Project Analysis Report

TED UNIVERSITY

Senior Project
Project Name:
NextRoute

The URL of the project web page:
https://m-aras.github.io/

TEAM MEMBERS:
Zeynep Tuba Acat
44728720420

Barış Kula
53530308108

Kıvanç Ersöz
19367334680

Mehmet Aras Altan
61810479028

SUPERVISOR:
Eren Ulu

JURY:
Fırat Akba
Emin Kuğu

Content

1.Introduction

1.1.Purpose of the System

Today’s young travelers are seeking more than traditional guided tours. They desire personalized and meaningful experiences tailored to their interests, whether in art, music, food, nature, or adventure. However, finding such activities, suitable accommodations, and local events can often be a complex and time-consuming process.

NextRoute aims to bridge this gap. By leveraging artificial intelligence (AI), the platform delivers highly personalized travel plans that align with the interests of young users. These plans include activity preferences, cultural events, and recommendations for local cuisine. Moreover, with a student verification system, it offers budget-friendly alternatives and discounts tailored for students, making travel more accessible.

The primary objectives of the system are:

- To create personalized travel itineraries based on users’ interests.
- To provide affordable travel options, including accommodations and dining.
- To increase travel accessibility by offering discounts and special deals for student users.
- To optimize travel experiences with real-time recommendations and updates.

In addition to serving individual users, NextRoute benefits hotels and businesses by allowing them to launch campaigns targeting young travelers and promote themselves through the platform. This creates a mutually beneficial ecosystem for both users and local businesses.

1.2.Design Goals

The design of NextRoute focuses on providing a seamless user experience and ensuring system efficiency through the following goals:

1.2.1.User Experience (UX):

- Optimize the user interface for quick and easy navigation on mobile devices.
- Offer features like filtering by interests, saving favorites, and managing travel plans.

1.2.2.Personalization:

- ○ Utilize an AI-powered recommendation engine to generate tailored travel itineraries based on user interests, past preferences, and travel dates.
- ○ Allow users to customize their activities, accommodations, and dining options.

1.2.3.Accessibility and Affordability:

- ○ Implement a student verification system to offer special discounts and campaigns exclusively for students.
- ○ Provide economical options for users with varying budgets.

1.2.4.Performance and Scalability:

- ○ Design an infrastructure capable of handling a growing user base and expanding to new geographical regions.
- ○ Optimize real-time updates and recommendation functionalities.

1.2.5.Security and Privacy:

- ○ Ensure compliance with GDPR standards to protect user data.
- ○ Encrypt user information and provide access only to authorized personnel.

1.3.Definitions, Acronyms, and Abbreviations

- **AI (Artificial Intelligence)**: Algorithms used to provide personalized recommendations tailored to user preferences.

- **NLP (Natural Language Processing)**: Technology for analyzing user inputs to understand their intents and needs.

- **GDPR (General Data Protection Regulation)**: An international data protection standard established by the European Union to safeguard user privacy.

- **Flutter**: An open-source software development kit for building mobile applications.

- **NextRoute**: The name of this project, aimed at offering personalized travel itineraries for young travelers.

- **Student Verification System**: A mechanism to verify users as students, granting them access to exclusive discounts and offers.

- **Real-Time Updates**: A feature that notifies users about new events, changes, or cancellations instantly.

1.4.Overview

NextRoute is a mobile application designed to simplify travel planning for young travelers by tailoring recommendations based on personal preferences. The platform provides users with the most suitable activities, accommodations, and restaurants for their travel dates and budgets. The core functionality of the system can be summarized as follows:

1.4.1.User Interaction:

- Users access the platform via a mobile application.
- They input their interests, travel dates, and budget preferences to receive tailored recommendations.
- Users can save, edit, or delete their favorite suggestions.

1.4.2.AI-Powered Recommendation Engine:

- Analyzes user inputs to generate personalized suggestions for activities, accommodations, and dining options.
- Leverages historical user data and event information to improve recommendation accuracy.

