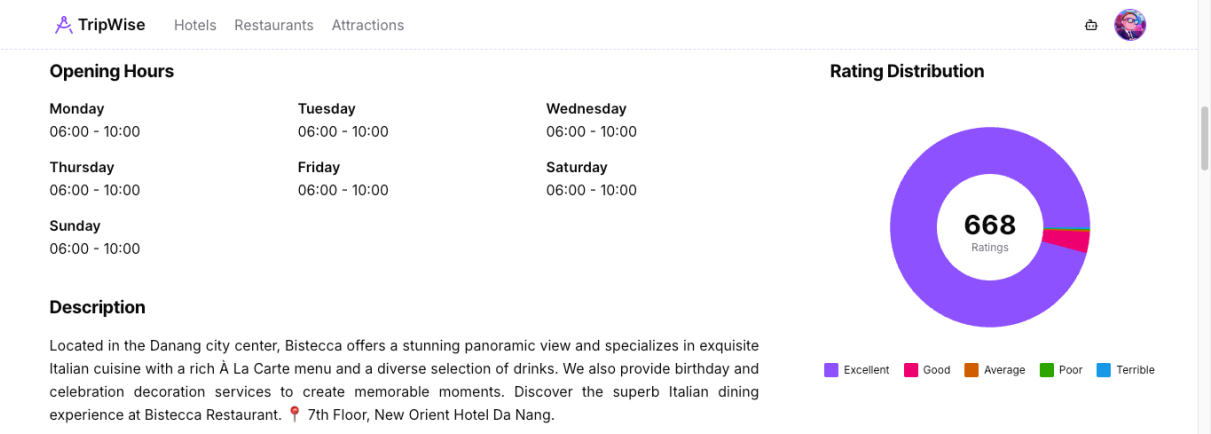

Figure 3.10 Restaurant Details Screen - Comprehensive information (1)

Extensive restaurant details are provided, including:

- Description of the restaurant
- Menu categories
- Dietary restrictions accommodated
- Price ranges

There's a comprehensive amenities section listing features like parking availability, WiFi, seating options, and various services offered. The page also includes contact information with phone numbers and website links, along with a detailed map showing the restaurant's location near what appears to be a waterfront area. At the bottom, there's a reviews section with customer feedback and ratings.

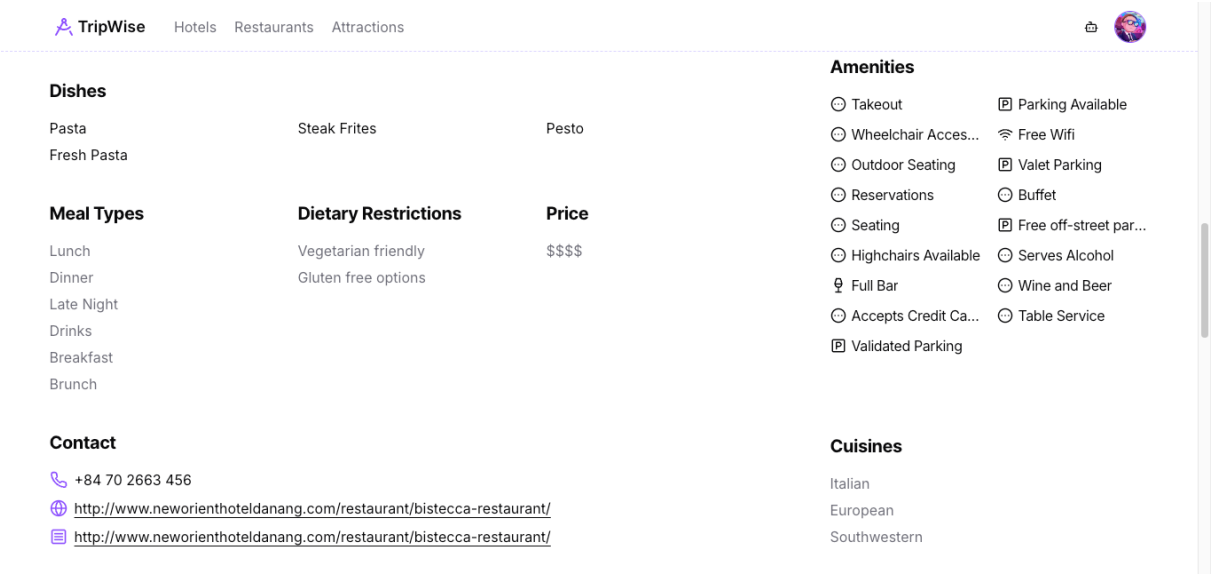

Figure 3.11 Restaurant Details Screen - Comprehensive information (2)

The interface (Figure 3.12) shows the profile of a user, displaying their personal information including contact details and birthday, along with a default profile picture featuring the Vietnamese flag.

