**Business Requirements Document (BRD)**

**Project Title:** Digital Chit Fund Management Platform

**1. Project Overview**

The "Digital Chit Fund Management" project aims to solve the problems faced by traditional savings groups in communities. Many of these groups struggle with transparency and organization, making it hard for members to see the financial status of their savings. Organizers also find it challenging to manage multiple groups efficiently. This project will create a secure online platform that helps manage participants, track fund allocations, and ensure everyone can easily understand the group's finances.

**2. Business Objectives**

* Enhance transparency in financial transactions.
* Automate participant management and fund allocation processes.
* Foster trust within the community through clear communication and reporting.

**3. Scope**

* **User Registration**: Implement registration and authentication for participants and organizers.
* **Group Management**: Allow organizers to create and manage multiple savings groups.
* **Financial Tracking**: Provide real-time visibility of contributions and group financial status for all members.
* **Transparency:** Ensure that all transactions and fund allocations are clearly visible to participants.

**4. Functional Requirements**

**4.1 Frontend (React & TypeScript)**

* **User Registration & Login**: User-friendly forms for account creation and secure login for participants and organizers.
* **Organizer Profile Page**: Display all groups managed by the organizer with options to view details and manage groups.
* **Group Search & Join**: Ability for participants to search for available chit fund groups and submit requests to join.
* **Transaction Dashboard**: Overview of user and group transaction history, including real-time updates on contributions and withdrawals.

**4.2. Backend (Node)**

* **User Management**: Implement authentication and authorization for both participants and organizers, ensuring secure access.
* **Group Management**: Functionality for organizers to create, manage, and delete chit fund groups, including member management.
* **Transaction Processing**: Handle the entire transaction lifecycle from contributions to withdrawals, ensuring real-time updates.

**5. Non-Functional Requirements**

**5.1 Performance:**

* The system must be able to handle peak loads, especially during high-demand periods, ensuring smooth operation during group contributions and transactions.

**5.2 Security:**

* Secure storage of user data, transaction details, and financial information, including encryption for sensitive data and compliance with data protection regulations.

**5.3 Scalability:**

* The system should scale easily to accommodate a growing number of participants and organizers without performance degradation.

**5.4 Usability:**

* The user interface must be intuitive and easy to navigate for users with varying levels of technical expertise, ensuring a seamless experience.

**6 User Stories:**

As a User(Participant):

* I want to register for an account so that I can participate in chit fund groups.
* I want to browse available chit fund groups so that I can choose which ones to join.
* I want to submit a request to join a chit fund group so that I can start participating in financial activities.
* I want to view my transaction history so that I can track my contributions and withdrawals.

As an Organizer:

* I want to register for an account so that I can create and manage chit fund groups.
* I want to create a new chit fund group so that I can gather participants for collective savings.
* I want to review and approve or reject participant requests to join my group so that I can maintain group integrity.
* I want to view the transaction history for my groups so that I can monitor financial activities and ensure transparency.
* I want to view the group information and payment status of all participants.

**7. Database Schema Overview**

The following outlines the key tables and their relationships based on the provided class diagram.

**1. Users Table**

* **Table Name**: users
* **Columns**:
  + user\_id (UUID, Primary Key)
  + user\_name (String)
  + user\_email (String, Unique)
  + user\_mobile\_num (String)
  + user\_address (String)

**2. Organizers Table**

* **Table Name**: organizers
* **Columns**:
  + organizer\_id (UUID, Primary Key)
  + organizer\_name (String)
  + organizer\_email (String, Unique)
  + organizer\_mobile\_num (String)
  + organizer\_address (String)

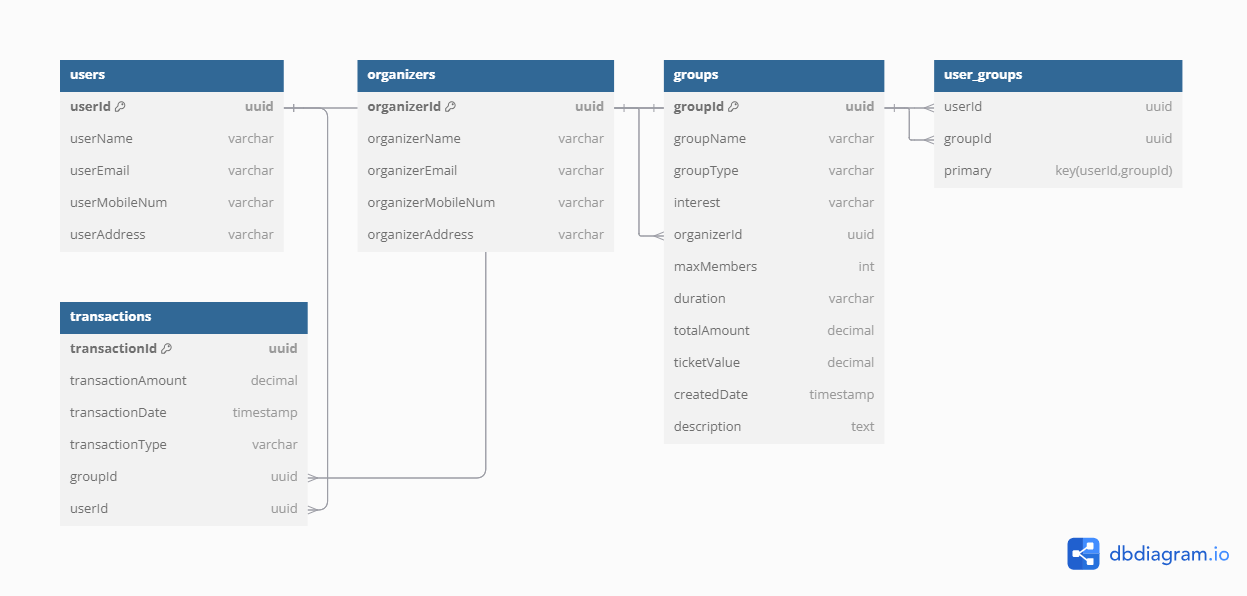
**3. Groups Table**

* **Table Name**: groups
* **Columns**:
  + group\_id (UUID, Primary Key)
  + group\_name (String)
  + group\_type (String)
  + interest (String)
  + organizer\_id (UUID, Foreign Key referencing organizers.organizer\_id)
  + max\_members (Integer)
  + duration (String)
  + total\_amount (Double)
  + ticket\_value (Double)
  + created\_date (Date)
  + description (String)

**4. Transactions Table**

* **Table Name**: transactions
* **Columns**:
  + transaction\_id (UUID, Primary Key)
  + transaction\_amount (Double)
  + transaction\_date (Timestamp)
  + transaction\_type (String, e.g., 'credit' or 'debit')
  + group\_id (UUID, Foreign Key referencing groups.group\_id)
  + user\_id (UUID, Foreign Key referencing users.user\_id)

**Relationships**

* **Users and Groups**:
  + A many-to-many relationship by *user\_groups* table, allowing multiple users to join multiple groups.
* **Organizers and Groups**:
  + A one-to-many relationship where each organizer can create multiple groups.
* **Groups and Transactions**:
  + A one-to-many relationship where each group can have multiple transactions associated with it.
* **Users and Transactions**:
  + A one-to-many relationship where each user can have multiple transactions.
* 

**8 Class diagram:**