1.4.3.Real-Time Updates:

- Keeps users informed about changes, cancellations, or additions to selected events.
- Sends notifications for upcoming activities or last-minute opportunities.

1.4.4.Student Verification System:

- Verifies user identity through uploaded student documents.
- Grants access to student-only discounts and campaigns.

1.4.5.Collaboration with Local Businesses:

- Allows hotels, restaurants, and event organizers to promote their services directly to young travelers.
- Facilitates targeted campaigns and offers tailored to the platform's user base.

NextRoute not only simplifies travel planning for users but also serves as a powerful promotional tool for businesses. By fostering interaction between users and local service providers, it creates a mutually beneficial ecosystem.

2.Proposed Software Architecture

2.1.Overview

In this section, you should describe the general purpose and functionality of the tourism application. The app is designed to offer a personalized tourism experience for students, allowing them to receive information about destinations, hotels, and restaurants, and create custom itineraries. Users will interact with a chatbot to ask about travel destinations, accommodation, dining options, and receive recommendations. The app will be developed using Flutter, while Python and Rasa will be used for the chatbot functionality.

2.2.Subsystem Decomposition

To ensure a modular and scalable structure, the system is divided into the following subsystems:

2.2.1.Frontend (Mobile Application)

- **Description**: The frontend serves as the user interface through which users interact with the application. It allows users to search for destinations, accommodations, and dining options while generating personalized itineraries.
- **Key Technologies**:
  - **Flutter**: A UI toolkit that enables a responsive and consistent design across mobile platforms.
- **Functionalities**:
  - Search for destinations, hotels, and restaurants based on user preferences.
  - Display personalized travel itineraries and recommendations.
  - Enable users to manage itineraries by adding, editing, or deleting activities.
  - Provide offline access to previously saved data, ensuring usability without internet connectivity.
- **User Interaction**:
  - The frontend communicates with the backend via RESTful APIs to fetch and display data dynamically.

2.2.2. Backend (Server)

- **Description**: The backend is the core processing unit of the system. It manages data flow between the frontend, chatbot, and database while ensuring business logic execution.
- **Key Technologies**:
  - **Python**: Powers the backend server, managing data operations and user authentication.
- **Functionalities**:
  - Handle API requests from the frontend.
  - Process itinerary generation requests using AI-powered algorithms.
  - Serve as the intermediary between the chatbot and the database.
  - Manage user accounts, authentication, and authorization.

2.2.3.Chatbot (Python & Rasa)

- **Description**: The chatbot is the primary interface for user interaction. It uses natural language understanding (NLU) to interpret user queries and provides real-time recommendations or information.
- **Key Technologies**:
    - **Rasa**: An open-source framework for building AI chatbots with advanced NLU capabilities.
    - **Python**: Handles logic for integrating Rasa with the backend and database.
- **Functionalities**:
    - Understand user intents such as "search for a hotel in Paris" or "show me Italian restaurants."
    - Provide fallback suggestions when specific queries cannot be answered.
    - Integrate seamlessly with backend systems to fetch relevant data.

2.2.4.Database

- **Description**: The database is the system's central storage for user data, itineraries, and travel-related information (e.g., hotels, destinations, restaurants).
- **Key Technologies**:
    - **MySQL or PostgreSQL**: Structured databases optimized for fast querying and secure data handling.
- **Data Stored**:
    - User profiles and authentication data.
    - Saved itineraries and user preferences.
    - Travel-related data like destinations, accommodations, and restaurants.
- **Functionalities**:
    - Ensure fast retrieval of frequently accessed data using caching techniques.
    - Securely store user data with encryption.

