School of Computer Science and Engineering

COMP3900: Computer Science Project 2022T2

Task Management System

Project Name: ProTask

Team Name: BackendOnly

Team Members:

Name	Student ID	eMail	Role
Jackie Cai	z5259449	z5259449@ad.unsw.edu.au	Developer
Hsin Chen	z5302902	z5302902@ad.unsw.edu.au	Developer
Julian Liu	z5165524	z5165524@ad.unsw.edu.au	Scrum Master
Raiyaan Ruhi	z5308345	z5308345@ad.unsw.edu.au	Developer

Submission Date: 05/08/2022

Table of Contents: Page

1. Introduction

The most important aspect of a successful project is organization. With the emergence of remote working this aspect has become simultaneously much more important and much more difficult to handle. Our application *ProTask* aims to offer a solution to this problem by creating an online platform to facilitate collaboration and improve work organization. *ProTask* is an online application where users are able to easily connect with each other to assign tasks and projects. The *ProTask* application sets itself apart from the pre-existing online management systems by offering novel functionality such as a busy factor to track business of connections, projects to better organize tasks and a public task board for users to pick up available tasks. Furthermore, there are user and task related skills to help managers understand the strengths and weaknesses of each of their connections, public connection boards to better aid users find other users with required skills and many other features.

This report will detail the architecture, design and implementation of the *ProTask* application.

2. Overview

2.1. General Architecture

The general architecture of the Protask app is to design and implement different functionalities that build upon itself to support future functionalities which adhere to the project objectives. The main foundation of the system is the user who creates a personal profile and then is able to create tasks that they want to accomplish or assign to one of their connections to do.

Users: The user types include task masters and project managers, both of which interact with the system through the interface layer, specifically the user interface. Their actions are then processed by the business layer, which then provides responses to their actions through the user interface.

Interface: The interface layer primarily utilizes React JavaScript to produce the user interface, including the CSS for web page customisation and HTML for formatting. The React JS interface will connect with the business layer through a Flask server, which will handle all requests from the business layer to the interface layer.

Database: The database layer will utilize SQLAlchemy storing all the data that the user inputs for profile and task related activities.

The architecture of the Protask app is using Flask Python for backend, storing our data into an SQLAlchemy database and using React.js, HTML and CSS in order to present it in the frontend to allow for user navigation as shown below:

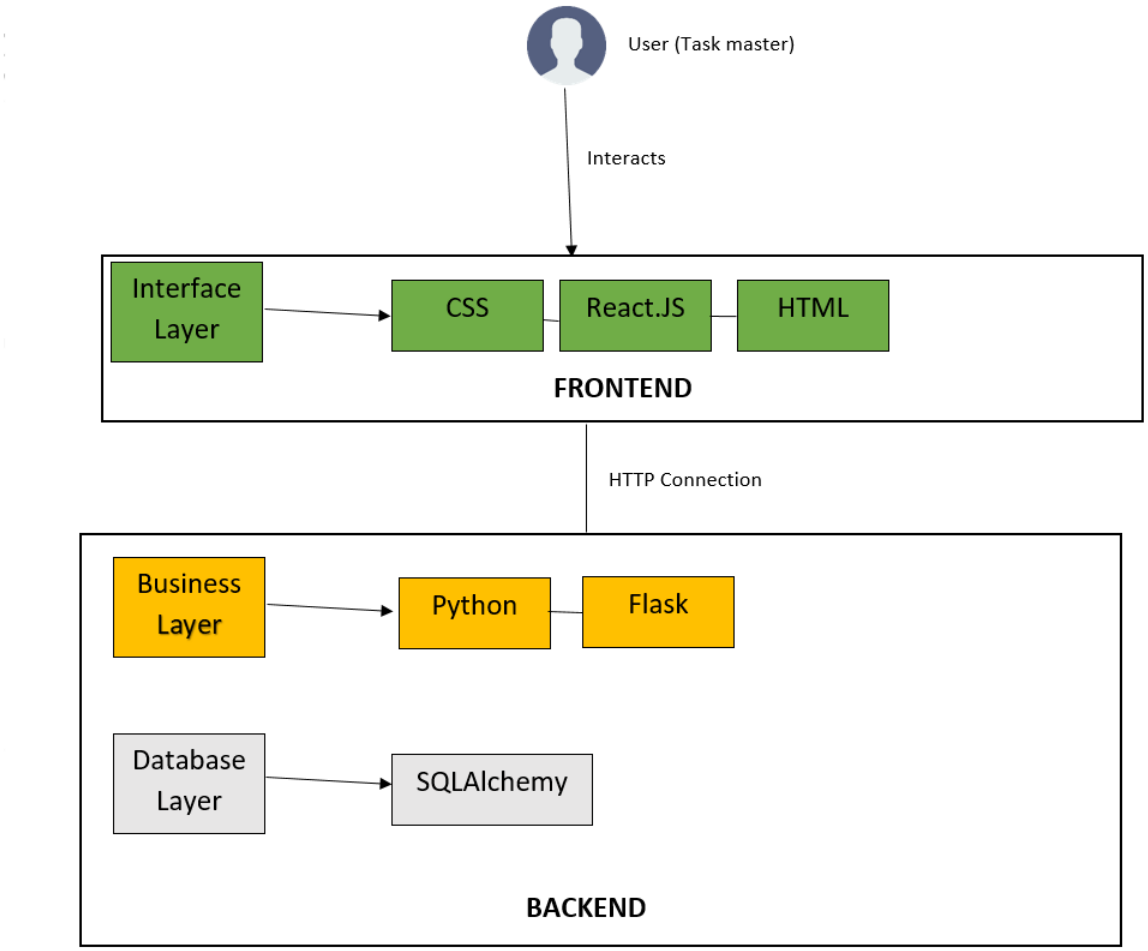


Diagram 1 : The Architecture of the Protask System

2.2 Database architecture:

The database system works with the backend system using SQLAlchemy which stores and extracts the relevant data whenever the frontend or the backend needs it. The database architecture can be separated into 5 different tables: user section, task section, project section, friend search section and notification section. The following 5 ER diagrams below show the relationship between all the tables created surrounding these 5 sections:

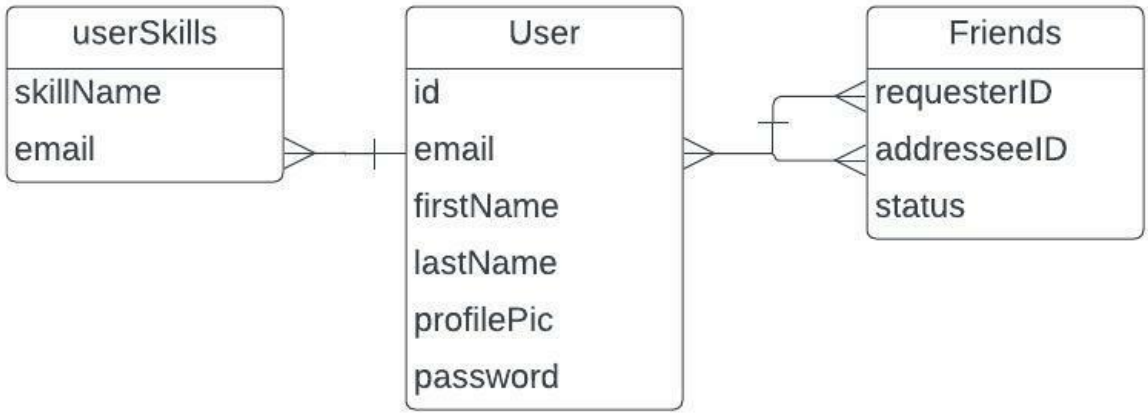


Diagram 2 : User Relationship

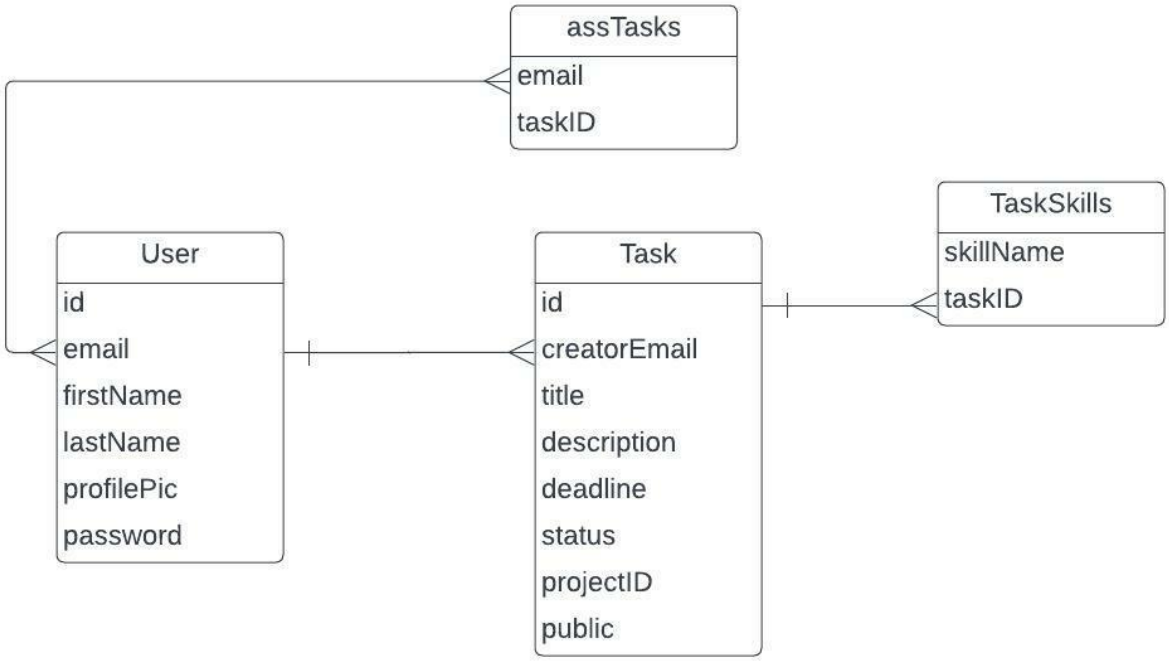


Diagram 3: Task Relationship

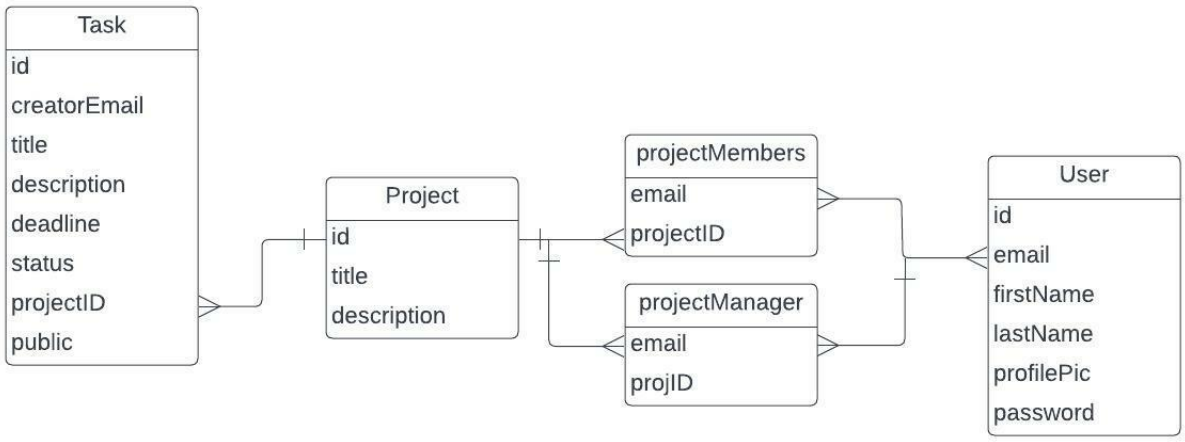


Diagram 4: Project Relationship

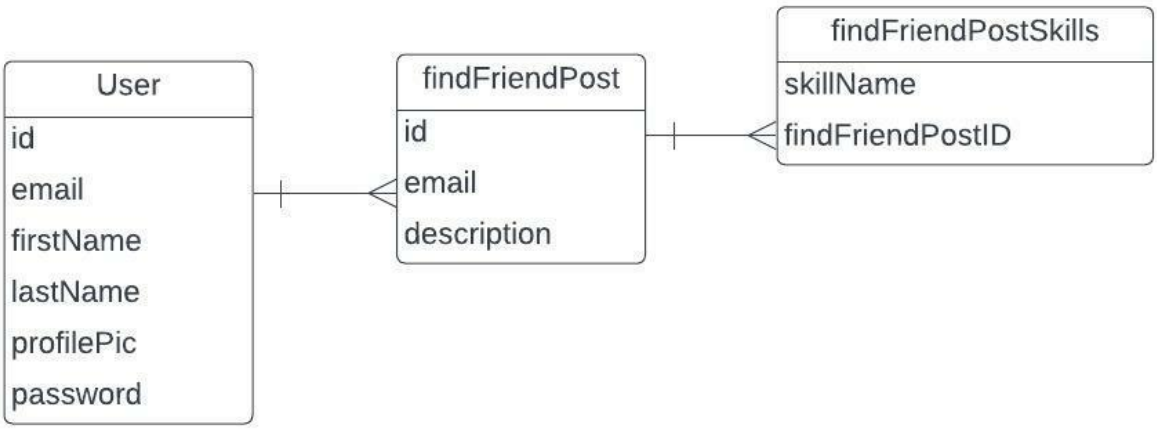


Diagram 5: Friend Search Post Relationship

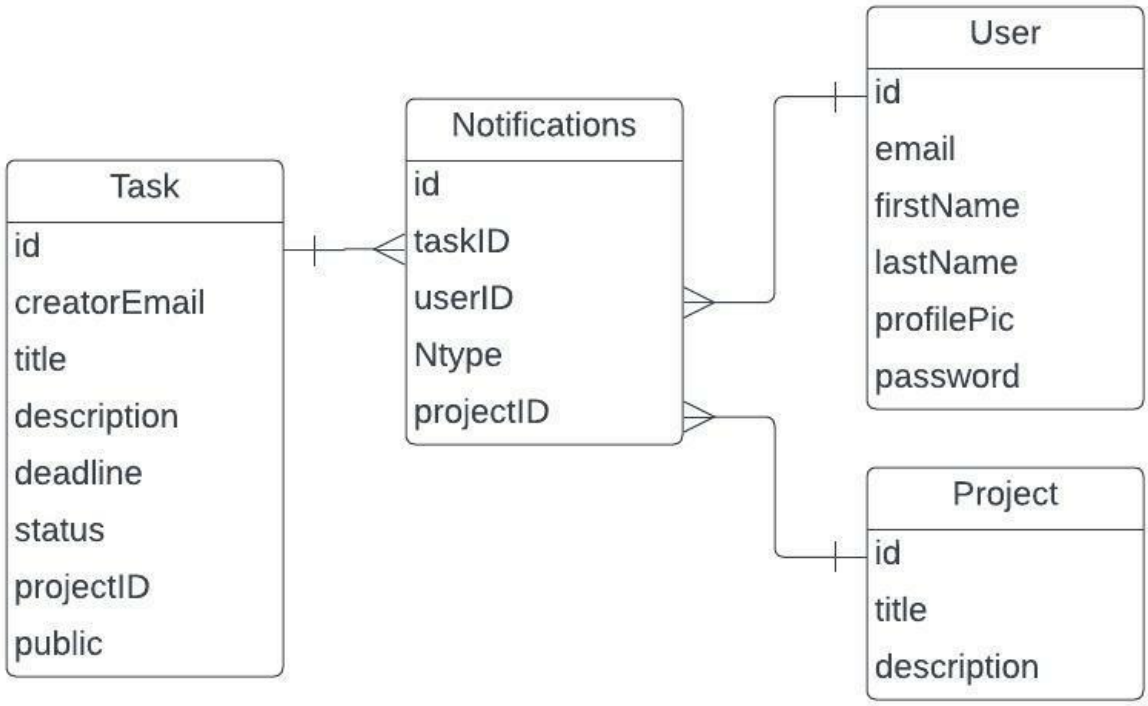


Diagram 6: Notifications Relationship

2.3 Backend Architecture

The back end is the intermediary between the database and the front end, it is built on the flask web framework using python. Processing the incoming request from the server goes through the logic and returns a relative response to the user. Clients will pass the request from the front end which is processed to the back-end server. The back end will store the incoming data in the database and call the data when the front end requests. There are five different files handling different parts of the functionalities of the app. As in the diagram below, the functionalities are split into five categories, authentication, task, project, skill, and recommendations:

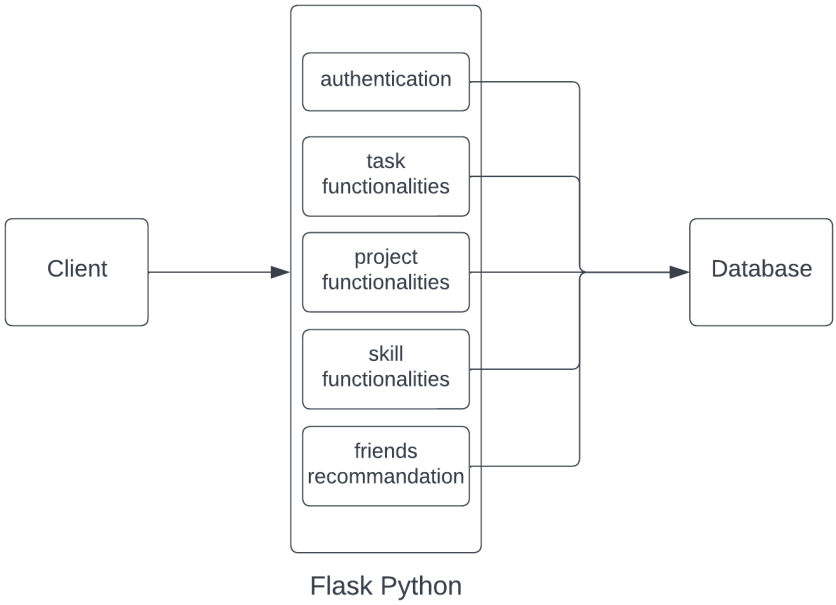
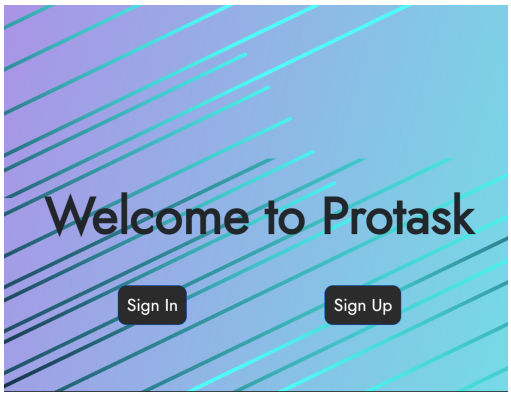


Diagram 7: Back-end Architecture

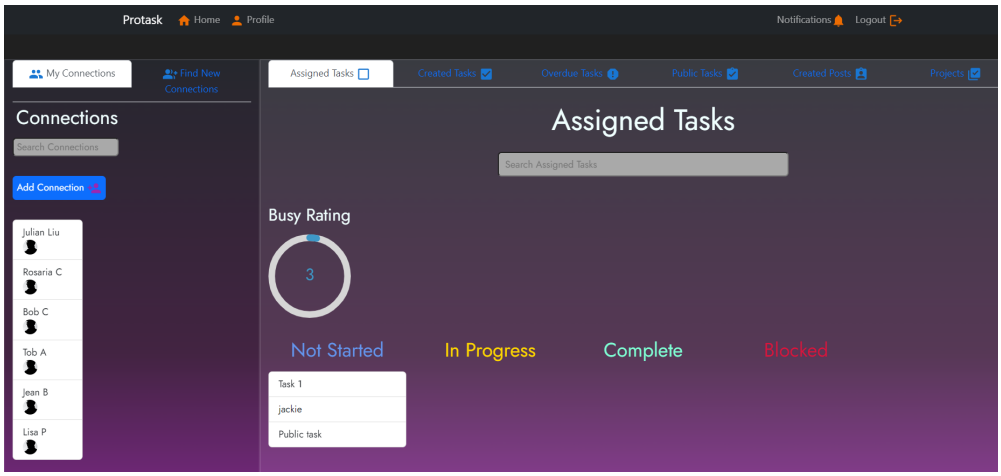
2.4 Front end Architecture

The front end was primarily designed with the philosophy of providing a smooth user experience, and shares many design conventions of similar applications through the use of React.JS,CSS and HTML. Maintaining a single page application using React facilitated the emphasis on providing a good user experience. In the app, the user will spend time on only several pages which provide all the functionality through seamless navigation between sections, thus keeping the user engaged and reducing the amount of redundant page loads and overall downtime of not using the application. Furthermore, graphic design principles were implemented to provide clarity, feedback and information to the user, contributing to the accessibility of the application.

