Non-technical Documentation for Enterprise Pro Task Management system V2

app.py (Main File)

This is the main file that runs the Flask web application. It handles user requests, manages user sessions, and interacts with the database to perform various tasks like creating projects, assigning tasks, and updating progress.

Class: `myClass`

This class contains all the logic for handling user interactions, such as logging in, creating tasks, and managing projects. It also connects to the database to store and retrieve data.

- `__init__` **Method**:
  - This method initializes the class and sets up the routes (URLs) for the web application. It also connects to the database and creates the necessary tables if they don't already exist.
  - It keeps track of whether a user is logged in, their role (admin, supervisor, or regular user), and their user ID.
- `index` **Method**:
  - This is the home page of the application. If the user is not logged in, they are redirected to the login page. If they are logged in, they see a list of projects they are assigned to, along with the tasks in those projects.
- `setProjectId` **Method**:
  - This method sets the current project ID based on the project the user selects. It redirects the user to the tasks page for that project.
- `login` **Method**:
  - This method handles user login. It checks if the username and password match what’s stored in the database. If they do, the user is logged in and redirected to the home page. If not, an error message is shown.
- `load_tasks` **Method**:
  - This method retrieves all tasks from the database and returns them. It’s used to display tasks on the tasks page.
- `supervisor` **Method**:

    - This method is for supervisors. It shows a list of all projects and their tasks. Regular users cannot access this page.
- `admin` **Method**:
    - This method is for administrators. It allows them to create new projects, users, and update user details. Only users with the "admin" role can access this page.
- `tasks` **Method**:
    - This method displays all tasks for a specific project. It also shows task updates and which users are assigned to each task.
- `search_tasks` **Method**:
    - This method allows users to search for tasks by their title. It uses a binary search algorithm to quickly find the task.
- `create_task` **Method**:
    - This method creates a new task. It takes the task title, details, due date, and assigns it to the current project. The task is then saved in the database.
- `sort_tasks` **Method**:
    - This method sorts tasks based on criteria like title, due date, or status. It helps users organize their tasks.
- `filter_tasks` **Method**:
    - This method filters tasks based on criteria like status (e.g., completed, in progress). It helps users focus on specific types of tasks.
- `update_progress` **Method**:
    - This method allows users to update the progress of a task. It saves the update in the database and displays it on the tasks page.
- `search_projects` **Method**:
    - This method allows users to search for projects by their title. It uses a binary search algorithm to quickly find the project.
- `filter_projects` **Method**:
    - This method filters projects based on criteria like status or team. It helps users find specific projects.
- `sort_projects` **Method**:
    - This method sorts projects based on criteria like title or status. It helps users organize their projects.
- `add_user_to_project` **Method**:
    - This method allows administrators to add users to a project. It updates the database to link users to the project.
- `remove_user_from_project` **Method**:
    - This method removes a user from a project. It updates the database to remove the link between the user and the project.
- `create_project` **Method**:
    - This method allows administrators to create a new project. It saves the project details in the database.

- `create_user` **Method**:
  - This method allows administrators to create a new user. It saves the user details in the database.
- `edit_project` **Method**:
  - This method allows administrators to edit the details of a project. It updates the project information in the database.
- `delete_project` **Method**:
  - This method deletes a project from the database. It also removes all tasks associated with that project.
- `modify_user` **Method**:
  - This method allows administrators to update user details like password, role, or team. It updates the user information in the database.
- `edit_task` **Method**:
  - This method allows users to edit the details of a task. It updates the task information in the database.
- `delete_task` **Method**:
  - This method deletes a task from the database. It also removes the task from the list of assigned tasks.
- `show_deleted_tasks` **Method**:
  - This method shows a list of tasks that have been deleted. It's useful for tracking changes.
- `assign_users` **Method**:
  - This method assigns users to a task. It updates the database to link users to the task.
- `get_user_status` **Method**:
  - This method checks if a user is logged in and returns their role and ID. It's used to control access to different parts of the application.
- `logout` **Method**:
  - This method logs the user out by clearing their session and redirecting them to the home page.