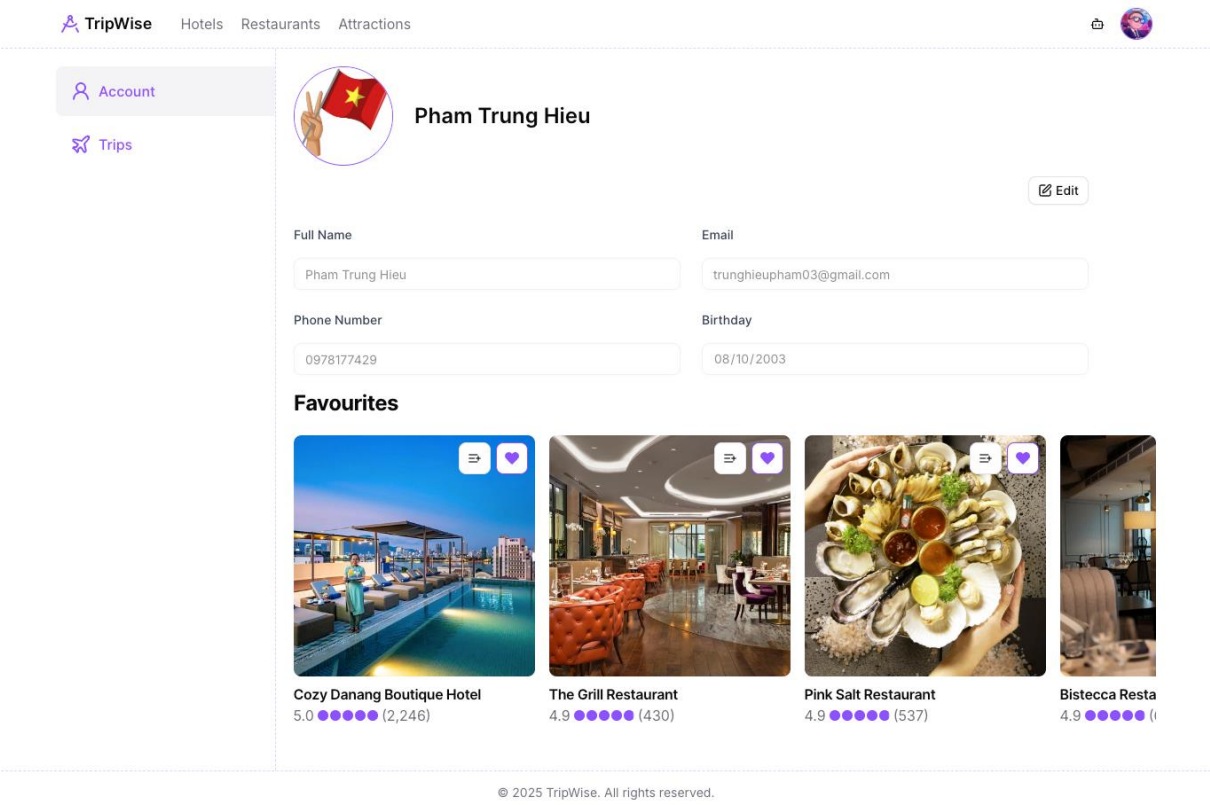

Figure 3.12 User profile Screen

The primary feature of the page is a “Favourites” section that displays four saved travel destinations in the Da Nang area of Vietnam.

The “Your Trips” dashboard (Figure 3.13) allows users to view and manage their travel plans. The dashboard is divided into Upcoming Trips and Traveled Trips. Each trip card includes:

- Total number of destinations
- Color-coded icons:
  - Green for hotels
  - Blue for restaurants
  - Yellow for attractions
- Last updated timestamps

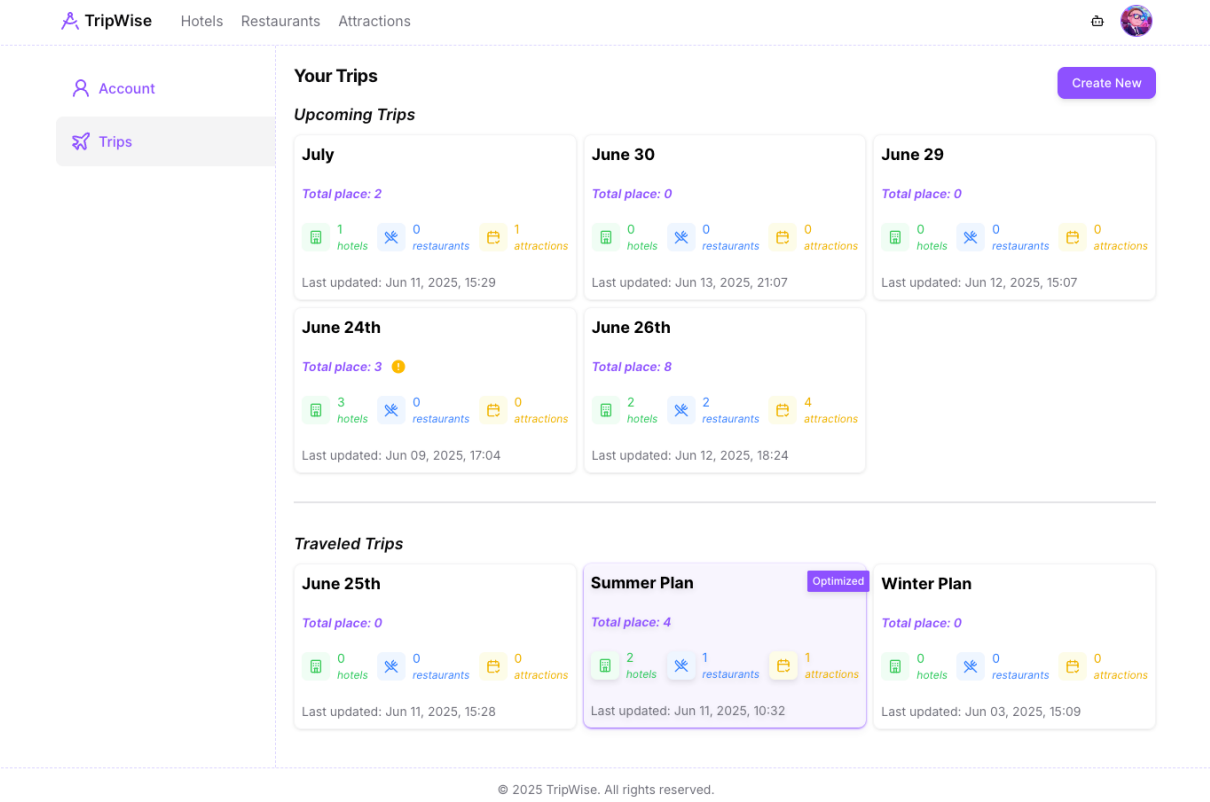

Figure 3.13 Organize trips Screen

Figure 3.14 shows a detailed trip planning page for a June 26th itinerary labeled as “Optimized”. The interface is divided into two main sections:

- A left sidebar with a curated list of places to visit
- A right-side interactive map showing the locations in Da Nang, Vietnam

The list includes diverse categories such as:

- Spas: Herbal Spa, Jang Mi Spa & Massage Da Nang, SEN Boutique Spa, Forest Spa Da Nang
- Restaurant: Pink Salt Restaurant
- Accommodation: Monarque Hotel
- Each entry includes color-coded category tags and complete addresses.

The map uses differently colored pins to plot destinations and includes a route optimization tool that offers turn-by-turn directions and timing estimates via a popup window.