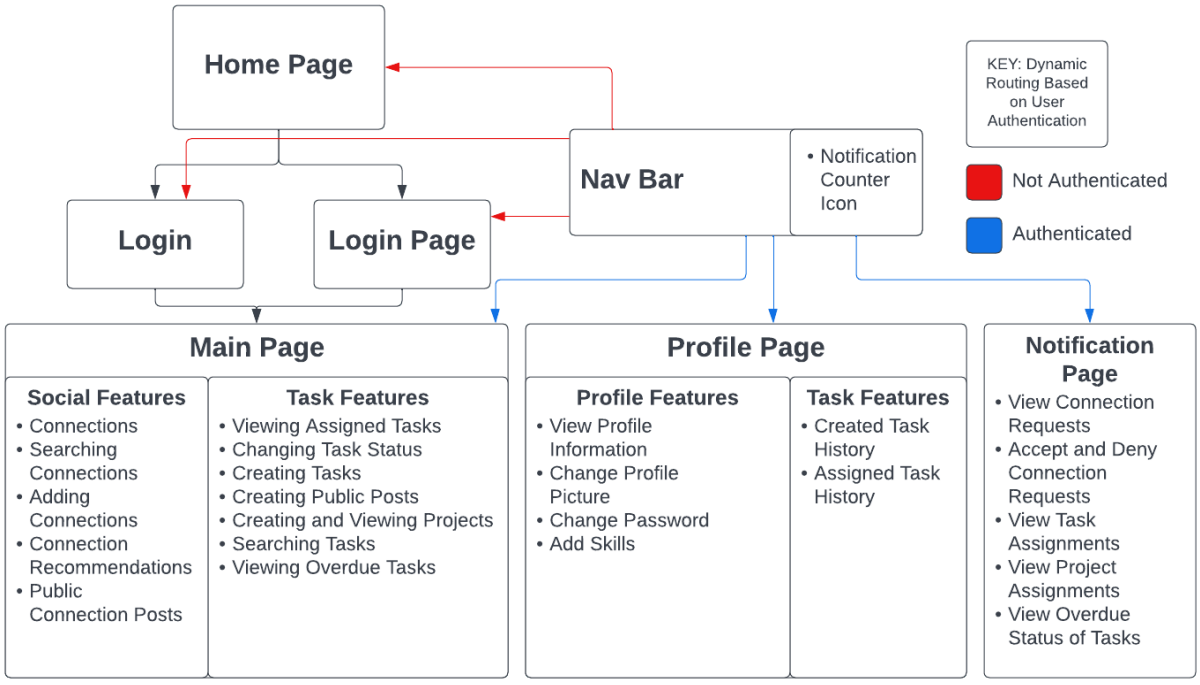
There are several primary pages used as navigation points around the application. The home page, main page and profile page, all linked by a navigation bar. The home page, as shown, directs to the login and register forms, allowing user authentication. Then, once authenticated, the user is taken to their main page, where they will spend the majority of their time.



The main page is designed for ease and convenience of use. The user can navigate around the site using the top navigation bar; i.e. to access the main page, notifications and their profile. Then, they can navigate through the



different sections of the app with the tab interface, with social features on the left panel and task features on the right panel. This simplistic navigation system keeps the app at a minimum number of routes and pages, and instead allows users to quickly and dynamically load all their required information on a single page. Below is a diagram showing links between all the 'single pages' in the application and all the different interfaces they contain. Each specific page provides maximum utility for their designated role, providing users with a convenient experience.



3. Functionalities

<table>
<tr><td colspan="2">Epic/Objective</td></tr>
<tr><td rowspan="4">User Story</td><td>Overview Functionality</td></tr>
<tr><td>Frontend Functionality</td></tr>
<tr><td>Backend Functionality</td></tr>
<tr><td>Database Functionality</td></tr>
<tr><td colspan="2">1. Task masters must be able to maintain a profile where they specify their name and contact details (email address).</td></tr>
<tr><td rowspan="4">1.1. As a user, I want to be able to register and sign in as a task master so that my details (name, email, skills, connections, tasks) can be saved</td><td>They enter a valid email which is checked by regex, first and last name, password, confirm their password which requires strong password check such as length of 8, 1 number, 1 capital and lowercase letter, which is inputted in the frontend and then stores the user in the database after going through the backend.</td></tr>
<tr><td>- Register form to get user information, including first name, last name, email, password and skills
- Login form to authenticate a user through their username and password
Components: LoginForm, RegisterForm</td></tr>
<tr><td>- Get the information passed from the frontend and then make a user into the database if all the information is valid and log the person in.
Functions: signup()</td></tr>
<tr><td>- Created user table, user skills table and friends table
- Created database functions to add a new user
Functions: addUser(), getUser(), viewFriends()</td></tr>
<tr><td rowspan="3">1.2. As a user, I want to be able to login to the site as a task master so that the details and my previous task can be saved in my profile to remember where I left off.</td><td>The user enters their email and password which they made their account with in the frontend, the backend checks if a user exists through the database and then logs the person on.</td></tr>
<tr><td>- Login form to authenticate a user.
Components: LoginForm</td></tr>
<tr><td>- Get the email and password from the frontend and if it corresponds to a user in the database creates a token for the user and logs the user in.</td></tr>
</table>

	Function: generate_toke(), login()
	- Same as 1.1 *Functions: viewFriends(), getCreatedTasks(), getAssTasks()*
1.3. As a user, I want to be able to log off the site so that I can keep my details safe from anyone accessing my computer.	The logged on person would click the log off button and the token is removed from the user and then returns to the home page.
	- Logoff button which clears the user's information and returns them to the home page *Components: Logout*
	- When a logged in user presses the log off button it gets the users token and removes the token and logs the users off. *Function: decode_token(), logoff()*
	- Same as 1.1
1.4. As a user, I want to be able to change my password so that if I do forget it I can retrieve my profile.	The user will need to enter their old password and the new password they want to change. If the new password does not satisfy the security standard, the password will not be changed. Otherwise, the new password will be set up. If the user forgets their password, the system will set a new password and send it to their email.
	- A forgot password form which takes in the user email and sends them a new password for logging in - A change password form which takes in the old password and a new password to change the password to *Components: PassRecover, PasswordChange*
	- Change password: - Check if the new password from the front end is available then replace the old password with the new password. - Forget password: - Set a random 8 characters password for user - Create a server to send a new password to user's email *Functions: change_password(), hash_password(), retrieve_password()*
	- Created database function to modify password field of user table

	Function: changePassword()
1.5. As a task master I want to be able to view my profile to see the details in my profile.	By clicking the profile button in the frontend they are able to see the details in their profile such as name, email profile, picture, skills and past tasks. The database retrieves those informations and passes them to the frontend.
	- A profile page which displays all information about the current user including names, email, profile picture, skills and past tasks *Component: MyProfile*
	N/A
	- Created database functions to get user information from user table - Created database function to get friends from friends table - Created database function to get skills from user skills table *Functions: viewFriends(), getCreatedTasks(), getAssTasks()*
1.6. As a task master, I want to be able to add a profile picture to my profile so that I can be identified easily.	The default profile picture will be set when the user signs up. Each user will be able to see the profile picture when they view their own or their friends' profile pages.The profile picture can be added at the profile page. (This is a new user story we added after the proposal.)
	- All registered users are given a default profile picture which can be changed *Component: signup()*
	- Get the user from the token - Check the image type, accept only JPG and PNG files - Remove the existing file and replace it with the new file *Functions: changeProfilePictureToken(), decode_token()*
	- Same as 1.1
1.7. As a task master, I want to be able to change my profile picture so that I can customize my profile.	The user can change their profile picture at their own profile page.
	- A button on the profile page which takes a file (jpg or png) as input and sends the new profile picture to the backend

	Components: MyProfile *Function: changePicture()*
	- Same as 1.6
	- Created database function to change profile picture field of user table *Functions: changeProfilePicture()*
2. Each task master must be able to view details on the profiles of any other connected task masters.	
2.1. As a task master, I want to be able to view tasks of people I am connected with, to track their progress	In their connections list if a user clicks on their connection they are able to see the list of assigned tasks of connection.
	- A tab on connection profiles which displays a list of their currently assigned tasks *Components: ConnectionTasks*
	Functions: getConnectedTask(), decode_token()
	- Same as Epic 1
2.2. As a task master, I want to see who I am connected with in order to view their task and profile.	Users on the side can see the list of connections they have added previously in the main page.
	- A list of connections, where clicking on a connection displays their profile and tasks *Components: Connections, Profile*
	N/A
	- Same as Epic 1
3. Every task master can become a project manager and must be able to create a project including a title, description, and add people to their project. Projects are a collection of tasks that are related to each other to accomplish a certain goal or large assignment.	
3.1. As a task master, I want to be able to create a project to become a project manager so that I can organize my tasks efficiently.	The user becomes a project manager when they create a new project automatically.
	- A button allowing users to create new projects. The user creating the project becomes the project manager. *Components: NewProject, ProjectList*

