SDD – Detailed Design

1. Introduction

This document describes the detailed design for a Travel and Social Networking Application aimed at improving outdoor experiences.
The system allows regional councils and local businesses to upload and manage routes, attractions, and local information while providing users with real-time updates, weather alerts, and dynamic connections with other travelers.
Additionally, the application enables Nature Authority representatives to receive trail condition reports from users and provide relevant updates when necessary.
This document details the system's components, functionalities, user interactions, data flow, and integrations with external services such as Google Maps and weather APIs.

2. System Overview

The system will consist of three main components: the User Interface (UI), the Backend API, and the Database.

User Interface (UI): This component will be responsible for allowing users to interact with the app.
Users will be able to log in, register, set preferences, view personalized tour routes, access maps, view recommendations, and receive advertisements related to the area they are visiting.
The UI will also allow users to interact with other users, share experiences, and get updates on their routes.

Backend API: The Backend API will handle all the data processing and communication between the frontend (UI) and the database.
It will process user inputs, fetch real-time data (such as weather updates, available attractions, and route information), and generate personalized route suggestions based on user preferences.
The Backend API will also facilitate communication with local authorities to update route maps and advertisements.

Database: The Database will store all essential data, including user profiles, preferences, activity logs, personalized route suggestions, map data, and advertisements. It will also store data provided by local authorities, such as tourist attractions, points of interest, and available services.
Additionally, the app will read from the database for relevant local attractions and advertisements tailored to specific areas.
The system will analyze the proximity of these attractions along a user's route and determine which advertisements to display based on the user's location and the

points of interest they are approaching. The database will be designed to handle large amounts of data, ensuring quick access and scalability as the system grows.

These components will work together to provide users with an intuitive and dynamic experience, ensuring they receive accurate, personalized, and real-time information while exploring new areas.
The integration of local authority data and real-time updates will allow the system to offer tailored experiences, including navigational assistance and relevant advertisements.

3. Design Considerations

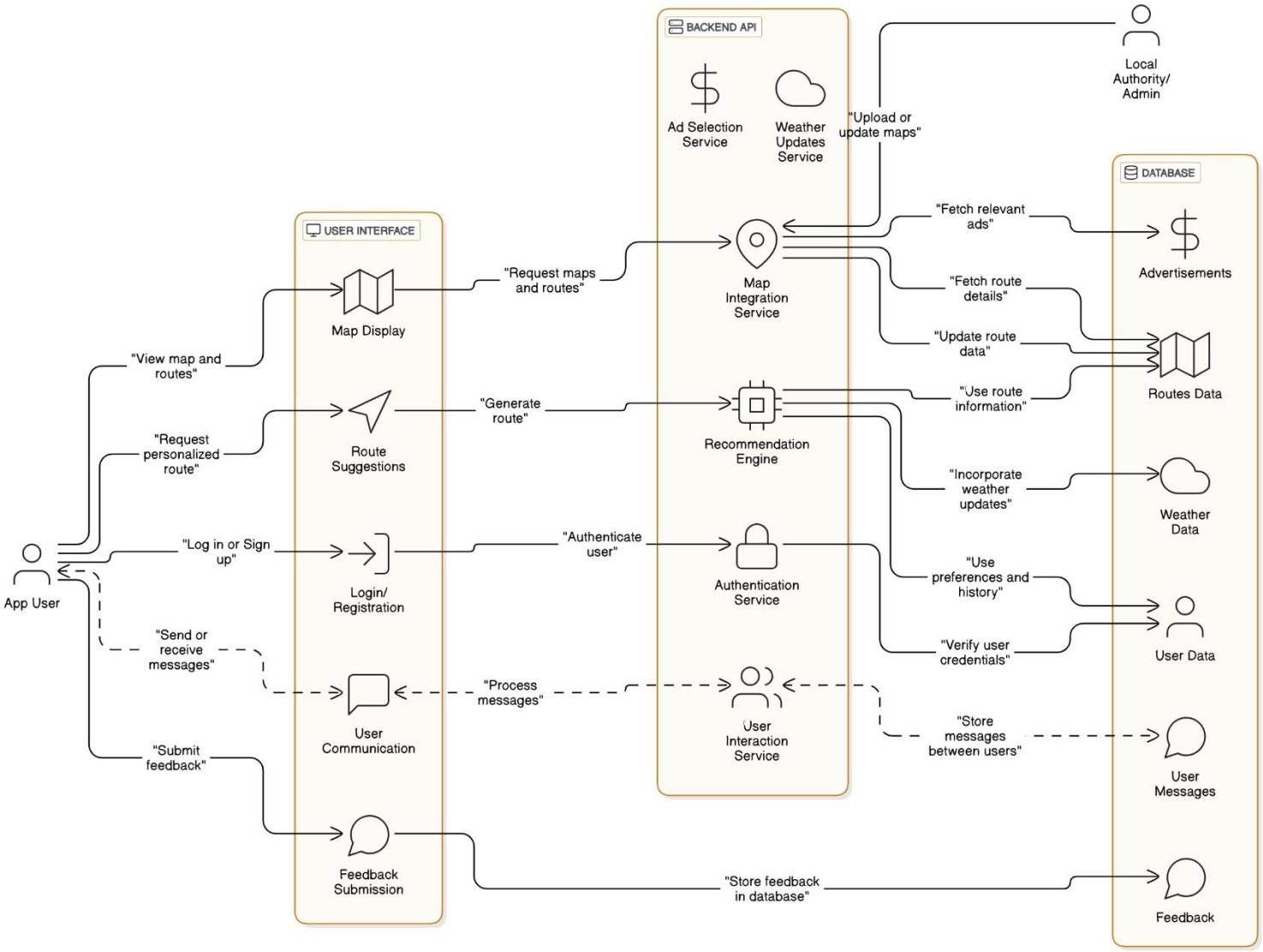
The system assumes that users will have a continuous internet connection, enabling real-time updates and location-based services.