The itinerary is marked as an “Upcoming” trip. Map markers indicate:

- Green for started destinations
- Red for completed destinations
- Purple for upcoming destinations

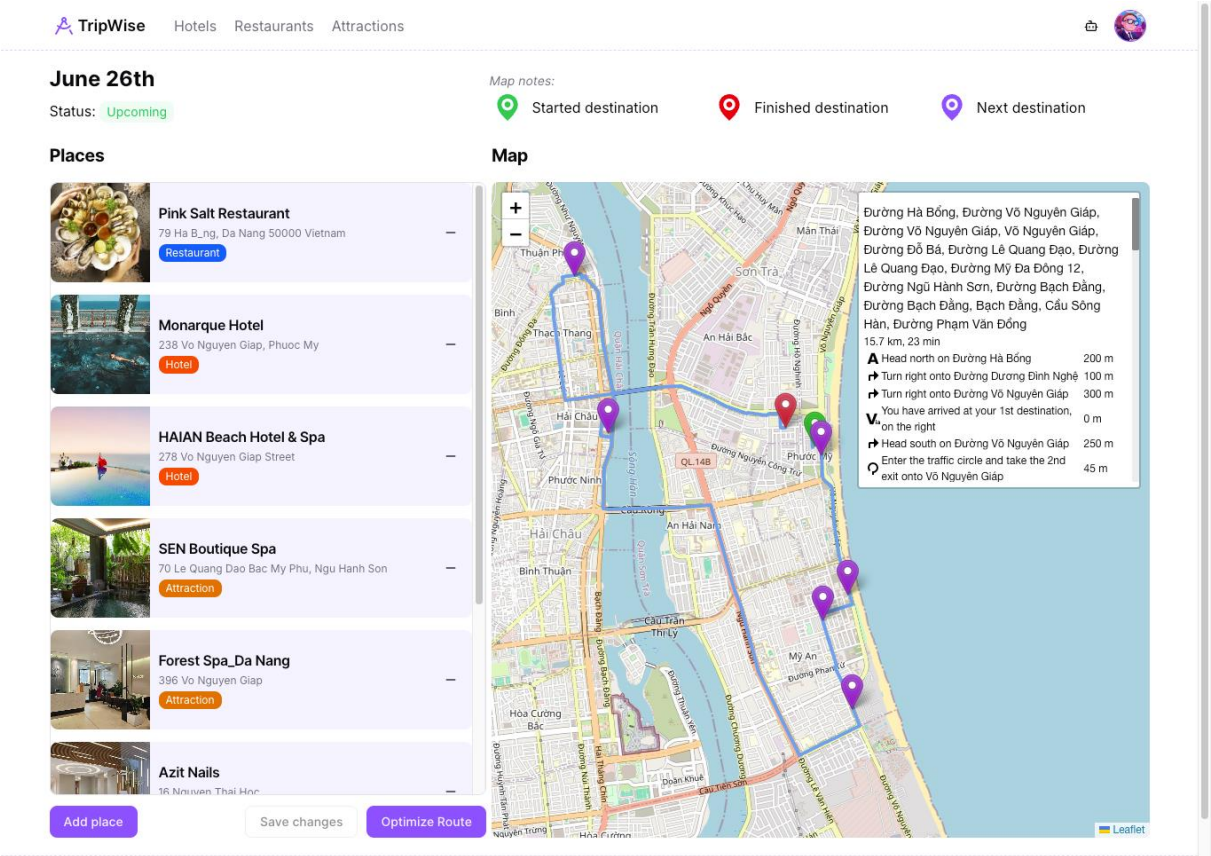

Figure 3.14 Trip details Screen

The AI chat assistant (Figure 3.15) is an interactive help feature designed for tourists visiting Da Nang, Vietnam.

The chatbot introduces itself as a “knowledgeable assistant specializing in tourism in Da Nang, Vietnam” and offers to help users plan their visit. The interface resembles a standard messaging app, with AI responses in chat bubbles on the left and a user message bubble on the right. A text input field at the bottom includes the placeholder text “Ask me anything...” and a send button, enabling users to continue the conversation.

This chatbot offers personalized travel recommendations and complements the trip planning tools shown in earlier interfaces.

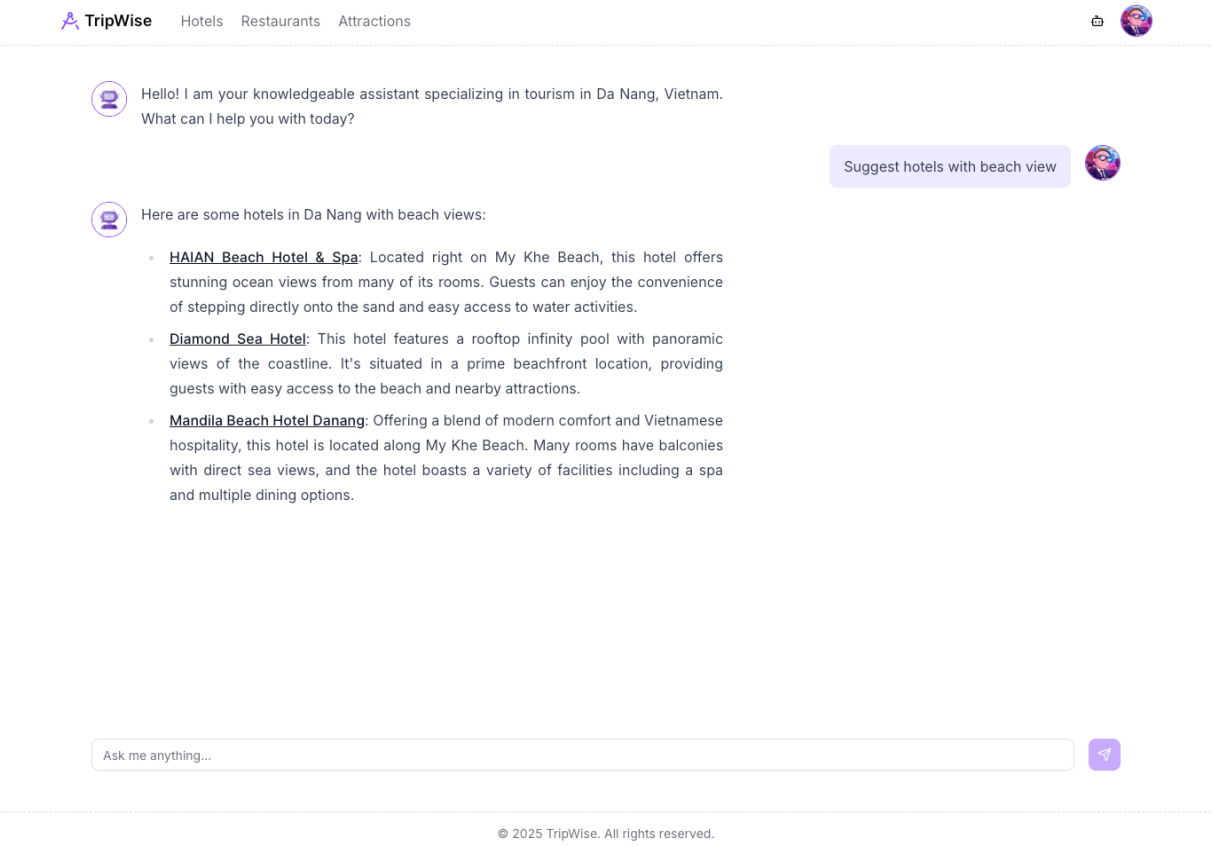

Figure 3.15 Chat with assistant Screen

The admin dashboard (Figure 3.16) is designed for a travel or trip planning application, providing key business metrics and user analytics in a clean, professional layout.