	N/A
	- Created project table - Created project manager table - Created project member table - Created a database function to add a new project *Functions: addProject(), viewCreatedProjects(), viewAssignedProjects()*
3.2. As a project manager, I want to be able to create projects with title and description so that I can manage all my projects easily.	Each user can create their own project with title and description. The creator can add their friends into the project as members.
	- A form which takes in all the information to create a new project including title, description and assignees *Components: NewProject*
	- Get the user from the token - check if the members to be added are project creators' friends - Add a new project to the database - Send a notification to all the added members *Functions: createProject(), decode_token(), sendAddedRToProject()*
	- Same as 3.1
3.3. As a project manager, I want to be able to add people in my connections to my project so that we can collaborate together on the tasks.	When creating a new project, the creator can add their friends into the project as project members. All the project members can create tasks in the project.
	- A search bar and form allowing project managers to search through their connections and add people as member to the project *Components: ProjectInfo* *Function: addMember()*
	- Check if the user is a project creator - Check if the members to be added are project owners' friends. - Add members into the project - Send a notification to all the added members *Functions: addMemToProject(), decode_token(), sendAddedToProject()*
	- Created database function to add to project member table

	Functions: addUserToProject()
3.4. As a taskmaster, I want to be able to search for suitable taskmasters in the projects so that I can adequately assign my project tasks to others.	Each member in the project can search through the search bar to find other members in the project.
	- A search bar which allows users to search through all members and managers of a project *Components: NewTask, EditTask* *Function: searchConnections()*
	- Get the user from the token - Find a list of members through the search string *Functions:searchProjectMemberToken(), decode_token()*
	- Created database function to search for users *Functions: searchTaskNameData()*
3.5. As a project master, I want to be able to promote a member of the project to project master so that I can distribute the responsibility of management of the project to other people.	In the project page, if a user clicks on a project and they are a project manager, they can promote a team member to a project manager if they are not a project manager already.
	- A button accessible to project managers which allows them to promote members to managers *Components: ProjectInfo* *Function: promoteMem()*
	- Create a function to get the user inside a project and promote another user inside the project to project manager given they are not manager already. *Function : ptomoteProjectManagerToken()*
	- Created database function to add entry to project manager table *Functions: promoteProjectManager*
4. Project managers are the owners of projects, they must be able to manage all tasks created in their projects even if they did not create the task. The system will notify the project managers when there are changes in their projects.	
4.1. As project manager, I want to be notified when a task is created in the project to approve that the task is relevant to the project.	When a task is created in the project, all project managers receive a notification a task has been created in their project in the notification page, they can approve or deny the task if they deny it the task is deleted.
	- A notification icon displaying number of notifications and notifications page showing the notification(s)

	- Buttons, within the notification itself, to approve or deny a new task created in the project *Components: NavBar, Notifications*
	- When a task is created in a project, send a notification to the creator of the project. *Functions: sendCreateTaskProjectNotif()*
	- Created notifications table - Created function to add entry to notification table - Created function to remove entry from notification table *Functions: addNotification(), viewNotif(), removeNotif()*
4.2. As a project manager I want to be notified when a task is approved as completed by the task master creator so that I can keep track of the project.	When a task is marked as complete a notification is sent to the project managers to see.
	Same as 4.1
	- When a task is completed a project sends a notification that a task has been marked as complete. *Functions: sendTaskCompleteProject()*
	Same as 4.1
4.3. As a project manager, I want to be able to change the status of each task so that I can maintain the progress of the project.	Any project manager is able to change a task's status inside their project.
	- A button on each task in a project which allows for a status change by clicking options in a dropdown menu *Components: AssignedTaskInfo* *Function: handleSelect()*
	- Project managers are able to change any task status in a project. *Functions: changeProjectTaskStatus()*
	- Created database function to modify task table *Functions: changeTaskStatus()*
4.4. As a task master/project manager in a project, I want to be able to create a task in the project so I can progress with the project.	The project creator and all the members in the project can create tasks with title, description, and deadline and assign them to each other in the project.
	- A button in the project which opens a form to create a new task in the project *Components: ProjectInfo, NewTask*

	- Get the user from the token - Check the title and description is not None - Check the task assignees are all in the project. - Add a new task to the database with projectID - Send a notification to all the assignees - Send a notification to the project manager *Functions: addNewTaskToProject(), decode_token(), checkMembers(), sendCreateTaskProjectNotif(), addNotification()*
	- Created a task table with the field "projectID". Project ID can be None if no project is associated with the task. *Functions: addTaskToProject()*
4.5. As a project maser, I want to be able to delete a project so that it can be considered as complete.	Any project manager is able to click the delete project button on the project details page which deletes the project, task associated with project and notification.
	- A button accessible to only project managers that allows them to delete the project upon clicking *Components: ProjectInfo* *Function: delProject()*
	N/A
	- Created database function to remove entry from project table and remove all associated other entries (project members, project managers, tasks, notifications) *Functions: deleteProject()*
4.6. As a task master, I want to be able to search for suitable task masters in the projects so that I can properly assign my project task to others	Users are able to search through project members to assign new project tasks to including themselves and project members.
	- Search bar component which searches through project members and returns members matching the search query *Components: ProjectInfo, NewTask*
	- Get the user from the token - Find a list of members through the search string *Functions: searchProjectMemberToken(), decode_token()*
	- Created database function to search through members in a project

	Functions: searchTaskMasterProject
5. Each task master must be able to request a connection with 1 or more other task masters (by email address), and accept or decline connection requests from other task masters (two task masters are connected if one of them accepted a connection request from the other).	
5.1. As a task master, I want to be able to connect with other task masters so that I can collaborate.	The user will need to send a friend request by email, and the other user will receive a friend request and get notification. Once the receiver accepts the friend request, they build a connection.
	- A form which takes in an email input to send a connection request to the user with that email *Components: AddConnection*
	- Check if the receiver email is already in the sender's friend list - Check if the friend request has been sent before - Send a friend request *Functions: send_request()*
	- Same as epic 1 - Created database function to add entry to friends table - Created database function to modify fields in friends table *Functions: SendFriendReq(), acceptFriendRequest(), viewFriendRequest(), viewFriends()*
5.2. As a task master, I want to be able to search for suitable task masters in my connections so that I can properly assign my individual tasks to others.	When creating a task the search would look through connections to find connections that match the search criteria to add to the task.
	- A search bar which searches through all connections of a user, and upon clicking the results of the search shows the information of the target user *Components: ConnectionSearch*
	- Check the user token - Search the task master by userID *Function: searchTaskMasterName(), decode_token()*
	- Created database function to search through connections *Functions: searchTaskMasterNameData()*

<table>
<tr><td rowspan="3">5.3. As a task master, I want to be able to accept the connection sent by another task master so that we are connected.</td><td>The user sends a friend request. Once the receiver accepts the friend request, they build a connection. The sender will receive a notification if the user accepts the friend request.</td></tr>
<tr><td>- A notification informing the user that another taskmaster has sent them a connection request
- Buttons, within the notification, allowing the user to accept or deny the connection request
Components: NavBar, Notifications</td></tr>
<tr><td>- The receiver accepts or declines the friend request by email
Function: accept_request()</td></tr>
<tr><td colspan="2">- Same as 5.1</td></tr>
<tr><td colspan="2">6. Task masters must be able to create a task, including a title, description, optional deadline, and assign it to either themselves or one of the task masters that they are connected with (each task must also have a system-assigned id that is shown, and is to be assigned to the creating taskmaster if not assigned to anyone else).</td></tr>
<tr><td rowspan="3">6.1. As a task master, I want to be able to create tasks with title and description so that I can assign them to my connection for completion</td><td>All users can create tasks with titles, descriptions, and deadlines. When creating tasks, the creator can assign tasks to the other users in their friend list.</td></tr>
<tr><td>- A form which takes in all task information including title, description, due date, skills and assignees to create a new task
Components: NewTask</td></tr>
<tr><td>- Get the user from the token
- Check if title and description are None
- If the assign list is empty, assign the task to task creator
- Check if all emails in the assign list are task creator’s friends
- Add a new task to the database
- Send a notification to all the assignees
Functions: createTask(), decode_token(), addNotification()</td></tr>
<tr><td colspan="2">- Created task table with relevant fields
- Created assTask table with relevant fields
- Created database function to add a new task to the task table
- Create database function to add assignees to the assTask table</td></tr>
</table>

	- Create database function to get all tasks of specific users *Functions: newTask(), addAssignee(), editTaskData(), getTaskFromID()*
6.2. As a task master, I want to be able to assign a deadline for the task that I made so that I can deliver my work on time.	The deadline is optional when creating a task.
	- An optional form input of a due date upon creating a task or editing a task *Components: NewTask, EditTask*
	- Same as 6.1
	- Same as 6.1
6.3. As a task master, I want to be able to see the system assigned ID of the task so that I know which task I have to do and is distinguishable.	The task will automatically assign when creating a new task.
	- Task cards displaying task information contain the task ID at the footer *Components: AssignedTaskInfo, CreatedTaskInfo, PubTask*
	N/A
	- Same as 6.1
6.4. As a task master, I want to be able to edit my task so that I can change details and deadlines based on changing work conditions.	After creating a task, the task creator can edit the title, descriptions and deadline at any time.
	- A button available to task creators, opening a form allowing them to edit the task by changing any of the fields *Components: EditTask*
	- Get the user from token - Check if the user is the task creator - Edit task by the input information *Functions: editTask(), decode_token()*
	- Created a database function to edit the fields in the task table *Functions: editTaskData()*
7. To progress the state of a task, a task master must be able to update the status of a task they have created or that they have been assigned. Valid task states include "Not Started" (start state for new tasks), "In Progress", "Blocked", and "Completed".	