2.3.Hardware/Software Mapping

The provided schema defines the structure and relationships of data entities in the system, and this forms the basis for mapping the software and hardware architecture. Here's how the system's components will interact with the hardware and software:

2.3.1.Users

- **Purpose**: Stores the core user data, including credentials, student status, and identifiers for user-specific operations.
- **Software Handling**:
  - **Frontend**: Users access their accounts via the Flutter mobile app.
  - **Backend**: User authentication and profile management are handled by Python, with secure data stored in the database.
  - **Database Table**: Users table holds the user details.
- **Hardware Mapping**:
  - User data is stored on the **Database Server**.
  - Authentication requests are processed on the **Application Server**.

2.3.2.Travel Plans

- **Purpose**: Maintains user-specific travel plans with dates and timestamps.
- **Software Handling**:
  - **Frontend**: Users create, view, and manage their plans via the app.
  - **Backend**: Handles CRUD (Create, Read, Update, Delete) operations for plans.
  - **Database Table**: TravelPlans stores user-specific plans.
- **Hardware Mapping**:
  - Managed by the **Application Server**, with data stored in the **Database Server** for fast retrieval.

2.3.3.Events

- **Purpose**: Stores information about events such as concerts, festivals, and local activities.
- **Software Handling**:
  - **Frontend**: Events are displayed based on user preferences and location.
  - **Backend**: Filters and retrieves events based on AI recommendations.
  - **Database Table**: Events contains all event details.
- **Hardware Mapping**:
  - Events are stored and retrieved from the **Database Server**.
  - Recommendations and filters are processed by the **AI Subsystem**.

2.3.4.Favorites

- **Purpose**: Tracks user-specific favorite events.

- **Software Handling**:
    - **Frontend**: Users can mark events as favorites for quick access.
    - **Backend**: Manages relationships between users and their favorite events.
    - **Database Table**: Favorites connects Users and Events.
- **Hardware Mapping**:
    - Queries for favorite events are handled by the **Application Server**.
    - Data is stored in the **Database Server**.

2.3.5.Reviews

- **Purpose**: Allows users to review hotels, restaurants, and other services.
- **Software Handling**:
    - **Frontend**: Users submit reviews through the app.
    - **Backend**: Stores and retrieves reviews for display.
    - **Database Table**: Reviews links users, businesses, and review data.
- **Hardware Mapping**:
    - Review data is stored in the **Database Server**.
    - Queries are managed by the **Application Server**.

2.3.6.Hotels & Restaurants

- **Purpose**: Provides details about accommodations and dining options.
- **Software Handling**:
    - **Frontend**: Displays hotels and restaurants filtered by user preferences.
    - **Backend**: Manages hotel/restaurant data and integrates with recommendations.
    - **Database Tables**: Hotels and Restaurants store respective details.
- **Hardware Mapping**:
    - Data is stored in the **Database Server**.
    - Recommendations and filters are processed by the **AI Subsystem**.

2.3.7.Hotel Services

- **Purpose**: Lists additional services offered by hotels, like WiFi or breakfast.
- **Software Handling**:
    - **Frontend**: Displays hotel-specific services.
    - **Backend**: Retrieves services linked to hotels.
    - **Database Table**: HotelServices links hotel services to Hotels.
- **Hardware Mapping**:
    - Queries for services are processed by the **Application Server**.
    - Data is stored in the **Database Server**.