Additionally, the system assumes that local councils will provide the data in an organized and accurate manner, ensuring the system can operate smoothly.

Users will have compatible devices (smartphones with GPS capabilities) to ensure that features such as navigation and personalized recommendations work correctly.

The app will be developed using Android Studio in Java, with communication between the app and the server utilizing protocols like JSON for data transfer and APIs for connection and information exchange.

4. System Architecture



5. Component Design

User Interface Module

- Purpose: This module provides the screens for users to perform various actions such as login or registration, displaying maps, showing route recommendations, submitting feedback, and interacting with other users.
- Input: User details (e.g., username and password), incoming data such as route recommendation requests based on categories or feedback submissions.
- Output: Access token (after successful login), displaying the map, personalized route recommendations, stored feedback.
- Dependencies: Depends on Backend API services such as Authentication Service, Recommendation Engine, Feedback Submission Service, and User Interaction Service.

Authentication API

- Purpose: This service is responsible for authenticating users and generating tokens for system access.
- Input: Username, password.
- Output: Token.
- Dependencies: Depends on the database for verifying user details (User Data).

Recommendation Engine

- Purpose: The recommendation engine is responsible for calculating personalized route recommendations based on user preferences, weather conditions, trail condition reports, and attraction availability.
- Input: User preferences, route history, weather data, and trail condition reports.
- Output: Personalized route recommendations.
- Dependencies: Depends on user data, route data, weather data and field reports.

Map Integration Service

- Purpose: This service is responsible for displaying maps and overlaying the routes on the maps.
- Input: Councils input route data into the system's database, requests to display maps or routes.
- Output: A map of the area with displayed routes.
- Dependencies: Depends on route data, advertisement data, and the recommendation engine.

Weather Updates Service

- Purpose: This service is responsible for updating real-time weather information.
- Input: Requests for weather information from the API interface.
- Output: Updated weather conditions for a specific area.
- Dependencies: Depends on data about weather limitations and conditions in the route area, based on the forecast and field reports.

User Interaction Service

- Purpose: This service is responsible for managing communication between users, including sending messages and responses.
- Input: Messages from users.
- Output: Responses and messages sent to other users.
- Dependencies: Depends on user data and the database for user messages.

Database Module

- Purpose: This module is responsible for storing user data, route data, weather data, advertisement data, user communications, and feedback.
- Input: User data, routes, advertisements, feedback, messages.
- Output: Storing data in an organized and accessible manner for all modules.
- Dependencies: Depends on modules like Authentication API, Recommendation Engine, and User Interaction Service.