7.1. As a task master who created a task, I want my task to be “Not Started” when I create the task so that I know no progress is being made on the task.	When a task is created the status is automatically set as “Not Started” to indicate a new task with no progress.
	- Upon task creation, the default status is “Not Started” *Components: NewTask*
	- When a task is created the status is automatically set as “Not Started”.
	- Same as epic 6
7.2. As a task master who has created a task and assigned it to someone else, I want to be able to approve or deny the completion of a task to ensure quality of work and to allow removal of the task.	When a task is marked as complete a notification is sent to the task creator when they approve it is put into the past task and if denied it moves back to in progress status.
	- A notification is sent to the creator of a task when the task is marked as ‘complete’ by assignees - Buttons, within the notification, allowing the creator to approve or deny the completion. Approving removes the task, denying changes the status back to ‘in progress’ *Components: NavBar, Notifications*
	- When the assignee marks a task as complete the frontend sends a notification and if the creator of task marks it as complete it is added to past tasks. *Function: changeTaskStatusCheck*
	- Same as 4.1
7.3. As a task master who is working on the task, I want to be able to change the progress status of the task from “Not Started” to “In Progress” so that other people can see I started the task.	A task assign or creator can click the status drop down menu in a Not Started task to In progress to indicate progress is being made in the task.
	- A button, available to assignees of a task, allowing them to change the status of a task by choosing from a dropdown menu of options *Components: AssignedTaskInfo*
	- Created a function that checks if a person is the creator or an assignee then if the task is in Not Started if all are true it calls the change task status to In progress for the backend. *Function: changeTaskStatusCheck()*
	- Same as epic 6

	- Created database function to change status of task *Functions: changeTaskStatus()*
7.4. As a task master who created a task, I want to be able to mark a task “In Progress” or “Not Started” to “Blocked” in order to put the task on hold.	A task assign or creator can click the status drop down menu in a Not Started or In Progress task to Blocked to indicate the task is not making progress or is on hold.
	Same as 7.3.
	- Created a function that checks if a person is the creator or an assignee then if the task is in Not Started or In progress if all are true it calls the change task status to In progress for the backend. *Function: changeTaskStatusCheck*
	- Same as epic 6 - Created database function to change status of task *Functions: changeTaskStatus()*
7.5. As a task master, I want to be able to mark tasks from “In Progress” to “Completed” that I have been assigned as completed so that the original creator is notified that the task is completed.	A task assign or creator can click the status drop down menu in a In Progress task to Completed to indicate the task is done. The notification of task completion will be sent to the task creator.
	Same as 7.3.
	- Creator and assignee of task can change status. - The completed task notification will send to the task creator *Functions: changeTaskStatus(), CompleteTaskNotification()*
	- Same as epic 6 - Created database function to change status of task *Functions: changeTaskStatus()*
7.6. As a task master, I want to be able to mark tasks I have been assigned to from “Blocked” to “In Progress” or “Not Started” so that I can resume an on hold task.	A task assign or creator can click the status drop down menu in a Blocked task to In Progress or Not Started to indicate progress is being made in the task.
	Same as 7.3.
	- Creator and assignee of task can change status. *Function: changeTaskStatusCheck*
	- Same as epic 6 - Created database function to change status of task *Functions: changeTaskStatus()*

7.7. As a task master, I want to be able to remove my own tasks that have been completed so that I can ensure quality.	A task creator can click the delete button and the task would be removed in the database and all assignees would be unassigned.
	- Tasks can be deleted manually through a button in the edit task form, or removed automatically after approval of completion *Components: EditTask, Notifications*
	- Calls the database function *Function: deleteTask()*
	- Same as epic 6 - Created database function to delete entry from task table *Functions: deleteTaskFromData()*
8. Over time, the number of tasks existing in the system will build up, so it must be possible for a task master to search, through all tasks assigned to themselves or any task master they are connected to, by any combination of id, name, description and/or deadline, and view the full details.	
8.1. As a task master, I want to be able search for my own assigned task using filters by combination of id, name, description and/or deadline so I can save time to view its details	The user can search through the search bar by typing in the search string in their assigned tasks database. The search string can be a combination of id, name, description and/or deadline.
	- A search bar which takes any input and compares the search query to all fields including id, name, description - Search returns a list of tasks based on how many matching fields there are, and upon clicking a task in the returned list, displays that task's information *Components: AssignedTaskSearch*
	N/A
	- Created database search function for assigned tasks *Functions: searchAssTasks(), searchCreatedTasks()*
8.2. As a task master I want to be able to view other task masters profiles so that I can add them to my assignees for help with my task.	A user can click on one of their connections in order to view the details of their connections and view their email, name and skills.
	- In the list of connections, clicking on a user displays their profile

	Components: Profile, Connections
	N/A
	- Same as epic 6
8.3. As a task master, I want to view the tasks that I have created using filters by combination of id, name, description and/or deadline so I can see what they are doing	The user can search through the search bar by typing in the search string in their created tasks database. The search string can be a combination of id, name, description and/or deadline.
	Same as 8.1 except with created tasks instead of assigned *Components: CreatedTaskSearch*
	N/A
	- Created database search function for assigned tasks *Functions: searchCreatedTasks()*
9. Task masters can add skills to their profile as well as skills to the created task in order to match appropriate task masters with appropriate tasks.	
9.1. As a task master I want to be able to add skills to my profile so that my connection can know my expertise.	A user can go to their profile page and add individual skills to their profile, if the skills don't already exist in the profile.
	- In the registration form, the user can enter skills to be added to their profile - Users can go to their profile and add more skills after signing in through an input on the profile page *Components: RegisterForm, MyProfile*
	- Created a function where the person adds a list of skills they possess which are unique into their profile and calls the database function to add it into the database. *Functions: addSkillsUserToken*
	- Created user skills table - Created database function to add skills to user - Created database function to view user skills *Functions: addSkillsUser()*
9.2. As a task master, I want to be able to add skills required to my created task to match	A user can add skills during task creation or edit skills in order to indicate a specific skill is required for a task.
	- In the task creation form, there is an input field to enter the skills required for the task

appropriate connections to skills.	- In the task edit form, there is an input field to add additional skills to a task *Components: NewTask, EditTask*
	- Created a function where the person adds a list of skills required for a task that are unique and calls the database function to add it into the database. *Functions: addSkillsTaskToken()*
	- Created task skills table - Created database function to add entry to task skills table *Functions: addSkillsTask()*
10. To view the work to be done by a task master, their profile must also include an "Assigned Tasks List", showing all of their assigned tasks with each task showing summary details of the task ID, title, and deadline (if specified), sorted by deadline (earliest to latest deadline, with tasks not having a deadline appearing last).	
10.1. As a task master, I want to be able to view all my assigned tasks list so that I can complete them on time.	When a user goes to the main page they are able see the list of task they have been assigned to and is sorted in the order of the due deadline, no deadline are at the bottom of the list
	- A list showing all assigned task titles, organized into four columns based on their status. Clicking a task in the list will show the task's information. *Components: AssignedTaskList*
	- Created a function that gets a user's assigned task then sorts them in the order of the deadline. *Function: getAssignedTask()*
	- Same as 6.1 *Functions: getTaskFromID()*
10.2. As a task master, I want to be able to see all my tasks sorted from the earliest deadline to the latest deadline so that I can prioritize the closest deadlines first.	A user is able to see both their created task and assigned task sorted by the deadline and the no deadline is at the bottom.
	- Task list is organized by deadline, with earliest due at the top *Components: AssignedTaskList*
	- Same as 10.1
	- Same as 6.1 *Functions: getTaskFromID()*

10.3. As a taskmaster, I want to receive a notification when I am assigned a task so that I can keep up to date with my task	The users will receive a notification if they are assigned to a task.
	- Notification icon and notification in notifications page which allows user to see who assigned them what task *Components: NavBar, Notifications*
	- Get the user from token - Send a notification with notification type '0' and the taskID *Functions: assignTaskNotification(), decode_token()*
	- Created function to add entry to notification table - Created function to remove entry from notification table *Functions: addNotification(), viewNotif(), removeNotif()*
10.4. As a task master, I want to receive a notification when a task I am assigned to is past the due date so that I can keep on top of my task	The user will get a reminder notification if there is any task past the due date when they log in.
	- Notification icon and notification in notifications page which informs user they have tasks past the due date - List of tasks past the due date *Components: NavBar, Notifications, PastDue*
	- Get the user from token - Check if there are tasks' deadline is earlier than today - Send a notification if there is a task past due date *Functions: getDueList(), dueNotification(), decode_token()*
	- Same as 10.3
11. The system must be able to show a task master an estimate of how busy each of their connected task masters will be over the next week. From 0% (minimum possible) through to 100% (maximum possible), or "100%+ (overloaded)" for any task masters it thought were overloaded, where this value can be based on a combination of their assigned tasks, task states, task deadlines, how long similar tasks have taken to complete in the past, and/or any other variables that you wish to use/introduce for the purposes of estimating how busy each task master is.	
11.1. As a task master, I want to be able to see how busy my connected taskmasters are so I	When a user clicks on one of their connections a circle would be shown to show the business factor.

can gauge if I should ask them for help.	- A busy factor circular bar shown in a user's profile indicating their busy-ness as a number from 1-100 based on the number of assigned tasks and their due date *Components: Profile, BusyFactor*
	- Created a function to calculate the business factor based on how many assigned tasks they have. *Function: busyFactor()*
	N/A
11.2. As a task master, I want to see a number from 0 to 100 or 100+ visually to gauge how busy my connected taskmasters are easily.	A user on their assigned task page, they can see the number of their busyness factors in a quantitative version.
	Same as 11.1
	- Same as 11.1.
	N/A
11.3. As a task master, I want this number for how busy they are based on the number of tasks they have, the deadlines and state of their task so I can gauge how busy they are accurately.	A user on their assigned task page, they can see the number of their busyness factors in a quantitative version.
	Same as 11.1
	Same as 11.1
	N/A
12. To view the work done in the past by a task master, their profile must also include a history of tasks that were completed by a connection or by the task master themselves showing basic details of the task if the task master needs a way to keep track of any past task they have completed.	
12.1. As a task master, I want to view the past task I completed so that I can keep track of what has been accomplished already	A user can go to their profile page and click on past assigned tasks and view any task they have completed in the past.
	- An assigned task history tab available in a user's profile page showing a list of all their previous assigned tasks which have been completed *Components: MyProfile, AssignedTaskHistory*
	- Created a function to get all the past tasks of the user. *Function: getPastAssTaskToken()*
	- Created database function to show past assigned