2.3.8.Admins

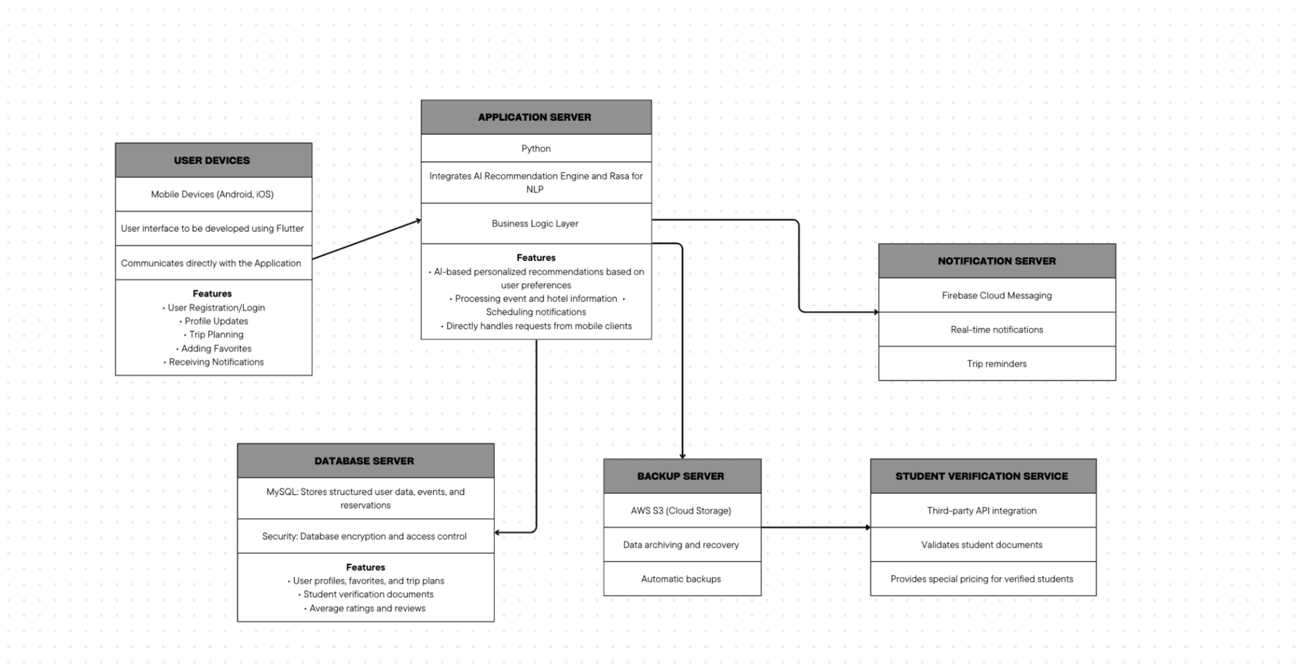
- **Purpose**: Administer and manage the platform.
- **Software Handling**:
  - **Frontend**: Admin panel for monitoring and management.
  - **Backend**: Admin authentication and task execution.
  - **Database Table**: Admins stores admin credentials.
- **Hardware Mapping**:
  - Admin-related operations are processed by the **Application Server**.
  - Credentials are stored securely in the **Database Server**.

2.3.9.AIRequests

- **Purpose**: Logs requests made to the AI recommendation engine.
- **Software Handling**:
  - **Backend**: Sends user preferences and retrieves recommendations from the AI system.
  - **Database Table**: AIRequests records all AI-related interactions.
- **Hardware Mapping**:
  - Requests are processed by the **AI Subsystem**.
  - Logs are stored in the **Database Server**.

2.3.10.AIErrors

- **Purpose**: Tracks errors encountered by the AI recommendation engine.
- **Software Handling**:
  - **Backend**: Logs error details for troubleshooting.
  - **Database Table**: AIErrors stores error logs linked to AIRequests.
- **Hardware Mapping**:
  - Error data is managed by the **AI Subsystem** and logged in the **Database Server**.


USER DEVICES
Mobile Devices (Android, iOS)
User interface to be developed using Flutter
Communicates directly with the Application
Features
• User Registration/Login
• Profile Updates
• Trip Planning
• Adding Favorites
• Receiving Notifications
APPLICATION SERVER
Python
Integrates AI Recommendation Engine and Rasa for NLP
Business Logic Layer
Features
• AI-based personalized recommendations based on user preferences
• Processing event and hotel information
• Scheduling notifications
• Directly handles requests from mobile clients
NOTIFICATION SERVER
Firebase Cloud Messaging
Real-time notifications
Trip reminders
DATABASE SERVER
MySQL: Stores structured user data, events, and reservations
Security: Database encryption and access control
Features
• User profiles, favorites, and trip plans
• Student verification documents
• Average ratings and reviews
BACKUP SERVER
AWS S3 (Cloud Storage)
Data archiving and recovery
Automatic backups
STUDENT VERIFICATION SERVICE
Third-party API integration
Validates student documents
Provides special pricing for verified students

