# 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 :

#### **Normal User Flow :**

###### Create Account :

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

###### Dashboard :

* 1. My Profile:
* View and update profile information.
  1. Post Projects:
* Share projects with details like tech stack, challenges faced, and solutions.
  1. Receive Feedback:
* Comments and suggestions from organizations or other users.
  1. Search:
* Find organizations, people (e.g., mentors, recruiters), and open-source projects.
* Follow Organizations and view their posts.
  1. Challenge Participation:
* Join coding challenges posted by organizations according to interests.
  1. 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).

###### .Interaction with Others:

1. Comment on posts, share ideas, and collaborate.
2. Engage in group discussions or join specific interest-based communities (e.g., AI, Web Development).

#### **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)

1. ****Analytics****:

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

1. ****Platform Settings****:

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

#### **Organization Flow :**

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

###### ****Create Profile****:

* 1. Organizations create a profile that includes their name, mission, and tech stack preferences.

###### ****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.

###### ****Search for Talent****:

* 1. Filter and search user profiles based on skills, experience, or interests.
  2. View user portfolios (projects, GitHub repos).

###### ****Engagement****:

* 1. Comment on user projects, offer feedback, and suggest improvements.
  2. Send direct messages or connect with potential candidates for job offers.

###### ****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))