ProManage

Bryan Turns , *Sonia Purisai* , *Sawyer Mckenney* , *Ian Haneke* and *Owen Hushka*

1. Description

ProManage is a simple and efficient app designed to improve workplace management. This tool helps managers and employees communicate better. With ProManage, managers can easily keep tabs on their team's tasks without micromanaging. Employees, on the other hand, can update their progress, when delivering items or taking breaks, keeping everyone informed.

The app features two main sections: the administrative side for managers and the employee interface. Managers use their section to assign tasks, track progress, and message employees about any concerns. Employees use their interface to see daily or weekly tasks, update their status, and chat with managers if problems arise.

ProManage is for streamlining communications and promoting a positive work environment. It supports a culture where employees feel valued and managers operate more effectively. ProManage will help companies with communication and managing projects.

2. GitHub Tracker & Repository

Link to Github repository: https://github.com/BryanTurns/ProManage
Link to GitHub project Board: https://github.com/users/BryanTurns/projects/1/views/1

2.1 Member Details

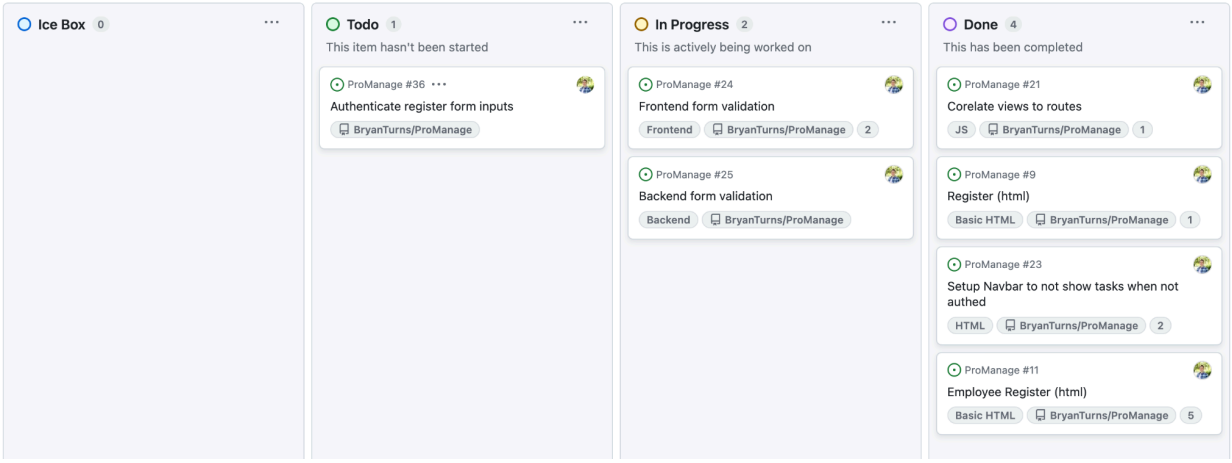
1. Bryan Turns https://github.com/BryanTurns
2. Sonia Purisai https://github.com/soniapurisai
3. Sawyer Mckenney https://github.com/sawyermckenney
4. Ian Haneke https://github.com/ihaneke786
5. Owen Huska https://github.com/OwenHushka1