6. Data Design

Users Table

- Columns:
    - user_id (Primary Key, int)
    - username (varchar)
    - password_hash (varchar)
    - role (varchar, user role in the system)

Routes Table

- Columns:
    - route_id (Primary Key, int)
    - start_point (varchar)
    - end_point (varchar)
    - length (float)
    - difficulty (varchar)
    - description (text)

Weather Table

- Columns:
    - weather_id (Primary Key, int)
    - route_id (Foreign Key to Routes Table)
    - temperature (float)
    - humidity (float)
    - precipitation (float)
    - wind_speed (float)
    - date (datetime)

User Messages Table

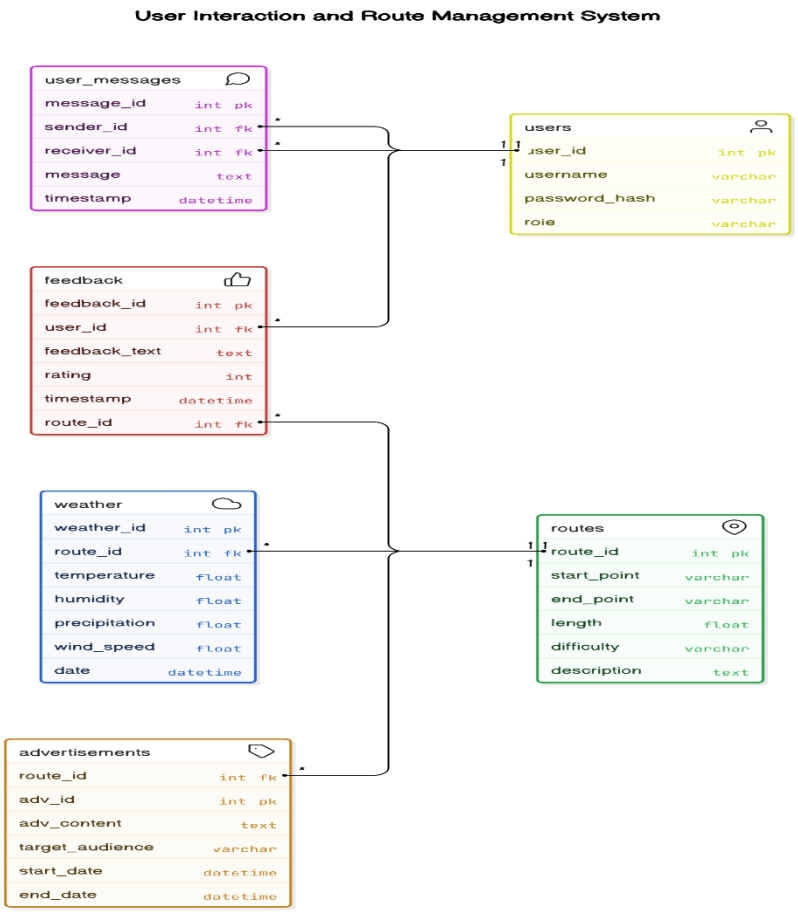
- Columns:
    - message_id (Primary Key, int)
    - sender_id (Foreign Key to Users Table)
    - receiver_id (Foreign Key to Users Table)
    - message (text)
    - timestamp (datetime)