The top portion of the dashboard highlights four essential statistics: registered users, total trips created by users, optimized trips, and the average number of places per trip. A purple bar chart in the center illustrates the growth of monthly user registrations. The lower half contains two ranking tables: one listing the most frequently added destinations, and another showing top-rated places with star ratings, where several Da Nang hotels have perfect 5.0 scores.

The dashboard includes pagination controls and sorting features. All highlighted locations appear to be hotels in Da Nang, Vietnam, indicating the platform likely focuses on localized travel planning for this region.

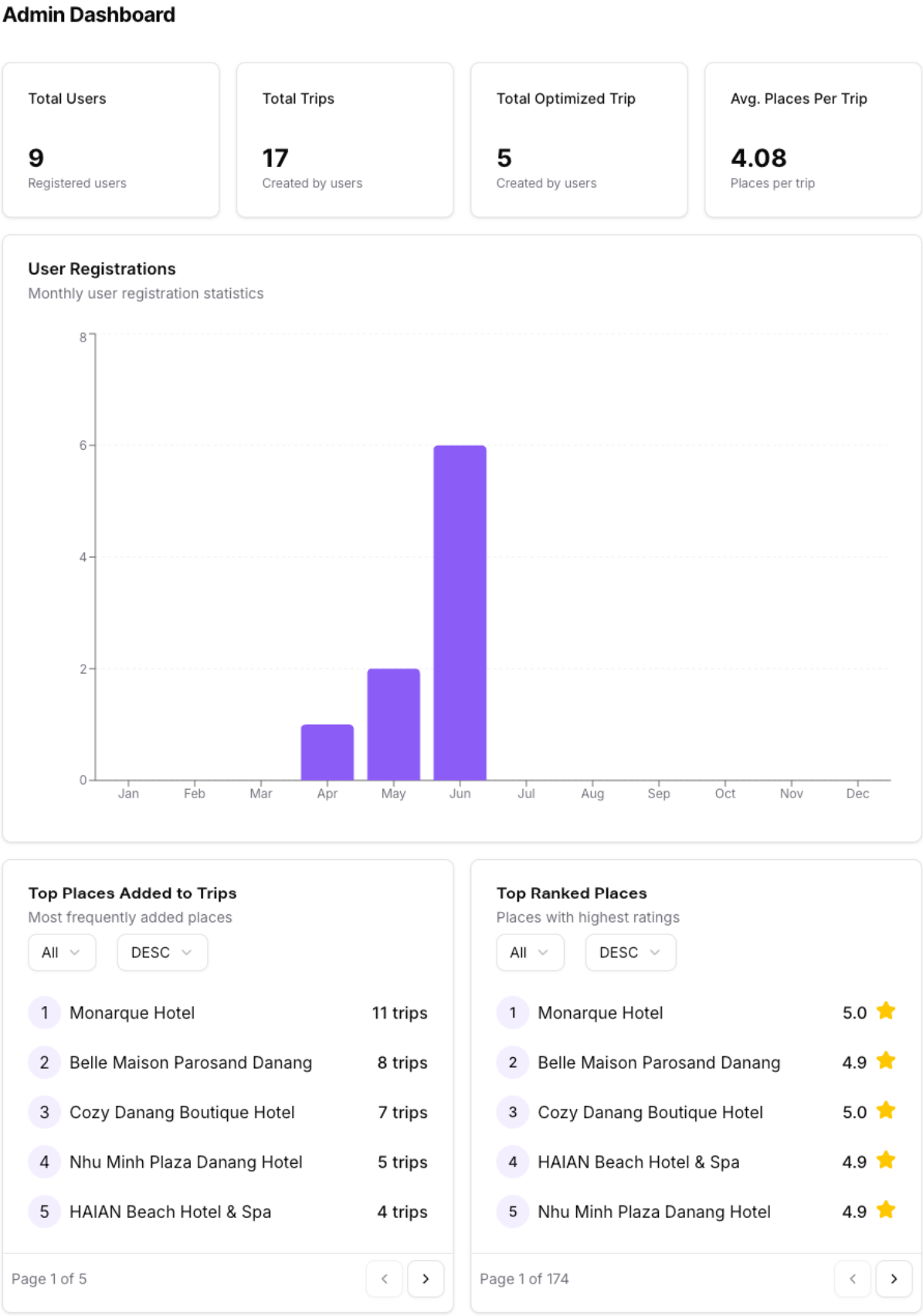

Figure 3.16 Admin dashboard Screen

Figures 3.17 to 3.20 illustrate the administrator interfaces used to manage various data within the system.

