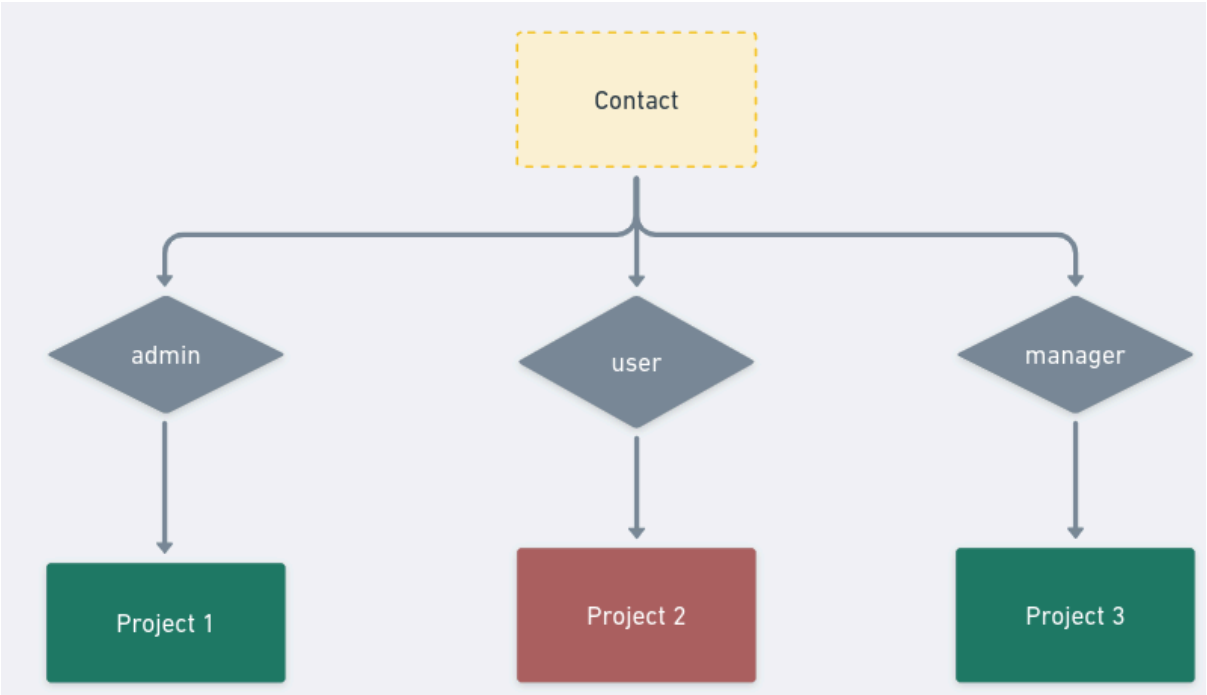
Managing Contact Roles



To manage contact roles efficiently, we can define the TypeScript interface as follows:

```
export interface Contact {
        id: number;
         firstName: string;
        lastName: string;
        phoneNumber: string;
        email: string;
        roles: {
                [projectId: number]: Role
        };
  }
```

In this interface, the roles property is an object where each key represents a projectId, and the value is the Role assigned to the contact for that project. This design facilitates efficient storage and retrieval of role information.

To implement this in different database systems:

1. **NoSQL Databases (e.g., MongoDB)**: We can embed project-role pairs directly within the contact document. Each contact document would contain an array of objects, where each object represents a project and its associated role. This approach allows for straightforward data retrieval and updates, as roles are stored alongside contact information.

```
{"_id": "contactId",
"firstName": "John",
"lastName": "Doe",
"phoneNumber": "123-456-7890",
"email": "john.doe@example.com",
"projects": [
   {
      "projectId": "project1",
      "role": "admin"
   },
   {
      "projectId": "project2",
      "role": "manager"
    }
 ]}
```

2. **Relational Databases (e.g., MySQL, PostgreSQL)**: We can use a relational schema involving three tables: Contacts, Projects, and a junction table. The junction table would map contacts to projects and store their roles, using foreign keys to link to the Contacts and Projects tables.

Example schema:

- **Contacts Table**: Stores contact details.
- **Projects Table**: Stores project details.
- **ContactProjectRoles Table**: Map contactId to projectId and include a role column to specify the role.