2.4.Persistent Data Management

Persistent data management refers to how the system handles, stores, and retrieves data in a consistent and secure manner. In the NextRoute tourism application, persistent data encompasses user information, travel plans, events, hotels, restaurants, and system logs such as AI requests and errors. Below is an explanation of how persistent data is managed based on the provided schema:

1. Database Structure

The system relies on a relational database to store and manage all persistent data. The database schema is designed with relationships and constraints to ensure data integrity and accessibility.

- Database Technology:

  MySQL or PostgreSQL will be used due to their scalability, support for complex relationships, and robust data handling capabilities.

- Primary Tables:
  - Users: Stores core user information such as name, email, and student verification status.
  - TravelPlans: Maintains user-specific itineraries, including travel dates and names.
  - Events: Contains details about events (e.g., concerts, festivals) relevant to user interests.

    - Hotels and Restaurants: Stores data about accommodations and dining options.
    - AIRequests and AIErrors: Logs interactions with the AI system and any errors for debugging and analytics.

2. Data Relationships

The database schema uses foreign key constraints to maintain relationships between tables, ensuring referential integrity:

- TravelPlans: Linked to Users via user_id, allowing retrieval of plans specific to each user.
- Favorites: Establishes a many-to-many relationship between Users and Events, storing favorite events for each user.
- Reviews: Links Users to specific businesses (e.g., hotels or restaurants), enabling user-generated ratings and comments.
- AIRequests: Connects to Users to log requests and track user-specific AI interactions.
- AIErrors: Linked to AIRequests to store error details for specific requests.

3. Data Storage Strategy

The application uses structured tables with indexes to ensure fast and reliable access to data.

- Indexing:
    - Primary keys (e.g., user_id, plan_id, event_id) are indexed for efficient querying.
    - Frequently searched fields, such as email in Users and location in Hotels and Restaurants, are indexed for quick lookups.
- Caching:
    - Frequently accessed data, such as popular events or hotels, is cached at the backend level to reduce database load and improve performance.
    - User-specific cached data (e.g., saved itineraries) ensures offline accessibility.
- Data Partitioning:
    - Large datasets, such as event or hotel listings, are partitioned based on geographical regions or categories (e.g., "concerts" or "low-budget hotels") for faster retrieval.

4. Data Security

The application employs multiple layers of security to protect persistent data:

- Encryption:

    - Sensitive data, such as user passwords, is hashed using secure algorithms (e.g., bcrypt).
    - All data in transit between the frontend, backend, and database is encrypted using SSL/TLS protocols.
- Access Control:
    - Role-based access control (RBAC) ensures that only authorized users can access or modify specific data. For example:
    - Users can only view and manage their own travel plans.
    - Admins have elevated privileges to manage global data.
- Audit Trails:
    - Modifications to critical tables (e.g., Users, TravelPlans) are logged for tracking and auditing purposes.

5. Backup and Recovery

To ensure data persistence even in the event of system failures, a robust backup and recovery strategy is implemented:

- Automated Backups:
    - Full backups are taken daily, and incremental backups are performed hourly.
    - Backups are stored securely in cloud storage solutions such as AWS S3 or equivalent.
- Recovery Plan:
    - Backup files are versioned to facilitate easy restoration.
    - Disaster recovery procedures ensure minimal downtime and data loss.

6. Scalability

The data management system is designed to handle a growing user base and expanding datasets:

- Horizontal Scaling:
    - The database can be distributed across multiple servers to handle increased traffic.
- Load Balancing:
    - Queries are distributed among servers to prevent bottlenecks.
- Read Replicas:
    - Read-heavy operations, such as retrieving event or hotel data, are routed to read replicas to improve performance.

