**🔶 1. Key Features of the ROSCA App**

* **User Registration & Login**
* **Group Creation & Joining**
* **Member Contribution Tracking**
* **Pot Distribution Management**
* **Payment Reminders / Status Notifications**
* **Admin Role for Group Oversight**
* **Transaction History**
* (Optional) **Mobile Responsiveness / App Version**

**🔶 2. Basic Tech Stack (Suggested)**

| **Layer** | **Suggested Tech** |
| --- | --- |
| Frontend | React / Next.js / Vue |
| Backend | Node.js (Express) or Django / Flask (Python) |
| Database | PostgreSQL / MySQL / MongoDB |
| Authentication | JWT or OAuth |
| Hosting | DigitalOcean, AWS, or Vercel (Frontend), Render (Backend) |
| Payments (Optional) | Stripe, PayPal, or local APIs |

**🔶 3. Database Schema Example**

Here’s a simple SQL-style schema to manage ROSCA logic:

**Tables:**

**users**

* id (PK)
* name
* email
* hashed\_password
* created\_at

**rosca\_groups**

* id (PK)
* group\_name
* admin\_user\_id (FK → users.id)
* contribution\_amount
* contribution\_interval (weekly, monthly)
* start\_date
* created\_at

**group\_members**

* id (PK)
* user\_id (FK → users.id)
* rosca\_group\_id (FK → rosca\_groups.id)
* order\_in\_rotation
* has\_received\_pot (boolean)

**contributions**

* id (PK)
* group\_member\_id (FK → group\_members.id)
* amount\_paid
* paid\_on (timestamp)

**pot\_distributions**

* id (PK)
* rosca\_group\_id (FK → rosca\_groups.id)
* recipient\_member\_id (FK → group\_members.id)
* amount\_distributed
* distributed\_on (timestamp)

**🔶 4. Project Development Roadmap**

**Phase 1: Backend API**

* User registration/login
* Create and manage ROSCA groups
* Add members to groups
* Track contributions & pot distributions
* RESTful or GraphQL API

**Phase 2: Frontend Website**

* Dashboard with active groups
* Group details page (show rotation, members, contribution schedule)
* Payment input / tracking UI
* Admin control interface (to distribute pots manually or automatically)
* Notifications for pending contributions

**Phase 3: Extra Features**

* Email/SMS reminders
* Payment API integration
* Mobile app version (React Native / Flutter)

**🔶 5. Example Starter Stack**

* **Backend**: Node.js (Express) + PostgreSQL using Sequelize ORM
* **Frontend**: React + Tailwind CSS for styling
* **Auth**: JWT-based authentication
* **Hosting**: Vercel (Frontend) + Render or Railway (Backend)

**🔶 Next Step:**

I can:

* Generate starter **code templates** (Node.js backend, React frontend).
* Build your **SQL database schema** as migration files.
* Provide **API endpoint designs**.
* Help with deploying your project.