Architecture Document

1. Application

    1. **Microservices**

        1. The architecture follows a **microservices approach**, separating functionality into independent services that communicate over APIs.
        2. Key microservices:
            - **Recommendation Service**: Generates product recommendations using the deep learning model.
            - **User Service**: Manages user data, including location and interaction history.
            - **Product Service**: Manages product catalog, including price, category, and product images.
            - **Location Service**: Handles geolocation services to determine proximity of products based on user latitude and longitude.

    2. **Event-Driven**

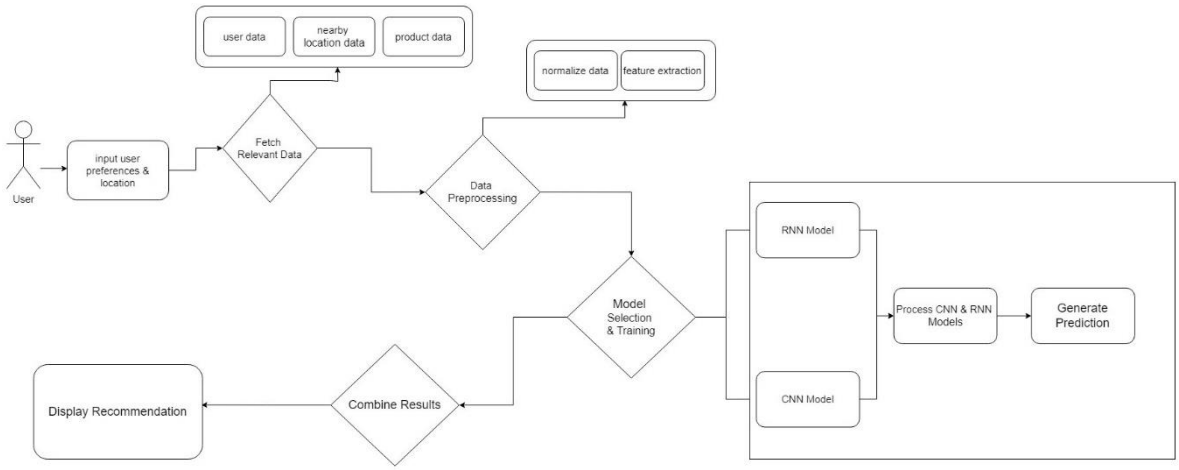
        1. The system can be **event-driven**, where real-time events such as a user viewing a product or changing location trigger the recommendation service.
        2. **Message Queues (e.g., Kafka)**: Used for passing real-time data between services.
            - When a user interacts with the app (e.g., by viewing a product), an event is triggered and passed to the recommendation service.

3. Serverless

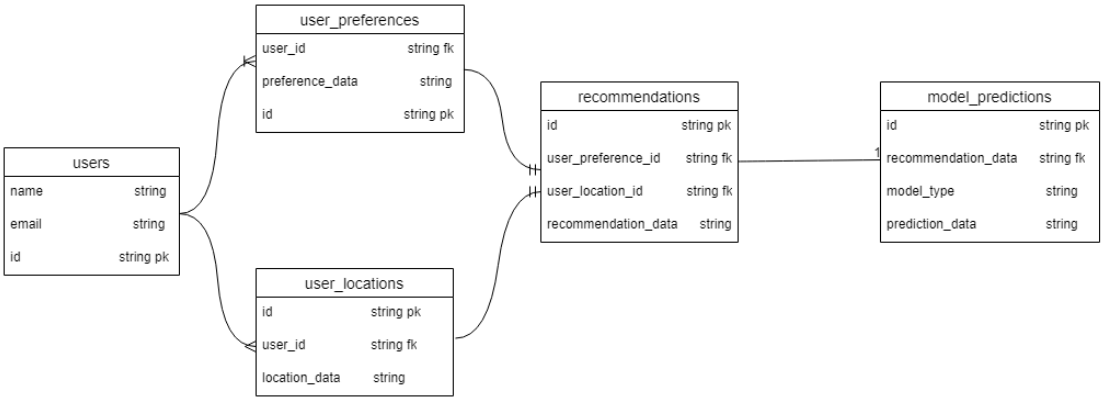
1. The recommendation system might use serverless components (e.g., AWS Lambda) for certain microservices.
    - Inference Microservice: Deploying the recommendation model as a serverless function to scale automatically with the number of requests.

2. Database

1. ER Diagram



2. Schema Design



3. Data Exchange Contract

1. Frequency of data exchanges

1. Real-time for recommendation generation:
   - User location updates trigger new recommendations.
   - Product interaction data is fed into the recommendation model after every interaction (e.g., view or purchase).
2. Batch Updates: For recalibrating the model using accumulated historical data.

2. Data Sets

1. User Data: Includes user profile details, interaction history, and geolocation (latitude and longitude).
2. Product Data: Information on product attributes such as category, price and discount
3. Interaction Data: Total interactions (clicks, views, purchases), along with timestamps.
4. Location Data: City, latitude, and longitude for calculating distance_to_user.

3. Mode of Exchanges (API, File, Queue etc.,)

1. API-Based Communication:
   - Microservices will communicate via REST APIs (e.g., for location updates or fetching product details).
2. Message Queue:
   - Event-driven communication using queues like Kafka to manage real-time interactions and data streaming between services.