3. How to use Promanage

Link to video VideoDemonstrationforPromanage.mp4

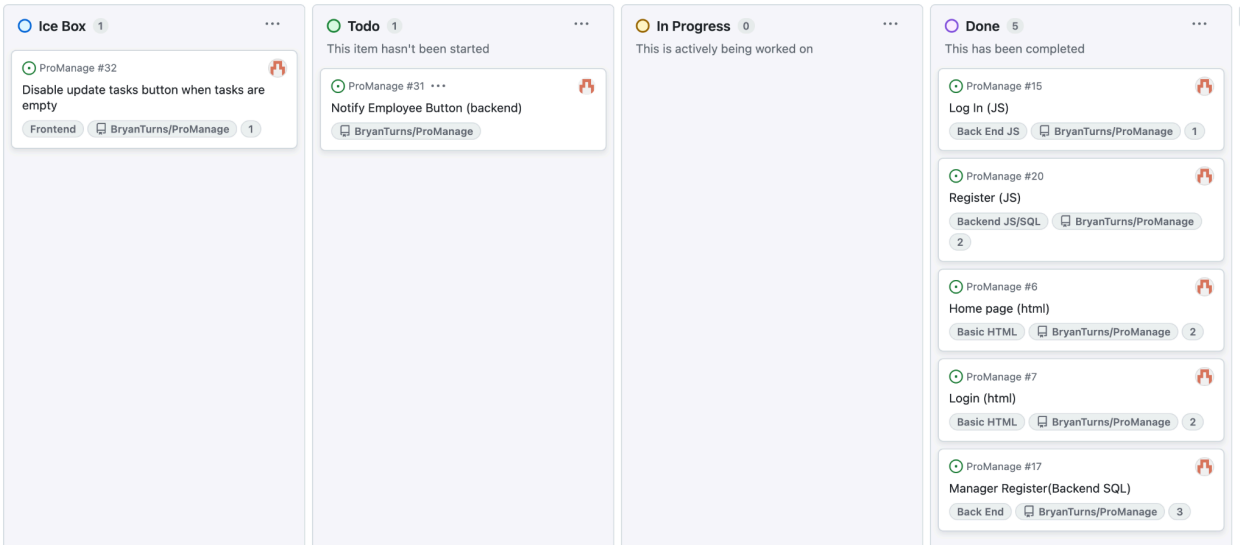
4. Contributions

4.1 Bryan Turns



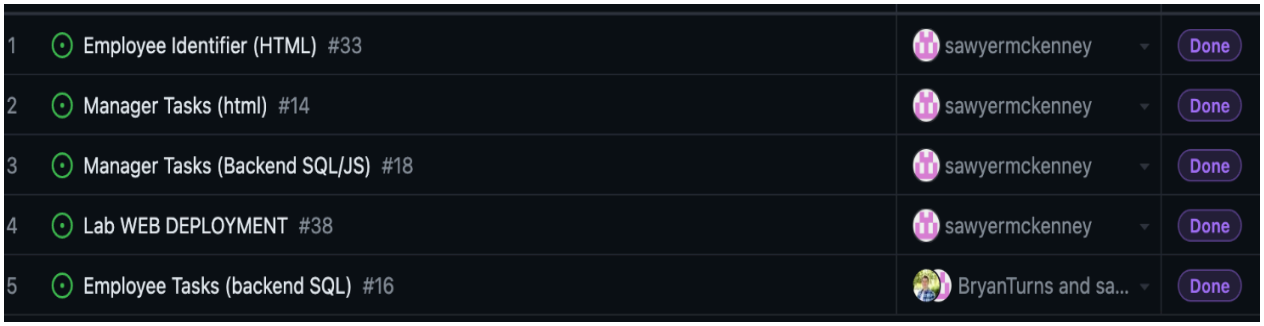
I Implemented the navbar using bootstrap. I created the frontend javascript that enabled users to switch between completed and incomplete tasks. I made create task and update task functional on the employee tasks page. I styled the manager tasks page with a combination of bootstrap, vanilla css, and tailwind.css. I created the frontend code. I also set up the development environment connecting docker, nodemon, express, tailwind, and the postgres db.

4.2 Sonia Purisai

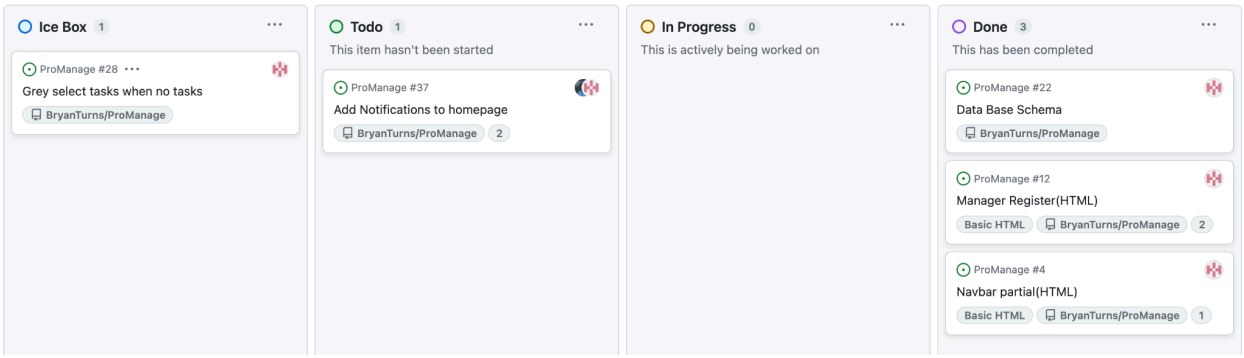


4.3 Sawyer Mckenny

I implemented the HTML part of the Manager Tasks page along with the modals that pop up upon each button press. After that I focused primarily on the manager side of things where I implemented the ability for managers to create, update and delete tasks. As far as the class labs go I focused on those throughout the week. I implemented the test cases that were required for lab 11 and also completed lab 13 where we deployed our application.