	tasks *Functions: getPastASSTask()*
12.2. As a task master, I want to view the past task I created that are completed so that I can keep track of what has been accomplished already	A user can go to their profile page and click on a past created task and view any task they created that has been completed.
	- A created task history tab available in a user's profile page showing a list of all their previous created tasks which have been completed *Components: MyProfile, CreatedTaskHistory*
	- Created a function to get all the past tasks of the user that they have created. *Function: getPastTaskToken()*
	- Created database function to show past created tasks *Functions: getPastTask()*
13. As a new task master in this application I wouldn't have any connections when I start off as well as having no tasks. Task masters should be able to find new tasks and new connections in order to build their network.	
13.1. As a task master, I want to be able to search for task to do so I can be proactive in working	A user can click on a public task and see any task their connections have made a task public to request to be assigned to the task.
	- A public task board where users can post tasks publicly for other taskmasters in their connections to assign themselves to - An option in creating tasks and editing tasks to make a task public or private *Components: PublicTasks*
	- Created a function to get the public task of connections. *Function: getPublicTaskToken()*
	- Created database function to find all public tasks *Functions: getPublicTask()*
13.2. As a taskmaster in a project, I want to be able to search for available tasks in a project so that I can be proactive in a project.	A user can click on a project public task inside and see any task their project members have made a task public to request to be assigned to the task.
	Same as 13.1. except within the scope of the project, and all project members can see project public tasks regardless of

	connection status with the creator
	- Created a function to get the public task of a project. *Function: getPublicTaskProjectToken()*
	- Created database function to find all public tasks within a project *Functions: getPublicTaskProject()*
13.3. As a task master, I want to see the task search to be sorted by the relevant skills that I have so that I can easily identify tasks that I can do.	In project and public tasks the task is sorted in the numbers of skills that the user has.
	- Public tasks are ordered by the number of skill matches they have with the user *Components: PublicTasks*
	- Both functions are sorted by the number of skills the user has and the task is looking for. *Functions: getPublicTaskProject(), getPublicTask()*
	N/A
13.4. As a task master, I want to be able to find non-friend taskmasters that have a specific skills so that I can add them into my connections and expand my network	A user is able to create a post to look for friends, they can attach skills to the post to specify what skills or people they are looking for and non-connections can see the post and add the person as a connection if they would like.
	- A public post board where users can create posts to advertise that they are looking for connections, with a description and a list of relevant skills they are looking for - Taskmasters viewing the public post board can send a connection request to people who have posted by clicking a button on the post *Components: NewPost, AllPosts, MyPosts, FindConnections* *Functions: SendReq()*
	- Created a function to add a post to find a connection. - Created a function to get all the find connection posts that are not your connections. - Created a function to get all the posts you created to find a friend. - Created a function to delete a post that you created. *Functions: deleteFindConnectionsToken(), findMyConnectionPostToken(), findConnectionToken(), addFindConnectionsToken()*

	- Created friend search post table - Created database function to get all posts - Created database function to get all posts created by user *Functions: findMyConnectionPost(), deleteFindConnection(), findConnectionPost()*
13.5. As a task master, I want to have recommended connections so that I may add connections that I may know	This function is used to recommend connections that your connections have which is then sorted by the number of mutual connections those recommendations have.
	- A recommended connections list ordered by the number of mutual connections - Users can send connection requests to the taskmaster by clicking a button on the recommendation *Components: ConnectionRecs, FindConnections*
	- Created a function to get all your recommended connections from your mutual connections and then sort it by the number of mutual connections those people have. *Function: friendRecommendation()*
	N/A

4. Third Party Functionalities

4.1. SQLAlchemy

In order to integrate postgreSQL databases with python, the API SQLAlchemy is used. This library provides all the functionality needed to interact with a SQL database wrapped up neatly into python functions. SQLAlchemy also integrates very nicely into the flask application allowing for a seamless connection between the two languages.

4.2. Back-end

- *Flask*
  In order to build a web application by using Python. Flask is used. Flask is a web framework. It is a nice choice for developing python web applications.
- *Hashlib*
  To increase the system's security, we store the hashed passwords in the database. The md5 algorithm from the hashlib is used. It prevents the password from being modified by the unknowing others and keeps the user's account safe.
- *PyJWT*

PyJWT is a library that encodes and decodes json web tokens in order to use for authorization to ensure the security of the user and the system.

- *PIL*
  PIL(Python Image Library) is a library that deals with images in python. The Image in PIL is used. We use this for opening and saving images when setting the profile picture.
- *Regex*
  We used regex to ensure that the email is a valid email that follows the format of any standard email.

4.3. Front-end

- *React*
  Provided the framework to produce a web application using JavaScript, bypassing the need to directly code in HTML.
- *React Bootstrap*
  Provided many tools for CSS manipulation within React to produce an effective minimalistic design. It was used primarily for lists, cards, tabbing interfaces and buttons.
- *React-Router-DOM*
  Provided the ability to create navigation routes within the app, to display different components based on the routes.
- *Material UI*
  Provided tools for graphic design including displaying avatars and icons following contemporary web design conventions, producing a clean user interface.
- *React Password Strength Bar*
  A component to measure the strength of a password. Based on the zxcvbn library which contains an extensive list of common and weak passwords. Allows for users to have more secure passwords.
- *React Circular Progressbar*
  A component primarily used to display a user's busy factor. It is a dynamic, animated circular bar which changes based on a user's busy rating, allowing for constant feedback to the user that is minimal in design but maximal in information.

5. Implementation Problems/Tricks

5.1. Database

Special care is taken whilst designing all the components of the database system to ensure correct functioning. Two of the most interesting implementation choices manifest themselves in the notifications subsystem and the user-friend subsystem.

In our system there are 3 main types of notifications that can pop up:

- Notifications related to friend requests
- Notifications related to tasks

- Notifications related to projects

In the design of this project, the friend request notifications are set up completely separately from the task and project notifications. This is due to the fact that friend notifications are very different and will require different fields and different implications. Friend request notifications require an addressee and sender whilst task and project notifications require a connection to a user, a task and possibly a project.

Friend requests and friend connections are both setup in the friends table. The friends table contains 3 fields, addressee, sender and status. When a user sends someone a friend request, an entry is added into the friends table with the relevant addressee and sender. The status of an unconfirmed friend request is set to 0. Once the addressee accepts the friend request the status changes to 1. Hence this table will handle both friends and friend requests. The only issue is that when checking for a user's friends, we must check that the status is 1 and that the user is either the sender of the addressee. This is because once two users are friends the relationship is reflexive. Instead of storing the same information twice, we will simply check both the addressee and the sender.

Since the task and project notifications are similar they are both stored inside the notifications table. This table has fields user (who is receiving the notification), notifType (integer value determining type of notification), taskID (task related to notification) and projectID (project related to notification). All notifications will have a task associated with them however they may not have a project associated with them. Therefore the projectID field is set as nullable allowing a notification with no projectID to be set.

5.2. Back-end

Implementation problems that we encountered during this assignment was when we were trying to pass in invalid data that the user would pass in from the front-end, small bugs appeared in the system that we have to solve especially during merging both the frontend and the backend. We solved the invalid data by making sure the backend and the database would only accept valid data into the system that the user inputs in the frontend. This ensures the system would not produce unexpected errors. During the merging of the backend and the frontend of the system, the first sprint took longer since there were alot of bugs and inconsistency during the merge. We had to spend a large amount of time to go through each error and make sure the merge was successful. We solved this problem by making sure that after the first sprint all the functions we write would have unit tests to ensure that it produced the expected result. We also met at the start of each sprint to ensure that we are on the same page in terms of functionality by writing stubs for the project and letting everyone know what it returns and better commenting, making the merge a smoother experience.

Another issue we had to go through would be the authorization for the project of how to log a user and ensure that hackers and malicious people would not misuse this system. We solved this by encoding JSON Web TOkens using PyJWT for each user to ensure identity. We also hashed the password so it would be more difficult for people to get the password of the user, but this was a issue as we store the hash of the password and not the password for security reasons so as a result if the user forgot their password it was impossible to get back the password as you cannot unhash a hashed string. We solved this by sending the user a new password when they forgot the password so that they are able to log back into their account and then change their password if they wish to.
One function that took a bit more technical expertise is the friend recommendation and sorting functions. Since the sorting required another variable to search, we used a dictionary to record the information we have to sort by and the associated id. Then we used a lambda function to sort by the matching information. For the friend recommendation system we made it simple by making it only 2 friends deep, and looking for the mutuals of friends then looking into mutuals of those friends. This would then require us to take the set of all those people and subtract the set of the user's friend and the user itself to ensure that the list is unique and transform them back into a list. Use sorting by the number of mutual friends that each of them have to ensure that the best matching recommendation was produced.
The final challenge for the backend was when we were building a new functionality such as having multiple project managers we had to change the whole systems and all the functions that were related to the new system as it affects all of them. It would produce bugs for the system as the new functionality would not make the system work with the old one. To solve this we made sure that we were peer programming to ensure that when we are looking through the code we can catch all the mistakes there are in the old system when implementing a new functionality.

5.3. Front-end

Two primary hurdles in designing the frontend were routing and dynamic changes to the UI based on events or changes in the backend.

Routing was initially solved by having each component render on its own route, and have components link to each other through button clicks. This soon became cluttered and unusable. To counter this, most components were extracted to become children of other components, so that they could be hierarchically rendered or rendered in parallel simultaneously, allowing for components to be pooled into several key pages rather than many different pages. This reduced the clutter and overall produced a better user experience.

Dynamic changes to the UI were initially attempted through implementing a web socket. However, our backend routing system was initially not designed with a web socket in mind, and thus it was not feasible to apply a web socket. Instead, polling was utilized to send GET requests to the server at constant intervals in order to ensure that information was dynamically updated on

the frontend. Different components have different polling rates based on their frequency of use to maximize the responsiveness of the UI, however the rates had to be balanced in order to maintain performance and prevent the app from running poorly.

