**Digital Chit Fund Management**

**Team Members:**

1. **Tuljesh Rahul Anajipuram**
2. **Rohitha Reddy Gantla**

**Problem Statement**: Traditional savings schemes in certain communities, like collective savings and borrowing groups, suffer from a lack of transparency and organization. Participants often have little to no visibility into the group’s financial status, and organizers find it hard to manage multiple groups. The community is in need of a secure platform that can automate the management of participants, handle fund allocations, and ensure clear and transparent processes for all involved.

1. **Understanding of the problem statement**

1. **Explanation of the problem context:**

Many traditional community savings and borrowing groups rely on manual, paper-based systems, which leads to issues such as:

* **Lack of transparency:** Members are not able to track the transactions and how the money is distributed.
* **Poor organization:** Group organizers often struggle with managing multiple groups and participants.
* **Fund allocations:** Disagreements may arise over fund allocation, particularly if certain members feel that the process is not transparent or equitable.

1. **Key Requirement identified:**

**Transparency**:

* Participants need visibility into the financial status of the group, along with clear and accessible reporting on contributions, and withdrawals.

**Automation**:

* + Automating participant management, including registration, membership tracking.
  + Automating fund allocations, ensuring proper distribution according to the agreed rules.
  + Managing payments.

**Security**:

* + Mechanisms for verifying and authenticating participants to prevent unauthorized access.

**Financial Management**:

* + Automation of contributions, withdrawals, and interest calculations to minimize errors and delays.

**User-Friendly Interface**:

* + Easy-to-use platform for both organizers and participants.

**Group Management**:

* + Features for managing multiple groups or schemes, allowing organizers to oversee several communities simultaneously.

**Audit Trail**:

* + A comprehensive and transparent record of all financial transactions for auditing and dispute resolution.
  + Ability to track all actions taken within the system, providing accountability for both organizers and participants.

**2. Solution Overview**

**a. Solution Summary:**

The **Digital Chit Fund Management** provides a digital solution that automates the management of participants, handles fund allocations, and ensures transaction transparency. The platform is built using the MERN stack (MongoDB, Express.js, React, Node.js) with TypeScript, providing user-friendly environment for both participants and organizers of community savings groups.

There are three actors in the project: Organizers, participants, and admin, each with their own role’s functionalities.

**Organizers** are responsible for creating and managing savings groups. They can approve or reject participant join requests and allocate funds according to the group’s financial plan. Organizers have access to all group transactions and financial records, providing them with full control over the group’s activities.

**Participants** are the members who join the groups, make contributions, and track their financial involvement. They can view available group plans, request to join groups, and make payments according to the agreed ticket value. Participants also have access to their transaction history, enabling them to track their contributions, withdrawals, and overall financial status within the group.

**b. Objective:**

* To create a secure, transparent, and automated platform to streamline the management of savings groups.
* To ensure that participants can view the status of funds and have full transparency into all transactions and activities.
* To provide organizers with an easy-to-use system for managing multiple groups and tracking financial interactions.

**3. Features and Functionalities**

**a. Core Features**

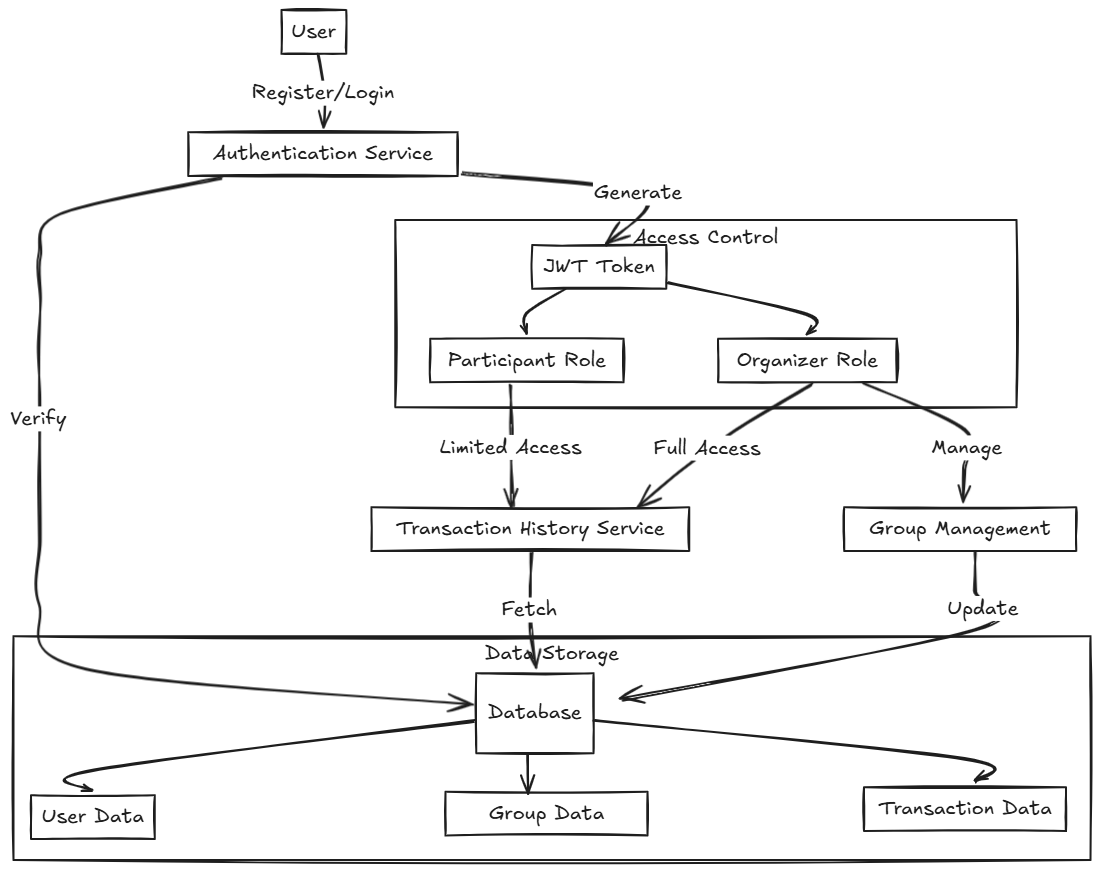
**Organizer Features**:

