

Peer2Loan – Chit Fund Management System

Phase–2 Report

Team Members
Ashish Jain
Kathan Patel
Ekansh Patidar
Karan Thayat
Chinta Vishnu Vardhan

Current Understanding of the Project

Peer2Loan is a digital platform designed to automate and streamline the management of chit funds (ROSCA model). The system replaces manual record-keeping with a transparent, secure, and efficient web application for both organizers and members.

Core Problem

Traditional chit funds require:

- Manual tracking of contributions
- Manual payout management
- No formal records or audit trails
- No reminders or penalty enforcement

Current Understanding

The system must:

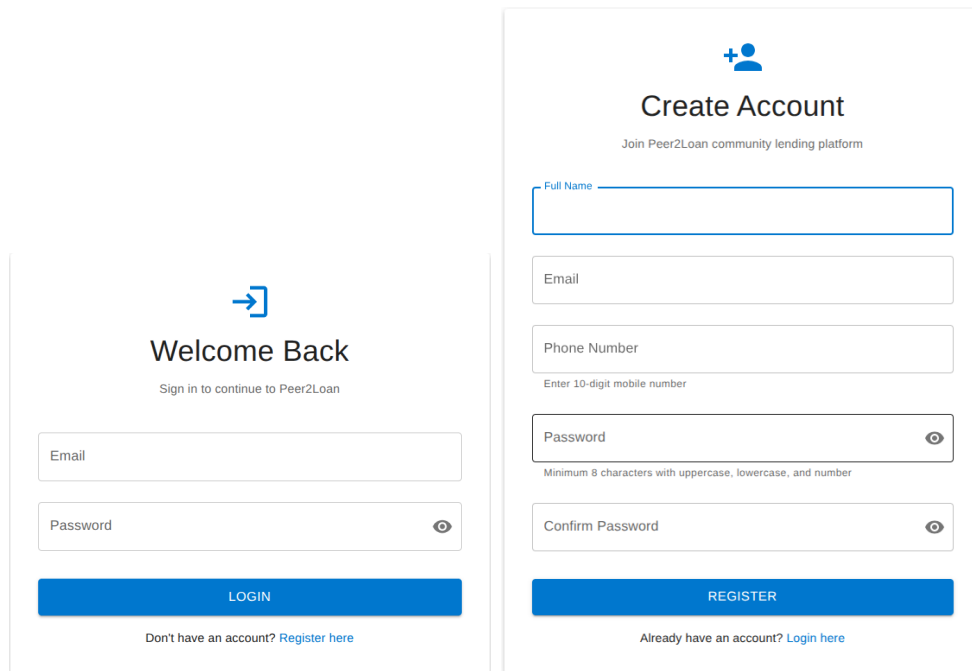
- Allow organizers to create/manage groups
- Allow members to join and contribute
- Automate monthly cycles and turn order
- Record payments and payouts with proof
- Apply penalties and compute member performance
- Maintain transparent ledgers and reports

UI Screens and User Journey

Below are the essential screens (images can be added later):

1. Login / Registration

User signs up, verifies account, and accesses dashboard.

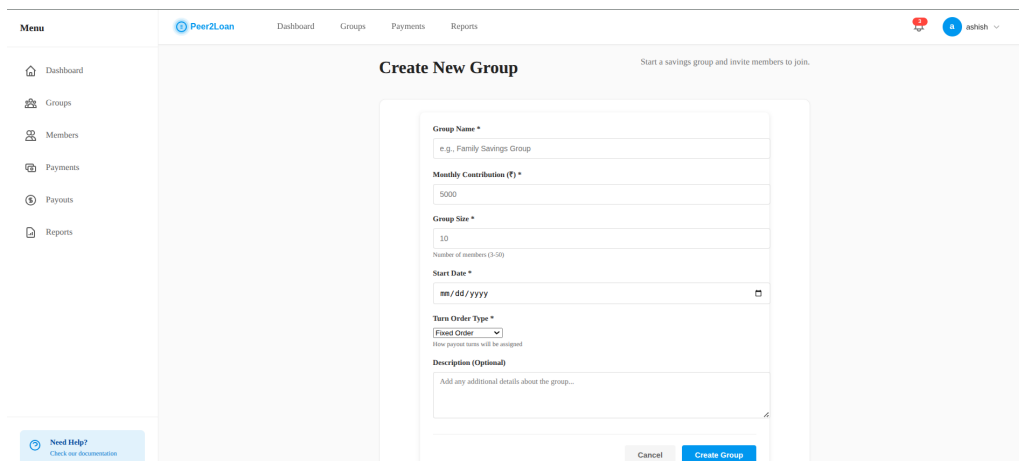


The image shows two side-by-side web forms. The left form is titled 'Welcome Back' with a blue icon of a person and a checkmark. It contains fields for 'Email' and 'Password' (with an eye icon for toggling visibility), a blue 'LOGIN' button, and a link 'Don't have an account? Register here'. The right form is titled 'Create Account' with a blue icon of a person and a plus sign. It contains fields for 'Full Name', 'Email', 'Phone Number' (with a note 'Enter 10-digit mobile number'), 'Password' (with a note 'Minimum 8 characters with uppercase, lowercase, and number' and an eye icon), and 'Confirm Password' (with an eye icon). It features a blue 'REGISTER' button and a link 'Already have an account? Login here'.