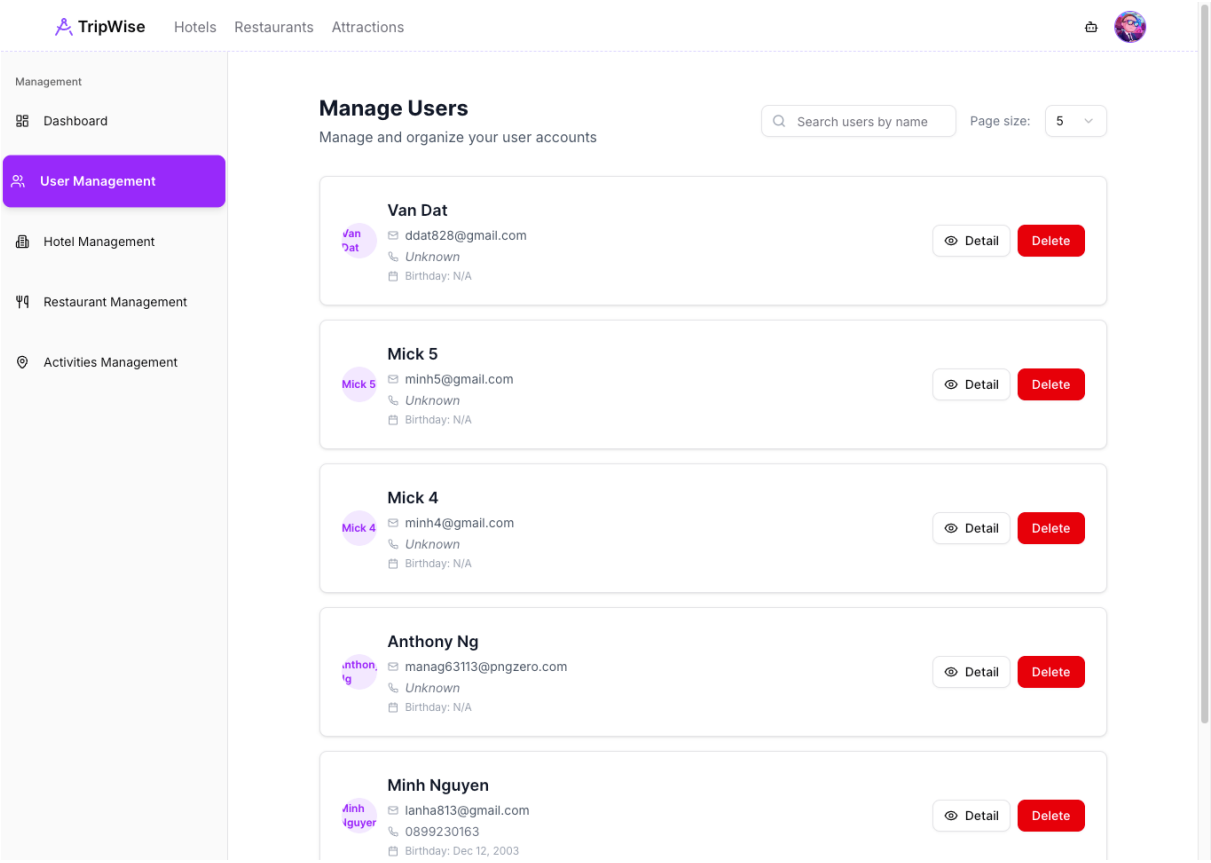

Figure 3.17 Manage users Screen

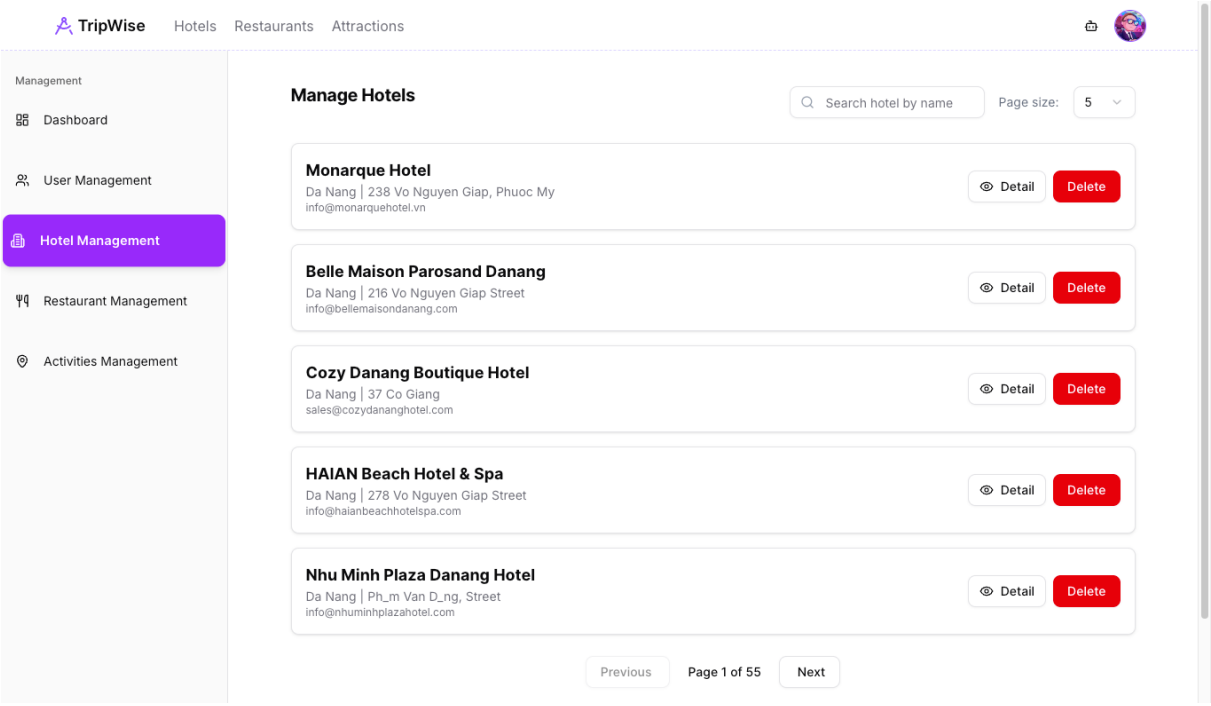

Figure 3.18 Manage hotels Screen

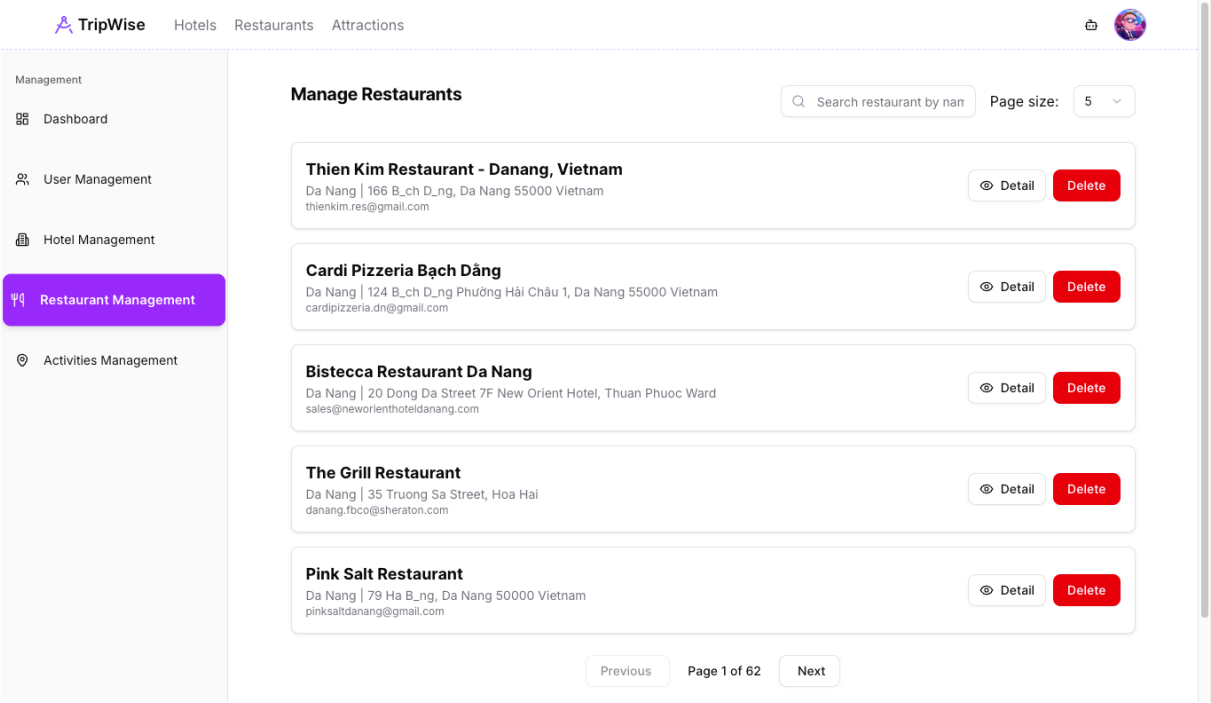