* Create and manage groups.
* Accept or reject participant join requests.
* View and manage the monthly financial plan.
* Access a complete history of group debit transactions.
* View all participants and their details in the group.
* View their transaction history.
* Edit and update the organizer’s profile.

**Participant Features**:

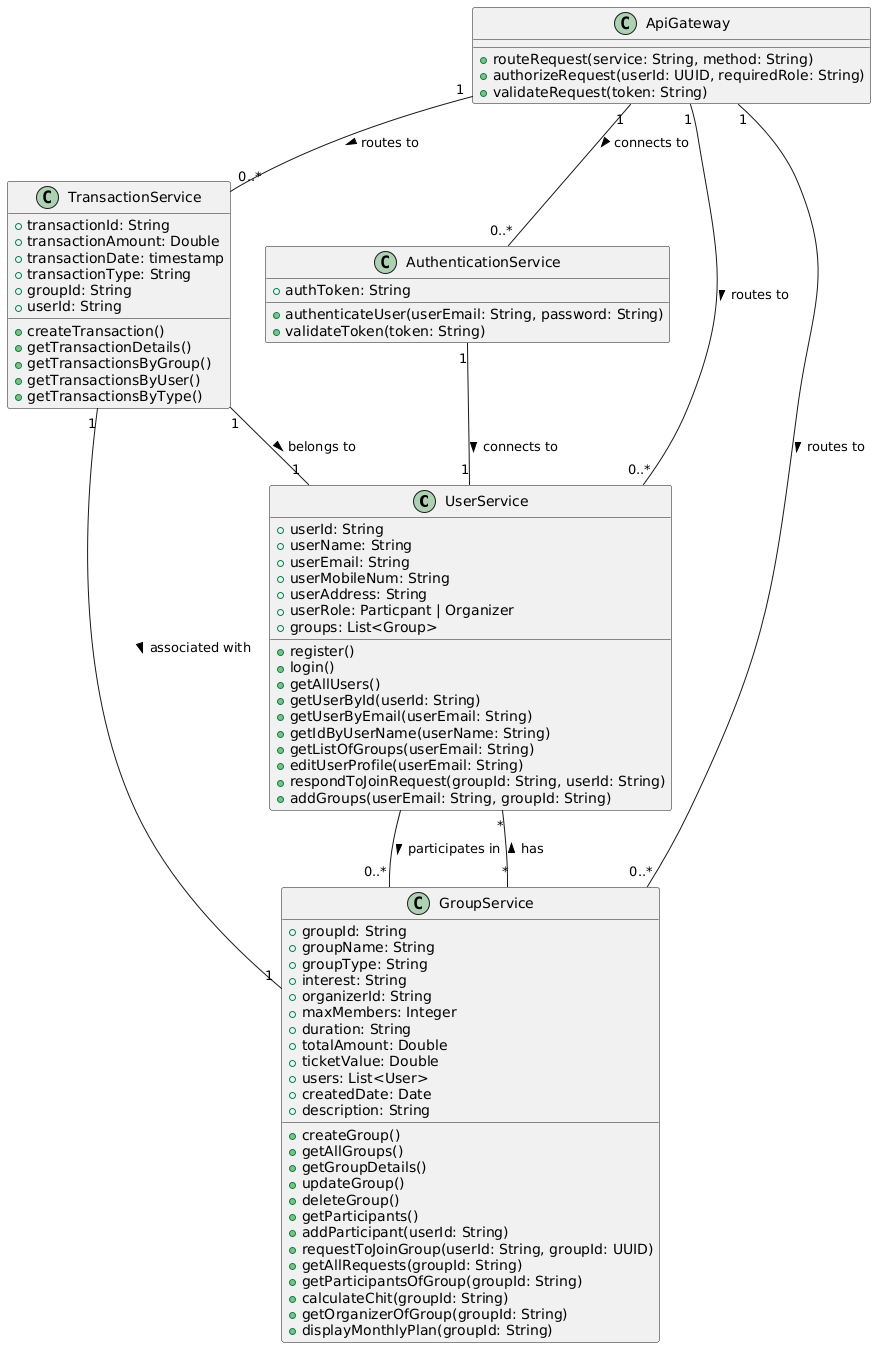
* View available group plans and their details.
* Request to join groups based on selected plans.
* Make payments according to the agreed ticket value.
* View all group debit transactions.
* View their transaction history.
* Edit and update the participant’s profile.

