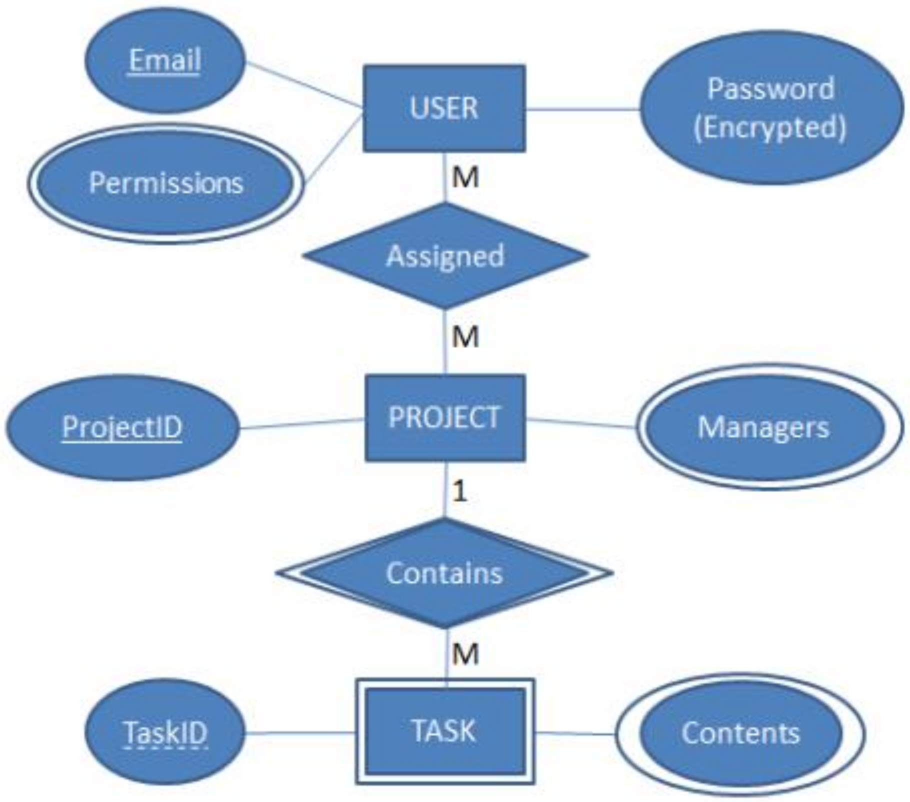
Schemas:



Example UserInfo JSON -

```
 { "name": "First Last",
  "email": "first.last@example.com",
“Password”:”xxxxxxxxxxxx”(Hashed password),
  "Projects": [
      {
         "Project_id": "xxxxxx”
      },
      {
         "Project_id": "xxxxx”
            }
         ]
}
```

Example ProjectInfo:

```
{“ProjectId”:”xxxxxxxxxxxxx”,
“Tasks”:[
“Taskid”:”xxxxxxxx”,
“Taskid”:”xxxxxxxx”],
```

“Manager”:”<email>//of user”
}

Example Task:
{“TaskId”:”xxxxx”,
“DueDate”:”xxxxxxxx”,
“Deliverable”:file/.doc/link,
“Assinged_Users”:[],
“Manager”:user
}

Query Logic:

Each User holds an array of project ID’s,
All Project’s contain an array of task ID’s
You can always query User -> project -> task -> User ….
Emails/ProjectId/TaskId Should all be unique identifiers used to get each in place of _id\;

Find:

User Info:
Query by email: users.find(“email”:”xxxx”)
Project:
Query by Project ID:
{“projects”: <id>}
Task:
Query by Task ID
{“Task”: <id>}

Delete:

Project from a user -
users.update({“email”:<email>},{$pull:{“Projects.projectid”:’<projectid>’}})
task from project -
task.update({“project”:<projectid>},{$pull:{“tasks.taskid”:’<taskid>’}})

Update(Add users/projects/tasks):

Project to a user -
users.updateOne({“email”:<email>},{$push:{“Projects”:’<projectid>’}})
Task to a user -
users.updateOne({“project”:<projectid>},{$push:{“tasks”:’<taskid>’}})

CRUD Ops:

Data Collections:

USER: Create new user in /register route, Read user data in /login and /userinfo routes, Update user data when changing password or adding projects, Delete user when removing account.

PROJECT: Create new project in /userinfo/projects route, Read projects in /userinfo/projects route, Update when adding users or tasks, Delete when removing a project.

TASK: Create new task in /userinfo/{project}/tasks route. Read tasks in /userinfo/{project}/tasks route, Update when adding new contents, Delete when removing task.

userPermissions: Create new permission when adding a user to a project, Read permissions when determining if user is an admin and whether or not user can edit a task, Update when changing user permission, Delete when removing user from a project.

projectManagers: Create new projectManager when adding user to a project, Read managers when determining if user can add/edit tasks, Update managers when changing user permission. Delete when removing user from a project.

taskContents: Create new task contents in /userinfo/{task}/edit route, Read task contents in /userinfo/{task}/edit and /userinfo/{task}/read routes, Update when editing task contents, Delete when removing task contents.

Tooling & Libs:

MongoDB