7. Data Retention Policy

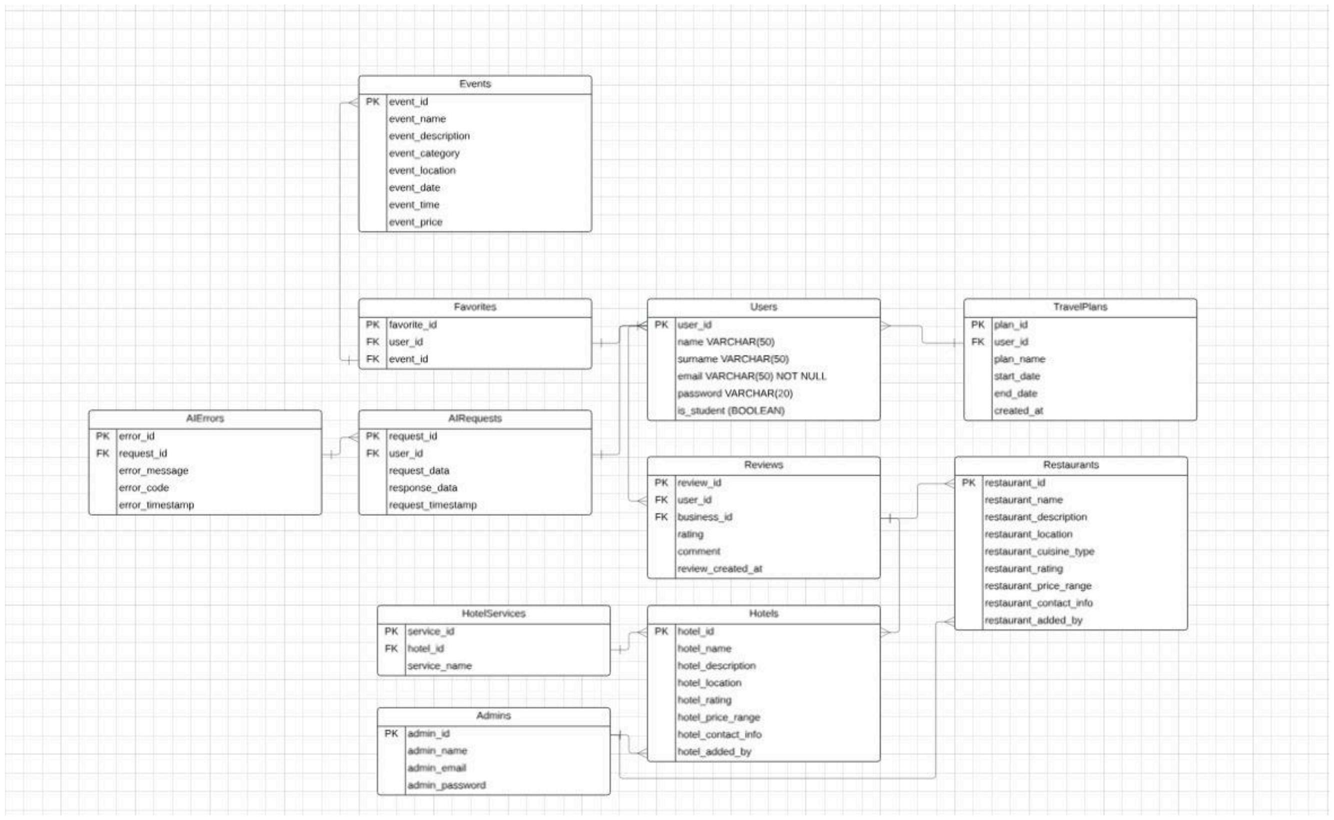
The system adopts a retention policy to optimize storage and comply with legal requirements:

- User Data:
    - Retained as long as the user account is active.
    - Deleted permanently upon user request, in compliance with GDPR.
- AIRequests and AIErrors:
    - Retained for 12 months for analytical purposes and troubleshooting.
- TravelPlans:
    - Retained for two years after completion for user reference and reactivation.