---

forms.py

This file contains the forms used in the application. Forms are used to collect data from users, such as login credentials, project details, and task information.

- `LoginForm`:
  - This form is used for logging in. It asks for a username and password.
- `CreateUserForm`:
  - This form is used to create a new user. It asks for a username, password, role (e.g., admin, supervisor), and team (e.g., police, intern).
- `UpdateUserDetailsForm`:

    - This form is used to update user details. It allows administrators to change a user's password, role, or team.
- `UsersInProjectsForm`:
    - This form is used to add users to a project. It asks for the project title and the usernames of the users to be added.
- `CreateProjectForm`:
    - This form is used to create a new project. It asks for the project title, details, status, review date, and owner.
- `UpdateProgressForm`:
    - This form is used to update the progress of a task. It asks for a progress update.
- `EditTaskForm`:
    - This form is used to edit a task. It asks for the task title, details, status, assigned date, and due date.

---

search_sort.py

This file contains helper functions for searching and sorting tasks and projects.

- `categorise_data` **Method**:
    - This method sorts tasks or projects based on criteria like title, due date, or status.
- `filter_data` **Method**:
    - This method filters tasks or projects based on criteria like status or team.
- `binary_search` **Method**:
    - This method searches for a task or project by its title using a binary search algorithm, which is faster than a regular search.
- `merge_sort` **Method**:
    - This method sorts a list of tasks or projects using the merge sort algorithm, which is efficient for large lists.

---

use_database.py

This file handles all interactions with the database. It creates tables, adds, updates, and deletes records, and retrieves data.

- `create_connection` **Method**:
    - This method connects to the SQLite database. If the database doesn't exist, it creates one.

- `create_tables` **Method**:
  - This method creates the necessary tables in the database, such as users, projects, tasks, and task updates.
- `add_user` **Method**:
  - This method adds a new user to the database.
- `add_project` **Method**:
  - This method adds a new project to the database.
- `add_task` **Method**:
  - This method adds a new task to the database.
- `add_task_update` **Method**:
  - This method adds a progress update for a task to the database.
- `add_user_into_project` **Method**:
  - This method adds a user to a project in the database.
- `add_assigned_task` **Method**:
  - This method assigns a task to a user in the database.
- `find_user` **Method**:
  - This method retrieves a user's details from the database based on their ID or username.
- `find_project` **Method**:
  - This method retrieves a project's details from the database based on its ID or title.
- `find_task` **Method**:
  - This method retrieves a task's details from the database based on its ID or title.
- `update_user` **Method**:
  - This method updates a user's details in the database.
- `update_project` **Method**:
  - This method updates a project's details in the database.
- `update_task` **Method**:
  - This method updates a task's details in the database.
- `delete_user` **Method**:
  - This method deletes a user from the database.
- `delete_project` **Method**:
  - This method deletes a project from the database.
- `delete_task` **Method**:
  - This method deletes a task from the database.

HTML Templates (index.html, login.html, admin.html, tasks.html, supervisor.html)

These files define the structure and layout of the web pages. They use HTML and Jinja2 templating to dynamically display data from the database.

- **`index.html`**:
    - This is the home page. It displays a list of projects and tasks assigned to the logged-in user.
- **`login.html`**:
    - This is the login page. It allows users to enter their username and password to log in.
- **`admin.html`**:
    - This is the admin page. It allows administrators to create projects, users, and update user details.
- **`tasks.html`**:
    - This page displays all tasks for a specific project. It allows users to create, edit, and delete tasks.
- **`supervisor.html`**:
    - This page is for supervisors. It displays all projects and tasks, allowing supervisors to manage them.

---

styles.css

This file contains the styling for the web pages. It defines the colors, fonts, layout, and overall look of the application.

Summary

This Task/Project Management system allows users to log in, create and manage projects, assign tasks, and track progress. Administrators have additional privileges, such as creating users and managing all projects. The system uses a database to store all data and provides a user-friendly interface for managing tasks and projects.