Figure 3.19 Manage restaurants Screen

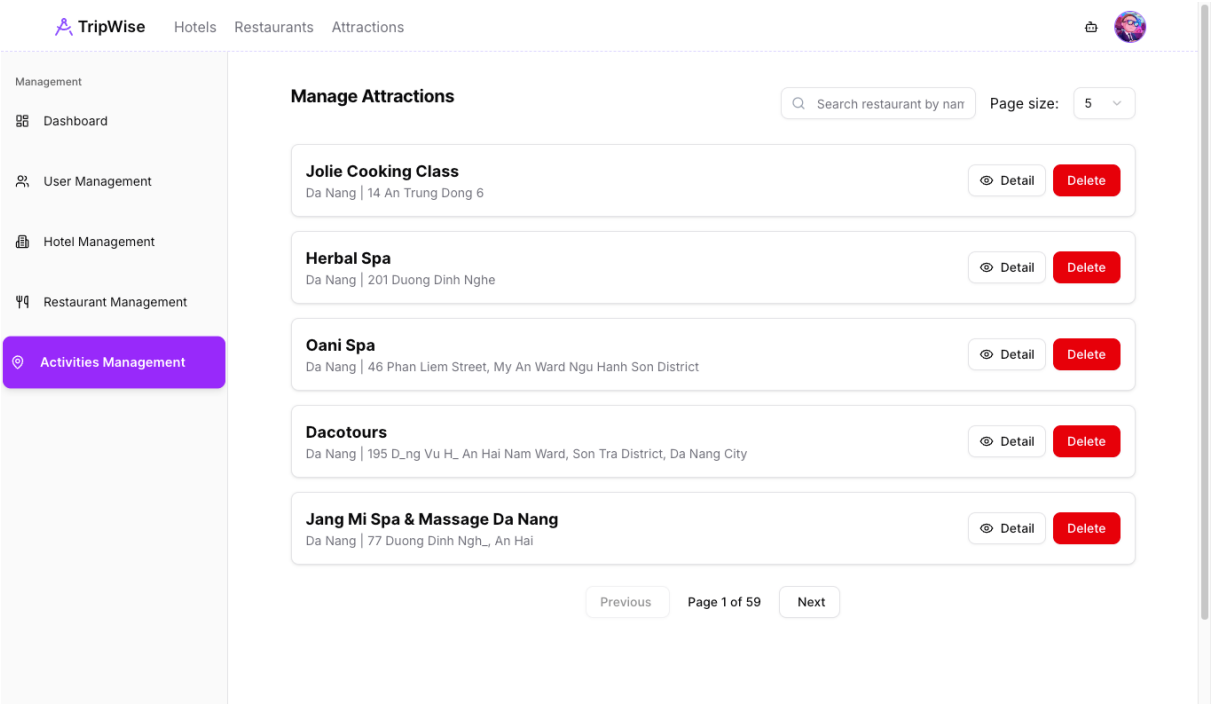

Figure 3.20 Manage attractions Screen

The sign-in and sign-up pages for guests registering as system users are illustrated in Figures 3.21 and 3.22.