8. High Availability

The database is configured for high availability to ensure uninterrupted service:

- Replication:
    - Data is replicated across multiple servers to provide redundancy.
- Failover Mechanism:
    - In case of a server failure, a standby replica takes over seamlessly.
- Monitoring:
    - Database health is continuously monitored, and alerts are triggered for anomalies.
    - By adopting these strategies, the system ensures that persistent data is managed efficiently, securely, and reliably, supporting both the scalability and robustness required for NextRoute's functionality. Let me know if additional details are needed!



2.5.Access Control and Security

Ensuring secure and authorized access to the system is a critical aspect of its design. The following mechanisms will be implemented:

2.5.1.Authentication:

- Users log in with their verified student email addresses.
- A secure authentication process ensures that only authorized users access the system.

2.5.2.Authorization:

- Role-based access control (RBAC) limits access to specific features.
- For example, users can only view and edit their itineraries.

2.5.3.Data Security:

- User data will be encrypted during transmission using **SSL/TLS** protocols.
- Sensitive information, such as authentication credentials, will be hashed before storage.

2.5.4.Input Validation:

- All user inputs will be validated at both the frontend and backend levels to prevent invalid or malicious data from being processed.

2.6.Global Software Control

This section explains how the application will maintain operational efficiency:

1. **Data Synchronization**:
    - Users can access their itineraries on multiple devices without any loss of information.
    - Changes to data are synchronized in real time.
2. **Version Control**:
    - Updates to the Flutter frontend and Python backend will be managed via **GitHub**.
    - Version control allows for seamless deployment of new features and bug fixes.
3. **Error Management**:
    - Comprehensive error logging and monitoring will be implemented.
    - User feedback mechanisms will help identify and resolve recurring issues.

2.7.Boundary Conditions

The boundaries of the system define its interactions and limitations:

1. **Mobile-Only Application**:
    - The app is designed exclusively for mobile platforms (Android and iOS).
    - There is no web-based version of the system.
2. **Chatbot Interaction**:
    - The chatbot interacts directly with users to answer queries and provide recommendations.
    - External systems, such as APIs for fetching hotel or restaurant data, are accessed through the backend.
3. **Database Integration**:
    - All travel-related data and user information will be stored in the database.
    - The database can be accessed only through secure backend processes.

2.8 Handling Common Issues

1. Internet Connectivity Failure

- **Solution**:
    - Display user-friendly error messages like "No internet connection. Please try again later."
    - Provide offline access to previously saved data and itineraries.
    - Implement retry mechanisms for failed requests once connectivity is restored.

2. Server Unavailability

- **Solution**:
    - Show messages like "We’re having trouble connecting to our servers."
    - Cache frequently accessed data locally for offline access.
    - Queue user actions that require server interaction and process them once the server is available.

3. API Failures

- **Solution**:
    - Implement fallback mechanisms to provide alternative suggestions if APIs fail.
    - Log errors and retry requests periodically.

4. User Input Validation

- **Solution**:
    - Validate inputs like email addresses and search queries before processing.
    - Display clear error messages for invalid inputs.

5. Device Compatibility

- **Solution**:
    - Conduct extensive cross-platform testing to ensure compatibility with various devices and screen sizes.
    - Use Flutter's responsive design features for consistent UI behavior.

3. Subsystem Services

The NextRoute application consists of several interconnected subsystems, each with specific services designed to manage and optimize the system's functionality. Below is a detailed breakdown of the services provided by each subsystem:

3.1 Frontend Services

- **User Authentication and Profile Management**:
  - Allows users to log in, register, and manage their profiles.
  - Verifies student status during registration using a student email address.
- **Search and Filtering**:
  - Provides a search interface for destinations, hotels, and restaurants with filters (e.g., price, rating, category).
- **Itinerary Management**:
  - Enables users to create, edit, and save custom travel plans.
  - Offers offline access to saved itineraries.
- **Interactive Chatbot Interface**:
  - Provides a conversational interface for users to ask questions and receive recommendations.
- **Responsive UI**:
  - Adapts seamlessly to various mobile devices and screen sizes.

