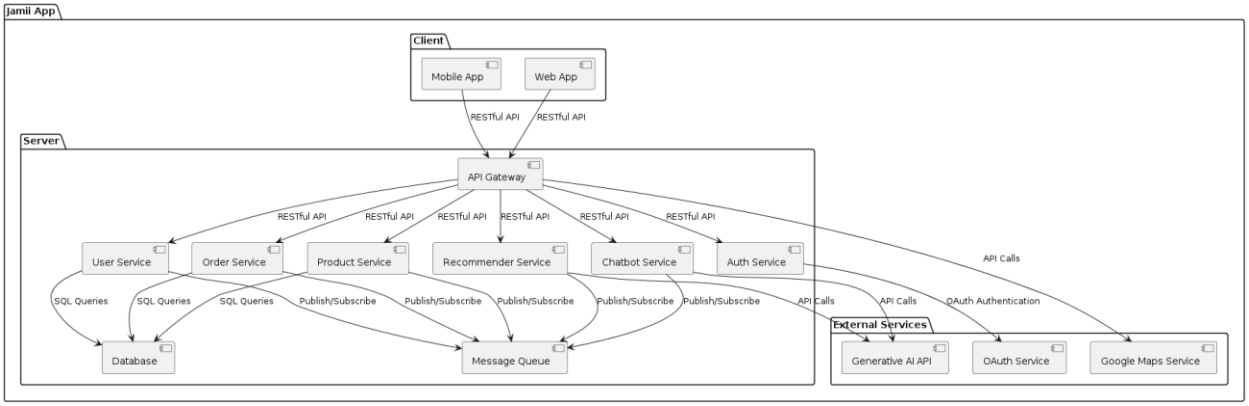
21Summer

Jamii app

Coordination model

Overview

The Jamii app will leverage several architectural and design patterns to create a robust, scalable, and maintainable system. This document outlines the coordination model, specifying the interfaces and patterns that will be utilized to achieve seamless integration and functionality.



System Interfaces

The system will interface with the following services:

1. **OAuth**: For secure authentication and authorization of users.
2. **Google Maps Services**: To provide location-based services and map functionalities.
3. **Generative AI APIs**: To enhance AI-powered features such as product recommendations and customer support.

Architectural and Design Patterns

1. Client-Server Architectural Pattern

The system will be built on a client-server architecture, where:

- **Client**: The client-side application (mobile or web) that interacts with the users.
- **Server**: The backend server that processes requests, handles business logic, and communicates with external services.

Communication Protocol: The client and server will communicate using RESTful APIs, ensuring a stateless and scalable interaction model.

3. Publish-Subscribe Pattern

The publish-subscribe pattern will be used to manage updates between the views and models efficiently. This will enable:

- **Publishers**: Components (e.g., models) that generate events.
- **Subscribers**: Components (e.g., views) that react to those events.

This pattern will decouple the components, allowing for more modular and maintainable code.

4. Pipe and Filter Pattern

The pipe and filter pattern will be employed for the recommender system and generative AI capabilities:

- **Filters**: Individual processing steps (e.g., data transformation, filtering, and AI processing).
- **Pipes**: The channels through which data flows between filters.

This pattern allows for flexible and reusable processing steps, enhancing the scalability and extensibility of the system.

5. Shared Data Store

A shared data store will be used to manage persistent data across the system. This will include:

- **Relational Database**: For structured data such as user profiles, product listings, and transactions.
- **NoSQL Database**: For unstructured data such as user activity logs and AI-generated content.

The shared data store will ensure data consistency and availability across the system.

Summary

The Jamii app will utilize a combination of architectural and design patterns to ensure a scalable, maintainable, and efficient system. By leveraging client-server architecture, MVC, publish-subscribe, pipe and filter, and shared data store patterns, along with integrating OAuth, Google Maps services, and generative AI APIs, the system will provide a seamless and robust user experience tailored to the needs of the Kenyan market.