6. User Documentation

6.1. How to run app

Application must be run inside Lubuntu 20.4.1 LTS virtual machine

Requirements:

- Flask 2.1.2
- Python 3.10.4 64 bit
- Werkzeug 2.1.2
- SQLAlchemy 2.5.1
- PyJWT 2.4 (Ensure JWT is not installed into the system)
- NodeJS v16+
- npm v8+

The following steps cover all the installation procedures and how to run the app

How to install and run:

1. Open a terminal in the VM and update the apt, run
   ```
   $ sudo apt update
   ```

2. Install pip
   ```
   $ sudo apt install python3-pip
   ```

3. It is preferable that the python version is 3.10, however the program still functions with 3.8 as it is on the VM

4. Install curl to retrieve the latest version of NodeJS
   ```
   $ sudo apt install curl
   ```

5. Retrieve version 16 of NodeJS and install. Run
   ```
   $ curl -sL https://deb.nodesource.com/setup_16.x | sudo -E bash -
   ```
   then, after retrieving, run
   ```
   $ sudo apt-get install -y nodejs
   ```

6. Install Flask, Flask-SQLAlchemy and Pillow
   ```
   $ pip install -U Flask
   $ pip install -U Flask-SQLAlchemy
   $ python3 -m pip install --upgrade Pillow
   ```

7. Update PyJWT

```
$ pip install --upgrade pyjwt
```

8. Navigate to the 'protask-ui' folder in the unpacked ‘project’ folder
   `project$ cd protask-ui`

9. Install all node js packages
   `project/protask-ui$ npm i`

10. Open a new terminal tab and navigate to the ‘project’ folder. From there, run main.py
    `project$ python3 main.py`

11. In the other terminal tab, in the 'protask-ui' folder, run the react app
    `project/protask-ui$ npm start`

12. The app should open a window with the default browser, and may take some time to load. Once loaded, see user guide.

6.2. How to use the app

First, click “sign up” and fill in all necessary information to create a new account at the home page. If you already have a ProTask account you can sign in directly. As the diagram below:

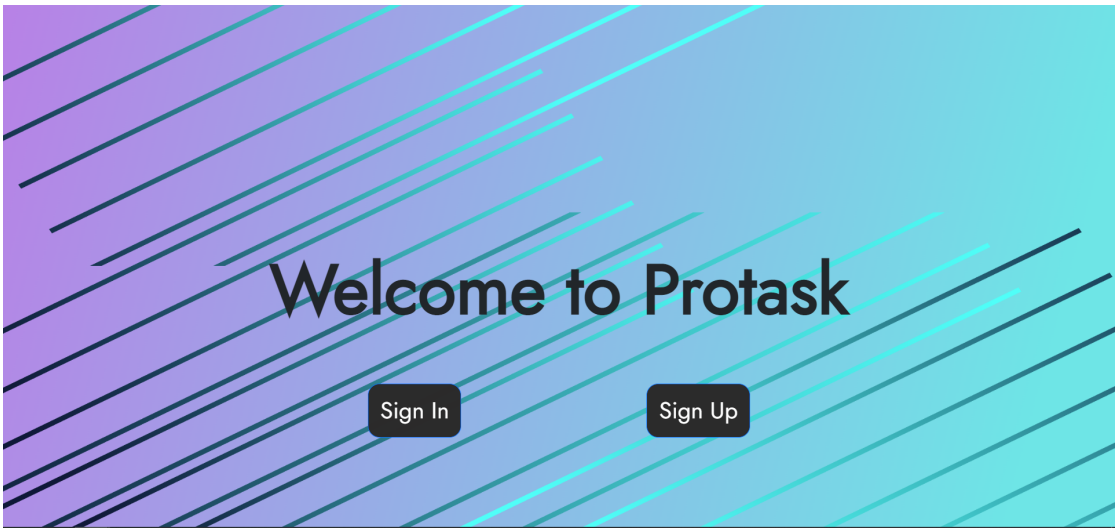


Diagram: Home page

After logging in, you can see the account's main page. You can view and change your personal information (changing profile picture, changing password, and adding new skills) by clicking “profile” at the left of the top. “Logout” and “notification” buttons are at the right of the top. You will receive a notification if there are any new activities in your account. As in the below diagram:

the left-hand side of the account's home is about the friend connections and the right-hand side is the details of your account activities:

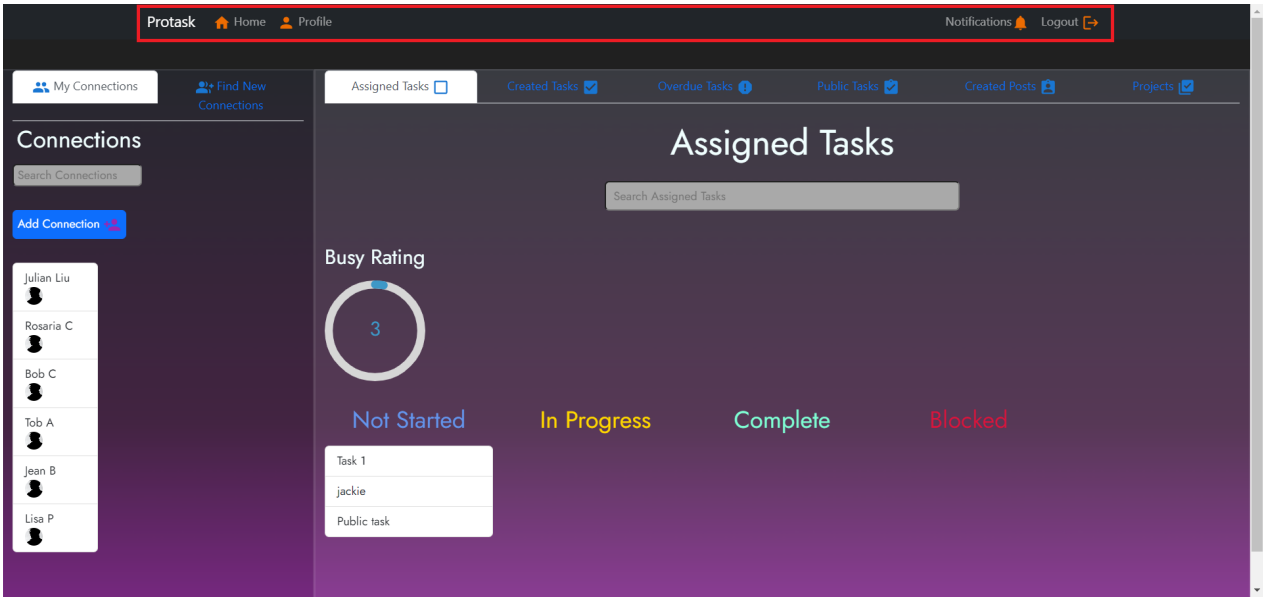


Diagram: Account's Main Page

Connections:

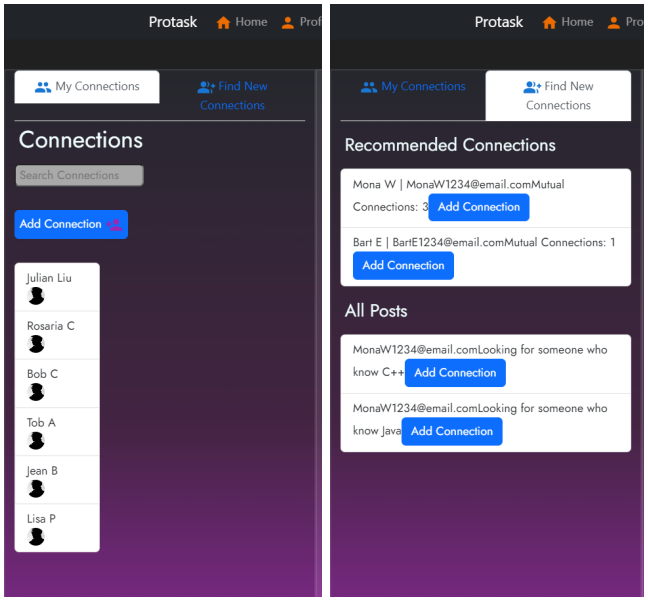


Diagram : Connections

- **My Connections**:
  Display all the connections you have built. You can click "Add Connection" on the top to add a new friend by email. The search bar at the top is for finding your existing friends.
- **Find Connections**:
  Display the friends recommendation and the public post from people who are finding new friends. Click "Add Connection" will send the friend request to the user.

There are 6 different categories for task management:

- **Assigned Tasks**:

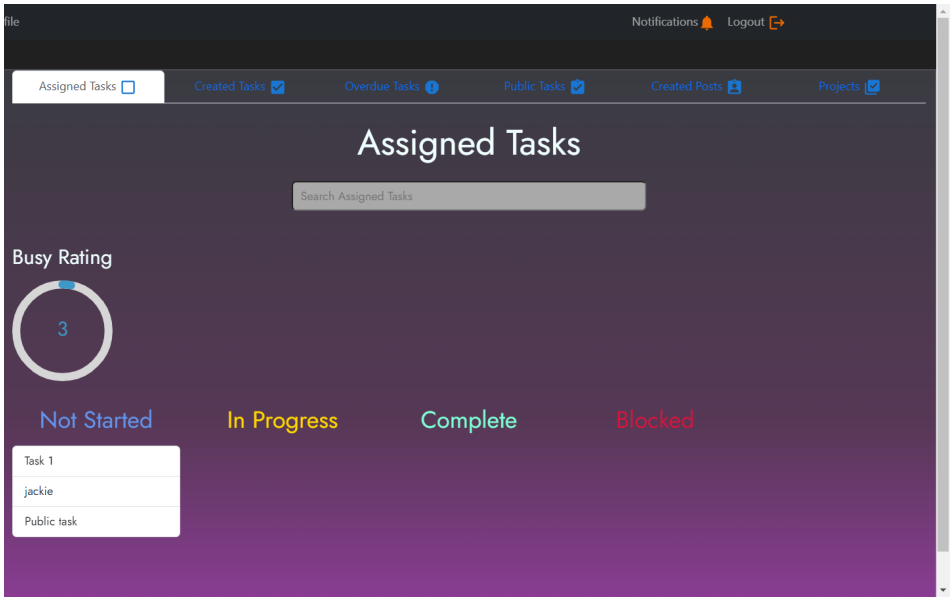


Diagram: Assigned Tasks Page

All the tasks that you are assigned will be displayed on this page. There are four columns "Not started", "In Progress", "Complete", and "Blocked" for tracking tasks' status. Click the task to change the status. You can easily find what to do by using the search bar. The busy rating represents how busy you are in the current situation. If there are more tasks close to the due date the busy rating will increase.