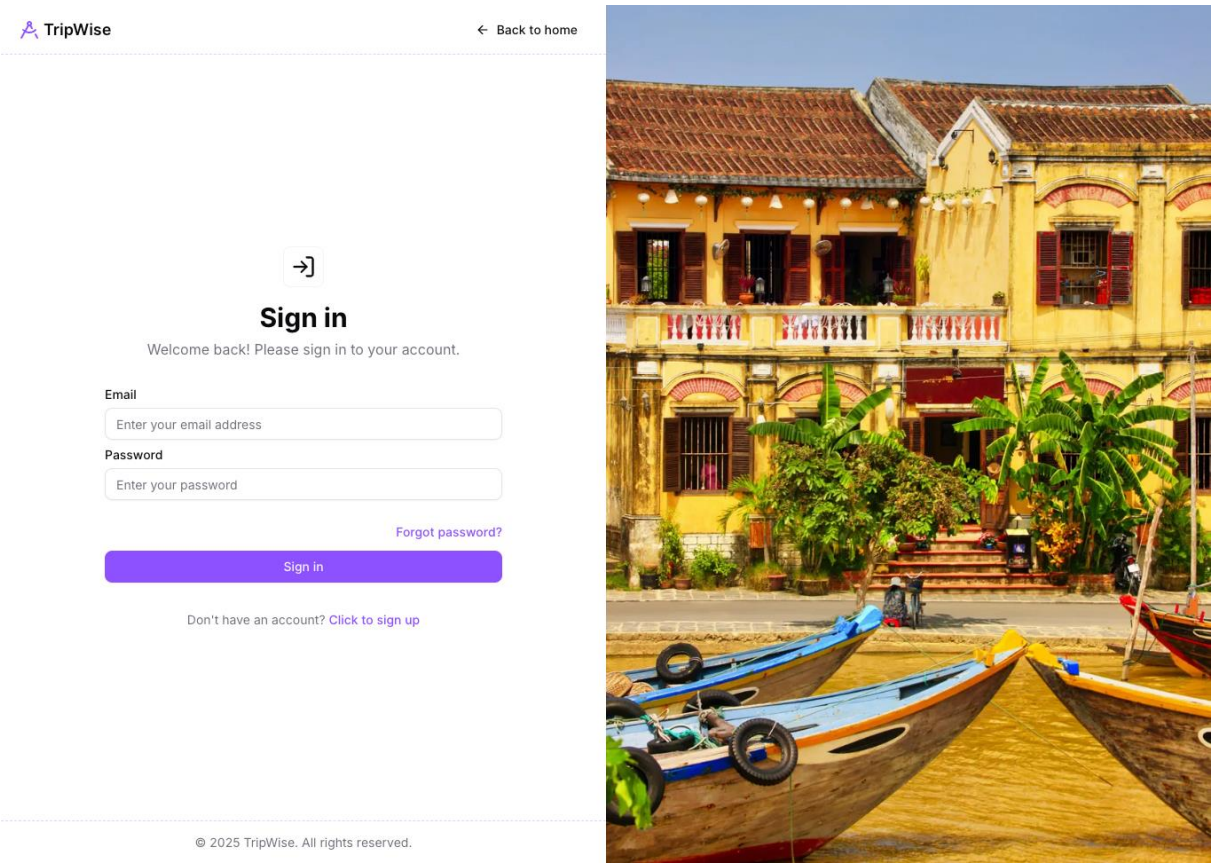

Figure 3.21 Sign in Screen

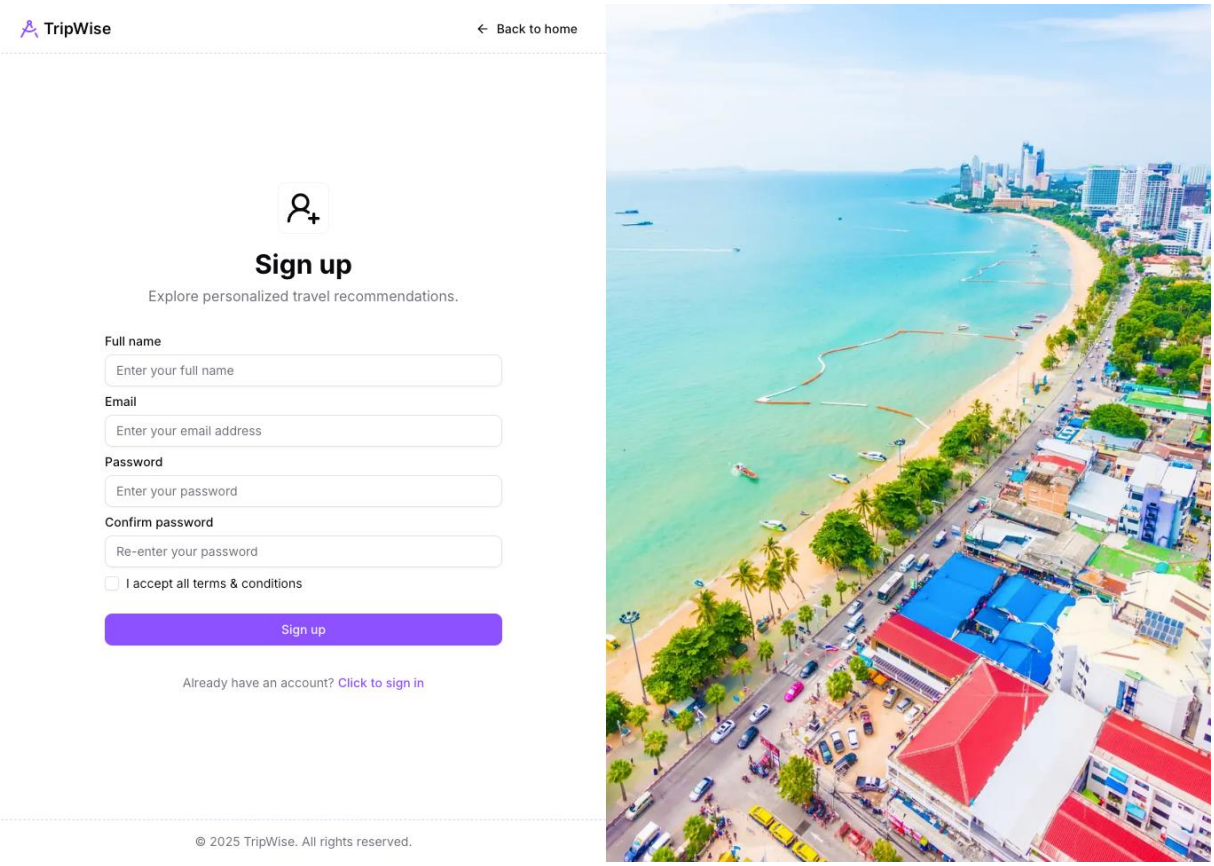

Figure 3.22 Sign up Screen

3.2. Evaluation

3.2.1. Advantages

The application demonstrates a well-rounded and thoughtful approach to solving the problem of personalized travel recommendation and route optimization. Key strengths include:

- Personalized recommendations: The system effectively recommends places to users based on their preferences, which significantly enhances user satisfaction and relevance.
- Review management system: The ability for users to create, update, delete, and read reviews ensures high interactivity and community engagement within the application.
- Administrative oversight: An administrator dashboard provides comprehensive system oversight, which is essential for managing and maintaining the quality of the service.
- User interface design: The UI is visually appealing and user-friendly, contributing to a positive user experience and prolonged user engagement.
- Efficient trip optimization: The route optimization algorithm delivers results within acceptable timeframes and successfully handles over 80% of test cases, indicating solid performance under typical conditions.

3.2.2. Disadvantages

Despite its strengths, the application faces some limitations that stem mainly from resource constraints and technical challenges:

- Outdated map API: Due to budget limitations, the current mapping service lacks up-to-date routing information, which can affect the accuracy and reliability of route suggestions.
- Performance bottlenecks with Neo4j: The interaction between Flask and the Neo4j database exhibits slower response times compared to PostgreSQL, which could impact scalability and user experience.
- Missing automated data collection: The absence of an auto-triggered cronjob for crawling place information reduces the system's ability to stay current with new or changing data.
- Limited knowledge base: With only around 800 places in the database, the chatbot sometimes lacks sufficient information to provide knowledgeable and diverse recommendations.