Advertisements Table

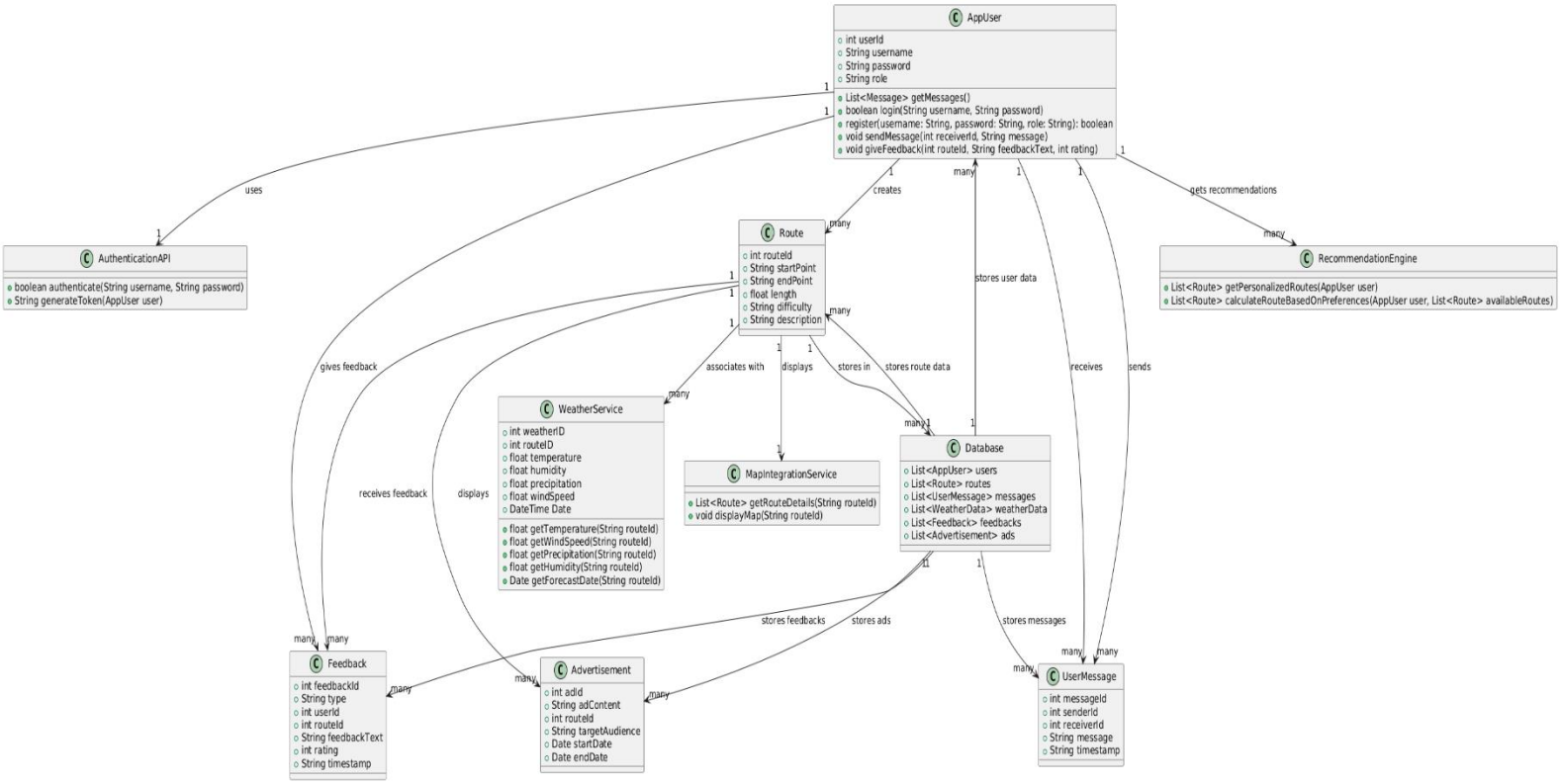
- Columns:
    - adv_id (Primary Key, int)
    - adv_content (text)
    - route_id (Foreign Key to Routes Table)
    - target_audience (varchar)
    - start_date (datetime)
    - end_date (datetime)

Feedback Table

- Columns:
    - feedback_id (Primary Key, int)
    - user_id (Foreign Key to Users Table)
    - route_id (Foreign Key to Routes Table)
    - feedback_text (text)
    - rating (int)
    - timestamp (datetime)



7. Detailed Class/Function Design



8. User Interface Design

1. Login/Register Screen:

Fields:

Username: Field for entering the username.

Password: Field for entering the password.

Actions:

Login button: A "Login" button that performs the login action after the user enters the username and password.

Forgot password link: A link to reset the password in case it is forgotten.

Sign up link: A link that leads to the registration screen if the user is new and wants to register.

Navigation:

If login is successful (the user exists and the password is correct), the user will be directed to the Home Screen.

If login fails, an error message will be displayed.

If the user is not registered, they can proceed to the registration screen.

2. Registration Screen:

Fields:

Personal Details:

- **Name** (Full Name)
- **Email**
- **Date of Birth**
- **Gender**

Preferences:

- **Types of Trails**: The user can choose the types of trails they prefer (mountains, forests, trails with water sources, nature, culture, archaeology, etc.).
- **Difficulty Level**: Different difficulty levels (easy, medium, hard).

Actions:

Register Button: A button to submit the registration details and preferences.

Validation:

All fields must be filled out.

The email address must be valid.

The user must select at least one trail type.

Navigation:

After the user successfully completes the registration, they will be directed to the Login Screen, where they can log in with the username and password they entered.

3. Home Screen:

Features:

Navigation Options: Use a bottom navigation bar that allows navigation between different sections of the app (such as trail search, displaying location on the map, reading feedback). Each section will open a different fragment.

Live Navigation: Option to navigate in real-time and use maps to find the closest trail based on the user's current location.

Find People Nearby: Option to search for people nearby – users will be able to find friends/other users and get updates on shared trail suggestions or trail conditions based on location.

Search Bar: A search bar that allows the user to search for trails by names or categories.

Profile Management: Option to manage and edit the personal profile.

Navigation:

Clicking on "Live Navigation" will take the user to the map screen, where all the closest or most requested trails based on their current location will be displayed.

Clicking on "Find People Nearby" will show a list of nearby users with the option to send them messages or join shared trails.

Clicking on "Search" will lead to a search screen with trail filters such as type, difficulty, length, etc.