- **Created Tasks**:

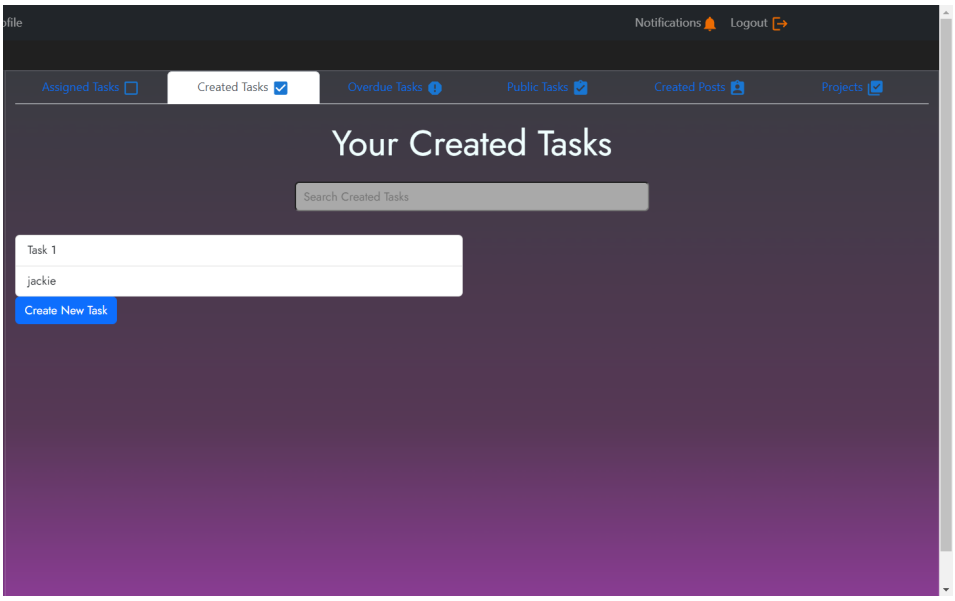


Diagram: Created Tasks Page

All the tasks that you have created will be displayed on this page. You can track the task you created here. If you click the task, you will find the button to add new assignees (only the task creator can add new assignees). Click “Create New Task” to create more tasks.

- **Overdue Tasks**:

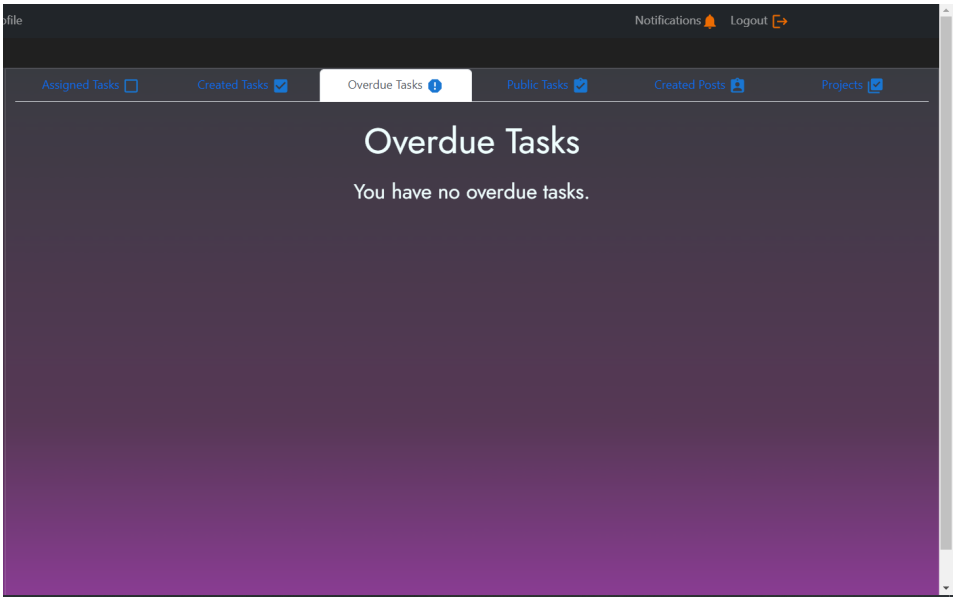


Diagram: Overdue Tasks Page

All the tasks that past the due date will be displayed on this page.

- **Public Tasks**:

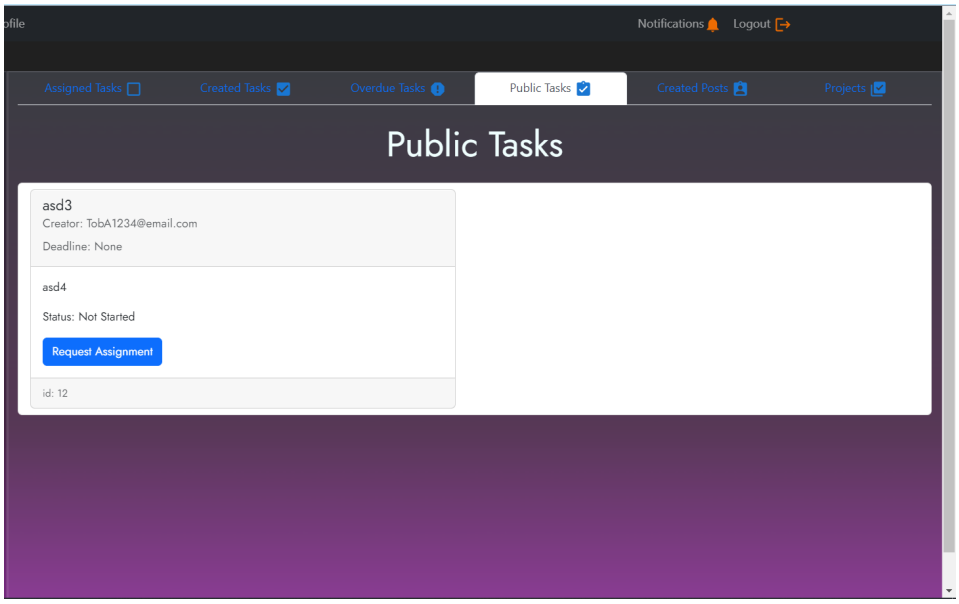


Diagram: Public Tasks Page

If you are interested in joining others' tasks you can find them here. Request the assignment and you will become a part of this task.

- **Created Posts**:

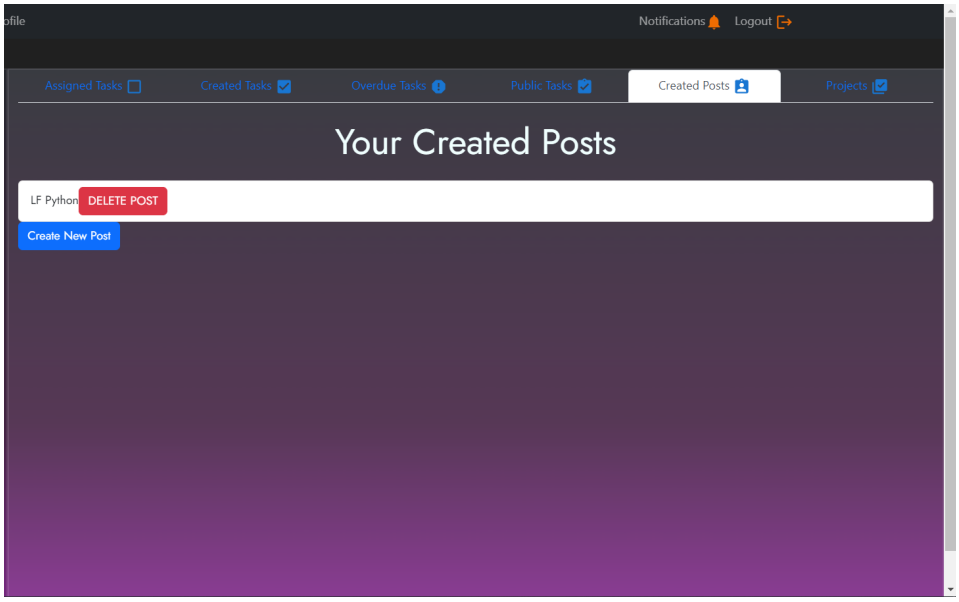


Diagram: Created Posts Page

We provide a place for you to find suitable people for your tasks or project. You can post your requirement here and the other user can send the request to you if they think they meet your requirement. Delete the post any time if you found suitable people.

- **Project**

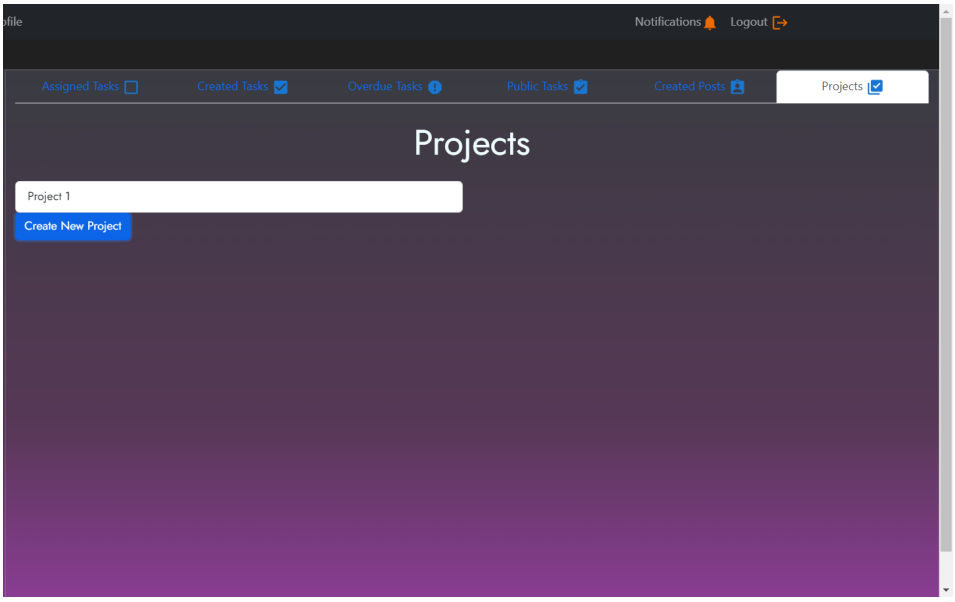


Diagram: Projects Page

You can create a project if you plan to work in a small group with many different tasks. The project page will display all the projects that you created or added. Click each project to see more information about the project. If you click the project, you will find the button of add new members, promote people as a project manager (only the project manager can add new members and promote a project manager) and create new tasks.

7. Reference