Supabase Authentiucation Cereboai  
  
Creating a multi-tier membership system with Supabase authentication and role-based access control (RBAC) is entirely feasible! Here's a comprehensive approach to setting up the system:

**Authentication Setup**

1. **Configure Authentication Methods**: Enable email/password authentication, OAuth providers (Google, GitHub, etc.), and Magic Links in the Supabase dashboard.
2. **JWT-Based Role Management**: Use Supabase's JWT tokens to manage roles. Each user's role will be included in their JWT token.
3. **Custom Claims**: Add custom claims to JWT tokens to store user roles. You can do this using Supabase's authentication hooks.

**Database Schema Design**

1. **Users Table**: Store user information.
2. **Roles Table**: Define the seven roles.
3. **Permissions Table**: Define specific permissions for each role.
4. **Marketplace Table**: Store marketplace items.
5. **MVP Ratings Table**: Store ratings and evaluations by Community Developers and Community SMEs.

**Example SQL Scripts**

**Users Table**

sql

CREATE TABLE users (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

email TEXT UNIQUE NOT NULL,

role TEXT NOT NULL DEFAULT 'Standard Member'

);

**Roles Table**

sql

CREATE TABLE roles (

role TEXT PRIMARY KEY,

description TEXT

);

INSERT INTO roles (role, description) VALUES

('Administrator', 'Full read/write access across the entire project'),

('Standard Member', 'Read/write access to own profile, access to Marketplace, can communicate with other members, default 10MB storage limit'),

('Premium Member', 'Read/write access to own profile, access to Marketplace, can communicate with other members, access to developer tools, can list projects in Marketplace, 50MB storage limit'),

('Standard Developer', 'Read/write access to own profile, access to Marketplace, can communicate with other members, access to developer tools, can list projects in Marketplace, 50MB storage limit'),

('Premium Developer', 'Read/write access to own profile, access to Marketplace, can communicate with other members, access to developer tools, can list projects in Marketplace, 100MB storage limit'),

('Community Developer', 'Read/write access to own profile, access to Marketplace, can communicate with other members, access to developer tools, can list projects in Marketplace, 100MB storage limit, can review MVPs'),

('Community SME', 'Read/write access to own profile, access to Marketplace, can communicate with other members, access to developer tools, can list projects in Marketplace, 100MB storage limit, can review and remove unsafe MVPs');

**Role Upgrade System**

1. **Role Upgrades**: Allow users to request role upgrades. This can be done through a dedicated table to store upgrade requests.
2. **Approval Workflow**: Implement an approval workflow for role upgrades.

**Row-Level Security (RLS) Policies**

1. **Enable RLS**:

sql

ALTER TABLE users ENABLE ROW LEVEL SECURITY;

ALTER TABLE marketplace ENABLE ROW LEVEL SECURITY;

1. **Define Policies**:

sql

CREATE POLICY "User is allowed to access their own data" ON users

FOR SELECT USING (auth.uid() = id);

CREATE POLICY "Marketplace access for members" ON marketplace

FOR SELECT USING (EXISTS (SELECT 1 FROM users WHERE users.id = auth.uid() AND users.role IN ('Standard Member', 'Premium Member', 'Standard Developer', 'Premium Developer', 'Community Developer', 'Community SME')));

**Best Practices for Scalability and Security**

* **Indexing**: Ensure proper indexing for tables to improve query performance.
* **Regular Audits**: Conduct regular security audits to identify and mitigate potential vulnerabilities.
* **Monitoring**: Implement monitoring and alerting for suspicious activities.