3.2.3. Conclusion

Overall, the application represents a commendable effort with practical functionality, a strong focus on user needs, and efficient route optimization capabilities. While there are areas for technical improvement, particularly in database interaction and data freshness, the system serves as a solid foundation for further development and real-world deployment. The work shows a good balance between theoretical implementation and practical usability, making it a worthy candidate for thesis defense.

CONCLUSION

Outcomes

Through the development of this system, I have gained a thorough understanding of working with modern frameworks and libraries. This process has enabled me to grasp their underlying structures and apply them effectively within the project. Moreover, engaging with current technologies has allowed me to stay abreast of emerging trends, equipping me to make informed decisions about my future career path.

The application features a user-friendly interface to ensure accessibility and ease of use. It also includes distinct functionalities for both administrators and end-users, enhancing the system's overall utility and adaptability.

Throughout the course of this project, I have significantly enhanced a range of essential skills. These include improved English proficiency, independent learning and research capabilities, project planning, and effective presentation techniques. This experience has not only expanded my technical and academic competencies but also contributed to my personal and professional growth.

Future work

Building upon the limitations identified in the evaluation section, the future development of this project will focus on the following key areas:

- **Expanding functionality and enhancing user experience:** The application will be further developed by incorporating additional features to better support user needs. Efforts will also be directed toward improving the overall user experience, including refining the interface to make it more visually appealing and intuitive.
- **Conducting independent research and developing personalized recommendation models:** Future work will involve conducting independent data research to design and implement personalized recommendation models. These models aim to provide users with tailored suggestions - such as recommended locations - based on individual preferences and behaviors. This strategy seeks to reduce reliance on third-party services, enabling the application to deliver more customized and accurate planning solutions.

REFERENCES

[1] Microsoft, "Retrieval Augmented Generation (RAG) in Azure AI Search," Microsoft, 15 04 2025. [Online]. Available: https://learn.microsoft.com/en-us/azure/search/retrieval-augmented-generation-overview?tabs=docs.

[2] J. Wilson, "Ep 153: Knowledge Cutoff – What it is and why it matters for large language models," 28 11 2023. [Online]. Available: https://www.youreverydayai.com/knowledge-cutoff-what-it-is-and-why-it-matters-for-large-language-models/.

[3] "What is retrieval-augmented generation (RAG)?," 30 10 2024. [Online]. Available: https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-retrieval-augmented-generation-rag.

[4] Amazon, "What is RAG (Retrieval-Augmented Generation)?," [Online]. Available: https://aws.amazon.com/what-is/retrieval-augmented-generation/.

[5] J. Stihec, "Retrieval-Augmented Generation (RAG) Improves AI Content Relevance and Accuracy," 30 08 2024. [Online]. Available: https://shelf.io/blog/retrieval-augmented-generation-rag-improves-ai-content-relevance-and-accuracy/.

[6] "Retrieval Augmented Generation: Everything You Need to Know About RAG in AI," 24 10 2024. [Online]. Available: https://www.weka.io/learn/guide/ai-ml/retrieval-augmented-generation/.

[7] I. Belcic, "What is RAG (retrieval augmented generation)?," 21 10 2024. [Online]. Available: https://www.ibm.com/think/topics/retrieval-augmented-generation.

[8] S. M. Adrien Payong, "How to Choose the Right Vector Database for Your RAG Architecture," 04 12 2024. [Online]. Available: https://www.digitalocean.com/community/conceptual-articles/how-to-choose-the-right-vector-database.

[9] S. Aquino, "What is RAG: Understanding Retrieval-Augmented Generation," 19 03 2024. [Online]. Available: https://qdrant.tech/articles/what-is-rag-in-ai/.

[10] E. Dogan, "RAG, or Retrieval Augmented Generation: Revolutionizing AI in 2025," 13 03 2025. [Online]. Available: https://www.glean.com/blog/rag-retrieval-augmented-generation.

[11] Google, "Generate grounded answers with RAG," [Online]. Available: https://cloud.google.com/generative-ai-app-builder/docs/grounded-gen.

[12] "Build RAG with in-line citations," [Online]. Available: https://docs.llamaindex.ai/en/stable/examples/workflow/citation_query_engine/.

[13] Google, "What is Retrieval-Augmented Generation (RAG)?," [Online]. Available: https://cloud.google.com/use-cases/retrieval-augmented-generation?hl=en.

[14] "3 Power-Ups: RAG's Impact on Large Language Models," 14 03 2024. [Online]. Available: https://hyperight.com/3-power-ups-rag-impact-on-large-language-models.

[15] Amazon, "Retrieve data and generate AI responses with Amazon Bedrock Knowledge Bases," [Online]. Available: https://docs.aws.amazon.com/bedrock/latest/userguide/knowledge-base.html.

[16] Microsoft, "Augment large language models with retrieval-augmented generation or fine-tuning," 16 01 2025. [Online]. Available: https://learn.microsoft.com/en-us/azure/developer/ai/augment-llm-rag-fine-tuning.

[17] M. Hu, "No Free Lunch: Retrieval-Augmented Generation Undermines Fairness in LLMs, Even for Vigilant Users," 10 10 2024. [Online]. Available: https://arxiv.org/html/2410.07589v1.

[18] R. Bellman, DYNAMIC PROGRAMMING, 1957.

[19] R. Bellman, "The theory of dynamic programming," 1954.

[20] Rosen, K. H, Discrete Mathematics and Its Applications (8th ed.), 2019.

[21] Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J, The Traveling Salesman Problem: A Computational Study, 2006.

[22] Robusto, C. C, "The Cosine-Haversine Formula," 1957.

[23] C. Veness, "Calculate distance, bearing and more between Latitude/Longitude points," 2019. [Online]. Available: https://www.movable-type.co.uk/scripts/latlong.html.

[24] R. Bellman, "Dynamic programming treatment of the travelling salesman problem," *Journal of the ACM ,* pp. 61-63 , 1962.

[25] G. O.-T. Team, "OR-Tools User's Guide," Google , 2023. [Online]. Available: https://developers.google.com/optimization.

[26] "First Solution Strategies," Google , 2023. [Online]. Available: https://developers.google.com/optimization/routing/routing_options.

[27] "Traveling Salesman Problem," Google , 2023. [Online]. Available: https://developers.google.com/optimization/routing/tsp.

[28] "Hybrid Recommender Systems," Google, [Online]. Available: https://developers.google.com/machine-learning/recommendation.

[29] Adomavicius, G., & Tuzhilin, A, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions.," *IEEE Transactions on Knowledge and Data Engineering,* 2005.