3.2 Backend Services

- **Data Processing and Management**:
  - Handles CRUD (Create, Read, Update, Delete) operations for user profiles, itineraries, and reviews.
  - Processes AI recommendation requests and returns results to the frontend.
- **API Integration**:
  - Manages communication between the frontend, database, and chatbot.
  - Retrieves external data for hotels, events, and restaurants as needed.
- **Caching and Optimization**:
  - Implements caching mechanisms to improve performance for frequently accessed data.
- **Security**:
  - Validates and encrypts user data to ensure privacy.
  - Implements role-based access control for secure interactions.

3.3 Chatbot Services

- **Natural Language Understanding (NLU)**:
  - Processes user inputs to determine intents and extract entities (e.g., "Find hotels in Paris").
- **Recommendation Service**:
  - Provides tailored suggestions for destinations, accommodations, and dining based on user preferences.
- **Fallback Mechanism**:
  - Offers alternative responses when user queries cannot be processed.

- **Context Management**:
    - Maintains the context of user conversations for seamless interactions.

3.4 Database Services

- **Data Storage and Retrieval**:
    - Stores all persistent data, including users, itineraries, events, hotels, and reviews.
    - Optimized for quick queries using indexing and partitioning.
- **Data Security**:
    - Encrypts sensitive data (e.g., passwords) and ensures secure access.
- **Analytics and Logs**:
    - Records AI requests, errors, and system logs for performance monitoring and debugging.

3.5 AI Services

- **Recommendation Engine**:
    - Analyzes user preferences and generates personalized travel itineraries.
- **Error Logging**:
    - Captures and stores errors encountered during AI operations for future troubleshooting.

4.Glossary

1. **AI (Artificial Intelligence)**: Technology used for analyzing user preferences and generating personalized recommendations.
2. **NLP (Natural Language Processing)**: A subset of AI that processes and interprets user inputs in natural language.
3. **CRUD**: Create, Read, Update, Delete – the primary operations for managing database records.
4. **SSL/TLS**: Secure Sockets Layer / Transport Layer Security – protocols for encrypting data transmitted over the internet.
5. **Frontend**: The user interface (UI) of the application, developed using Flutter for mobile platforms.
6. **Backend**: The server-side component that manages business logic and data processing, developed with Python.
7. **Chatbot**: An AI-powered conversational agent built using Python and Rasa to interact with users.
8. **Caching**: A mechanism for storing frequently accessed data temporarily to improve performance.
9. **Database**: The structured repository for storing persistent data such as user profiles, travel plans, and event details.

10. **Student Verification System**: A feature that authenticates users as students using their email addresses.
11. **Role-Based Access Control (RBAC)**: A method of restricting system access based on user roles.

5.References

1. **Flutter Documentation**: Flutter development resources for building responsive mobile applications.
    - URL: https://flutter.dev/docs
2. **Rasa Documentation**: Comprehensive guide for building AI-powered chatbots with Rasa.
    - URL: https://rasa.com/docs
3. **MySQL Documentation**: Reference for designing and managing relational databases.
    - URL: https://dev.mysql.com/doc/
4. **GDPR Guidelines**: Regulations for data privacy and security in compliance with European standards.
    - URL: https://gdpr-info.eu/
5. **Python Official Documentation**: Resources for backend development and integration.
    - URL: https://docs.python.org/3/
6. **AWS Backup and Security Documentation**: Best practices for data backup and secure storage.
    - URL: https://aws.amazon.com/backup/
7. **Stack Overflow**: Community-driven resource for troubleshooting development issues.
    - URL: https://stackoverflow.com/