Project Overview - [DevSphere]

A social media platform where programmers can showcase their work, interact with other developers, receive challenges, and get recognized by organizations for their skills.

Project Flow :

1. Normal User Flow :

1. Create Account :

a) Sign up via email or GitHub (OAuth).
b) Basic profile setup (Name, Bio, Location, Skills, GitHub/LinkedIn links).
c) Select programming interests (e.g., Python, React, Machine Learning).

2. Dashboard :

a) My Profile:

- ➢ View and update profile information.

b) Post Projects:

- ➢ Share projects with details like tech stack, challenges faced, and solutions.

c) Receive Feedback:

- ➢ Comments and suggestions from organizations or other users.

d) Search:

- ➢ Find organizations, people (e.g., mentors, recruiters), and open-source projects.
- ➢ Follow Organizations and view their posts.

e) Challenge Participation:

- Join coding challenges posted by organizations according to interests.

f) **Badges and Achievements:**

- Earn badges for solving challenges, posting quality projects, or getting likes/comments.
- After earning several badges, users get a Blue Tick (verification status).

3. .Interaction with Others:

i. Comment on posts, share ideas, and collaborate.

ii. Engage in group discussions or join specific interest-based communities (e.g., AI, Web Development).

2. Admin Flow :

Role: Manage platform integrity, user content, and activity.

1. User Management:

- View and manage all user profiles (edit, delete, ban if necessary).
- Flag inappropriate content (comments, posts). (Optianal)

2. Analytics:

- Monitor user activity, engagement rates, and platform growth.
- Generate reports for tracking challenges, badge distributions, and user achievements.

3. Platform Settings:

- Configure platform settings (e.g., challenge categories, notification preferences, security settings).

3. Organization Flow :

Role: Organizations or recruiters that interact with users for recruitment, challenges, and collaboration.

1. **Create Profile:**
    1. Organizations create a profile that includes their name, mission, and tech stack preferences.
2. **Post Challenges:**
    1. Organizations can create challenges for users to solve, providing rewards (e.g., job offers, recognition).
    2. Specify difficulty, expected outcomes, and timelines for challenges.
3. **Search for Talent:**
    1. Filter and search user profiles based on skills, experience, or interests.
    2. View user portfolios (projects, GitHub repos).
4. **Engagement:**
    1. Comment on user projects, offer feedback, and suggest improvements.
    2. Send direct messages or connect with potential candidates for job offers.
5. **Job Postings:**
    1. Post job openings or freelance opportunities targeted at specific skill sets.
    2. Include salary range, job requirements, and application instructions.

Technologies and Tools :

Frontend: React, JavaScript, HTML, Tailwind CSS.
Backend: Node.js, Express.js,
Database: MongoDB
Version Control: Git

Scheamas for Database for reference

1. Users

- `user_id` (INT, PRIMARY KEY, AUTO_INCREMENT) - Unique identifier for each user.
- `username` (VARCHAR(255), UNIQUE) - Unique username for the user.
- `email` (VARCHAR(255), UNIQUE) - User's email address.
- `password_hash` (VARCHAR(255)) - Hashed password for security.
- `first_name` (VARCHAR(255)) - User's first name.
- `last_name` (VARCHAR(255)) - User's last name.
- `bio` (TEXT) - User's short bio or description.
- `location` (VARCHAR(255)) - User's location.
- `profile_picture` (VARCHAR(255)) - URL of the user's profile picture.
- `created_at` (TIMESTAMP) - Timestamp of when the user account was created.

2. Organizations

- `organization_id` (INT, PRIMARY KEY, AUTO_INCREMENT) - Unique identifier for each organization.
- `name` (VARCHAR(255), UNIQUE) - Name of the organization.
- `description` (TEXT) - Description of the organization.
- `website` (VARCHAR(255)) - Organization's website URL.
- `logo` (VARCHAR(255)) - URL of the organization's logo.
- `created_at` (TIMESTAMP) - Timestamp of when the organization was created.

3. Posts

- `post_id` (INT, PRIMARY KEY, AUTO_INCREMENT) - Unique identifier for each post.
- `user_id` (INT, FOREIGN KEY REFERENCES Users(user_id)) - ID of the user who created the post.
- `organization_id` (INT, FOREIGN KEY REFERENCES Organizations(organization_id), NULLABLE) - ID of the organization that created the post (if applicable).
- `content` (TEXT) - Content of the post (text, code, links, etc.).
- `created_at` (TIMESTAMP) - Timestamp of when the post was created.

4. Comments

- `comment_id` (INT, PRIMARY KEY, AUTO_INCREMENT) - Unique identifier for each comment.
- `post_id` (INT, FOREIGN KEY REFERENCES Posts(post_id)) - ID of the post being commented on.
- `user_id` (INT, FOREIGN KEY REFERENCES Users(user_id)) - ID of the user who wrote the comment.

- `content` (TEXT) - Text content of the comment.
- `created_at` (TIMESTAMP) - Timestamp of when the comment was created.

5. Likes

- `like_id` (INT, PRIMARY KEY, AUTO_INCREMENT) - Unique identifier for each like.
- `user_id` (INT, FOREIGN KEY REFERENCES Users(user_id)) - ID of the user who liked the post.
- `post_id` (INT, FOREIGN KEY REFERENCES Posts(post_id)) - ID of the post that was liked.

6. Follows

- `follow_id` (INT, PRIMARY KEY, AUTO_INCREMENT) - Unique identifier for each follow relationship.
- `follower_id` (INT, FOREIGN KEY REFERENCES Users(user_id)) - ID of the user who is following.
- `followed_id` (INT, FOREIGN KEY REFERENCES Users(user_id)) - ID of the user being followed.

7. Challenges

- `challenge_id` (INT, PRIMARY KEY, AUTO_INCREMENT) - Unique identifier for each challenge.
- `organization_id` (INT, FOREIGN KEY REFERENCES Organizations(organization_id)) - ID of the organization that created the challenge.
- `title` (VARCHAR(255)) - Title of the challenge.
- `description` (TEXT) - Description of the challenge.
- `start_date` (DATE) - Start date of the challenge.
- `end_date` (DATE) - End date of the challenge.
- `reward` (TEXT) - Reward for completing the challenge.

8. User_Skills

- `user_id` (INT, FOREIGN KEY REFERENCES Users(user_id))
- `skill_name` (VARCHAR(255)) - Name of the skill (e.g., "Python", "JavaScript", "React")

9. Badges

- `badge_id` (INT, PRIMARY KEY, AUTO_INCREMENT) - Unique identifier for each badge.
- `name` (VARCHAR(255)) - Name of the badge.
- `description` (TEXT) - Description of the badge.
- `criteria` (TEXT) - Criteria for earning the badge.

10. User_Badges

- `user_id` (INT, FOREIGN KEY REFERENCES Users(user_id))
- `badge_id` (INT, FOREIGN KEY REFERENCES Badges(badge_id))

11. Challenge_Participants

- `user_id` (INT, FOREIGN KEY REFERENCES Users(user_id))
- `challenge_id` (INT, FOREIGN KEY REFERENCES Challenges(challenge_id))