**b. User Flows**



**4. Architecture Diagram**

**a. System Architecture:**



**b. Key Components**

**User-Service:** Ithandles user-related functionalities. It manages the details of the platform’s users, whether they are **Participants** or **Organizers**, and provides methods for registration, login, profile editing, and retrieving user information.

**Group-Service:** Itmanages all functionalities related to groups within the platform. It handles creating groups, as well as managing participants and fund calculations.

**Transaction-Service:** Itis responsible for managing all financial transactions related to groups, users, and their associated activities. This includes recording and fetching transaction data.

**Auth-service:** It manages the authentication of the platform. It validates user login credentials.

**Api-Gateway:** It acts as the entry point for all external API requests, routing them to the appropriate service (e.g., UserService, GroupService, TransactionService). It is responsible for handling request routing and enforcing security measures like authentication and authorization.

**5. Technical Stack**

**a. Frontend (React & TypeScript)**

* **React**: JavaScript library for building the user interface, utilizing TypeScript for type safety and better maintainability.
* **Tailwind CSS**: Utility-first CSS framework for creating responsive and modern UI components with minimal custom CSS.

**b. Backend**

* **Node with TypeScript**: JavaScript runtime for building scalable server-side applications.
* **Express**: Web application framework for Node.js, providing a simple way to manage routes, middleware, and API requests.
* **JWT Authentication**: Used for secure user authentication and session management, ensuring only authorized users can access specific features.

**c. Database**

* **MongoDB**: NoSQL database for flexible data storage, ideal for handling unstructured or semi-structured data such as user profiles and transaction records.

**6. Prerequisites and Requirements**

**a. Technical Requirements**

* **Node.js** and **npm** installed for backend development.
* **MongoDB** for database management.
* **React** and **TypeScript** setup for frontend development.
* **JWT** (JSON Web Tokens) for secure authentication.
* **Tailwind CSS** for styling the frontend UI.

**b. Data Requirements**

* **User Data:** Includes participant name, contact details, role (participant/organizer), and financial transaction history.
* **Transaction Data:** Includes details of contributions, withdrawals, with timestamps and status.

**c. Access Permissions**

* **Participant Role:** View and manage personal contributions, request to join groups, and view transaction history.
* **Organizer Role:** Create and manage groups, fund allocations and participant management.
* **Admin Role:** Full access to the platform for managing users, roles, and financial records.

**7. Future Improvements**

**a. Planned Enhancements**

* Integration of Payment Gateway for real-time transactions.
* Notifications to users.
* Adding agent service who acts as a mediator between organizer and participants in transactions.

**b. Scalability Considerations**

* **Database Sharding:** To handle increasing amounts of data and ensure fast access as the number of participants and transactions grows.
* **Load Balancing:** Distribute traffic across multiple servers to ensure high availability and minimize downtime.

**8. Conclusion**

**a. Summary of Achievements**

* A fully functional platform has been developed to automate the management of collective savings groups, making the process more transparent, secure, and organized.
* Core features like participant management, fund allocation, and transaction transparency have been successfully implemented.
* The system has been designed with access permissions and user-friendliness in mind, with JWT-based authentication and a responsive frontend using React and TypeScript.

**b. Value Provided**

This platform empowers community savings groups by automating their processes, ensuring transparency, and providing secure access to financial records. It saves time for organizers and fosters trust among participants by making every transaction traceable and transparent. Additionally, the platform's automation reduces human errors and fraud, improving the overall efficiency and reliability of the group.