Figure 1: Login / Registration Screen (placeholder)

2. Organizer Flow

Create Group → Invite Members → Activate Group → Manage Monthly Cycle → Execute Payouts



The image shows a 'Create New Group' screen within a web application. The top navigation bar includes 'Menu', 'Peer2Loan', 'Dashboard', 'Groups', 'Payments', and 'Reports'. A user profile 'ashish' is visible in the top right. The left sidebar menu lists 'Dashboard', 'Groups', 'Members', 'Payments', 'Payouts', and 'Reports'. The main content area is titled 'Create New Group' with a subtitle 'Start a savings group and invite members to join.' The form contains the following fields: 'Group Name *' (with a placeholder 'e.g., Family Savings Group'), 'Monthly Contribution (₹) *' (with a placeholder '5000'), 'Group Size *' (with a placeholder '10' and a note 'Number of members (3-50)'), 'Start Date *' (with a date picker showing 'mm/dd/yyyy'), 'Turn Order Type *' (a dropdown menu set to 'Fixed Order' with a note 'How payout turns will be assigned'), and 'Description (Optional)' (a text area with a placeholder 'Add any additional details about the group...'). At the bottom right are 'Cancel' and 'Create Group' buttons. A 'Need Help? Check our documentation' link is in the bottom left.

Figure 2: Create Group Screen (placeholder)

3. Group Dashboard

View Group Overview → Track Contributions → Confirm Payments → Monitor Cycle Status → Execute Monthly Payout

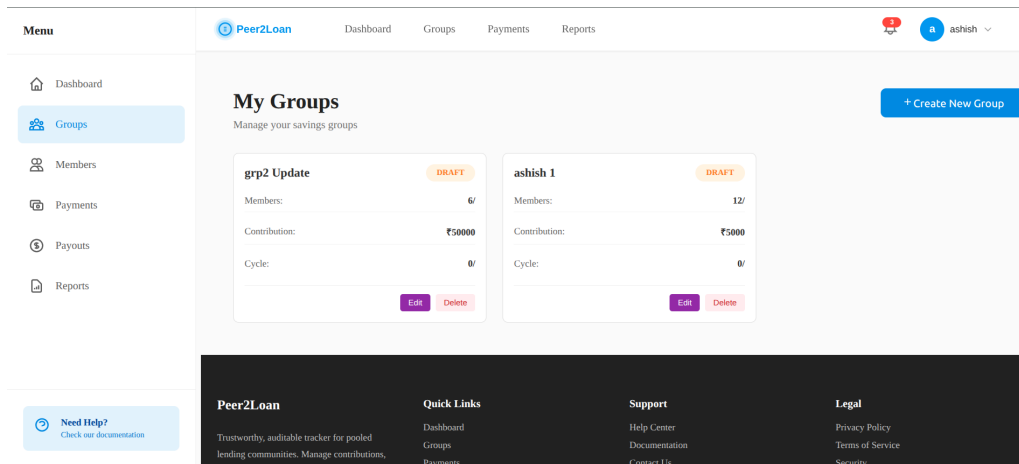


Figure 3: Group Dashboard Screen (placeholder)

Revised Solution Diagram



Figure 4: Revised Solution Architecture (placeholder)

Description

The updated architecture includes:

- **React Frontend** (UI, routing, context state)
- **Express Backend** (REST APIs, business logic)
- **MongoDB** (document-based schema)
- **Background Jobs** for reminders, penalty checks, cycle generation
- **File Uploads** using Multer for payment/payout proofs

Tech Stack Used

Frontend

- React 18, React Router 6
- Vite build tool
- Axios for API communication
- CSS3 (responsive design)

Backend

- Node.js 20, Express.js 4
- JWT authentication
- bcrypt password hashing
- Multer (file uploads)
- node-cron for background tasks

Database

- MongoDB 6
- Mongoose 8 ODM
- Collections: Users, Groups, Members, Cycles, Payments, Payouts, Penalties, AuditLogs

Revisions After Phase 1 Submission

1. Architecture Changes

Assumption Changed: A monolithic approach was planned. **Revised Decision:** Separated frontend and backend completely.

Reason:

- Independent deployments
- Simpler debugging
- Better scalability

2. Turn Assignment

Old Assumption: Turns will be assigned manually. **New Decision:** Auto-assignment + optional manual override.

3. Cycle Generation

Old Assumption: Organizer creates cycles manually. **New Decision:** System auto-generates cycles when group activates.

4. File Upload Proofs

Old Assumption: No upload mechanism. **New Decision:** Implemented Multer-based storage for payment proofs and payout proofs.

5. Performance Score

Old Assumption: Not planned. **New Decision:** Introduced a scoring mechanism based on:

- On-time payments
- Late/missed contributions
- Streak calculation

6. UI/UX Improvements

- Form validation
- Responsive screens
- Better loading states

POC Demonstration Video

Below is the link to the Phase-2 POC Demo (2–4 minutes):

- **YouTube Link:** <https://www.youtube.com/watch?v=yyck31qt8a0>