Contextual Role-Based Access Control - TaskBuddy

Definition

Contextual Role-Based Access Control (RBAC) is an authorization model where a user's role and permissions are determined dynamically based on the specific context or resource, such as a project. Instead of having a fixed role globally (like ADMIN or USER), users are assigned roles like TEAM_LEAD or MEMBER within individual projects.

Why It's Ideal for TaskBuddy

- Users can be Team Lead in one project and Member in another.
- Permissions vary per project.
- Matches real-world collaborative workflows.
- Scalable for multi-project systems like Trello, GitHub, etc.

Backend Implementation (Theory)

1. Tables:

- users (id, name, email, password)
- projects (id, title, created_by)
- project_members (id, project_id, user_id, role, status)

2. On login, fetch user's projects and their role in each.

3. Before allowing actions (e.g., assigning tasks), check:

```
  SELECT role FROM project_members WHERE user_id = ? AND project_id = ?
```

4. Use this role to permit or deny access.

Frontend Implementation (Theory)

1. Receive userRole (TEAM_LEAD / MEMBER) from backend with project data.

2. Conditionally render UI or restrict actions using:

```
  if (userRole === 'TEAM_LEAD') { show assign button }
```

3. OR, route users directly to their respective dashboard pages based on role.

Example:

```
  useEffect(() => {
    if (userRole === 'TEAM_LEAD') navigate('/project/lead');
    else navigate('/project/member');
  }, []);
```

Routing Benefits

- Clean separation of concerns (different UIs per role)
- No messy conditional rendering on a shared page
- Better UX and easier scaling when new roles are added
- More secure and intuitive navigation

Conclusion

Contextual RBAC is perfect for apps like TaskBuddy. It offers flexibility, clean role handling, and real-world alignment with how users participate in multiple projects. Implementation is straightforward with minimal backend checks and clear frontend logic or routing based on roles.