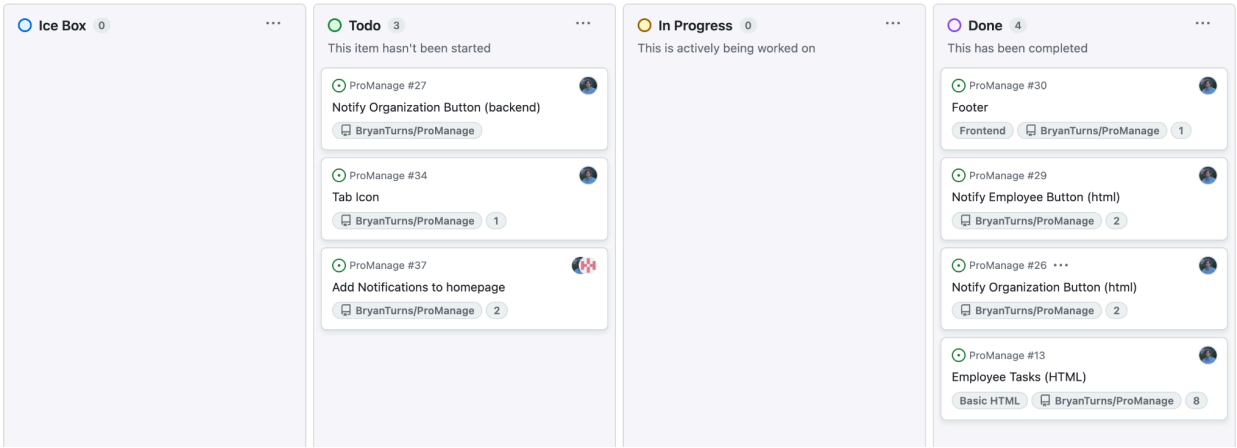


4.4 Ian Haneke



In the first week, I made the wireframes for the employee tasks page and the manager tasks page. Later on in the project, I built a lot of the homepage using html and bootstrap to include the carousel and formatting of the page, I built the Database schema for alerts, I worked on the layout of the navbar, and I modified the formatting of the manager and employee task pages.

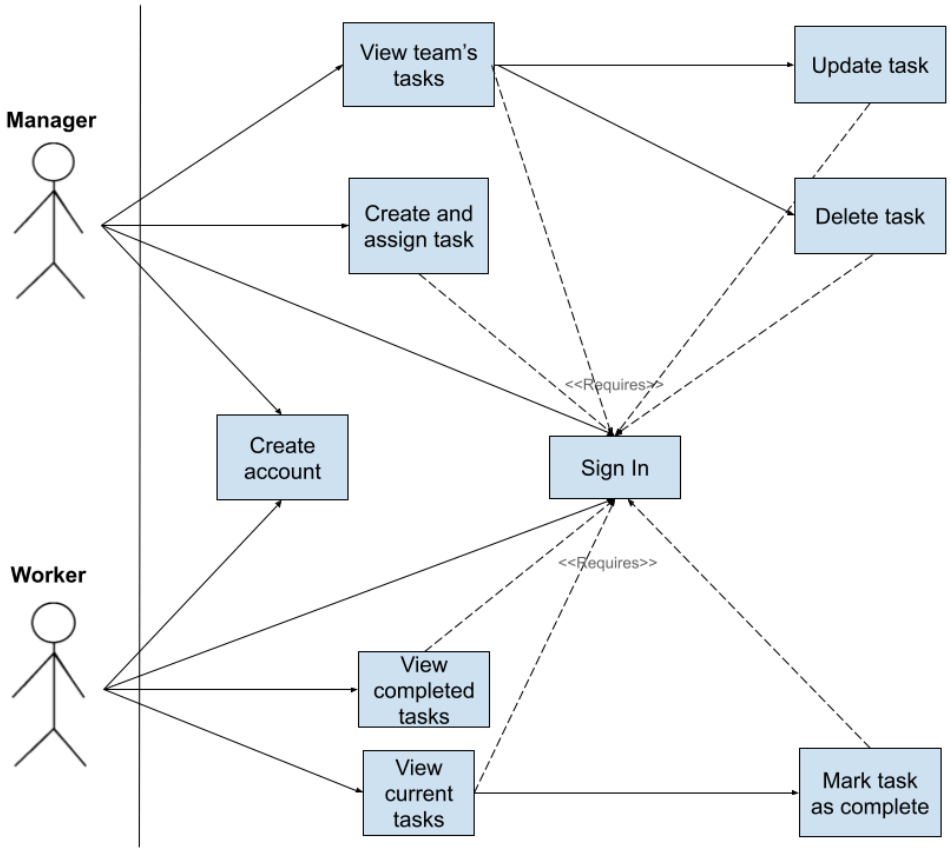
4.5 Owen Hushka



I made the wireframes for the sign in page and the register page. I then created the Employee Task page front end in HTML. I added the notify employee button as well as the notify organization button. This

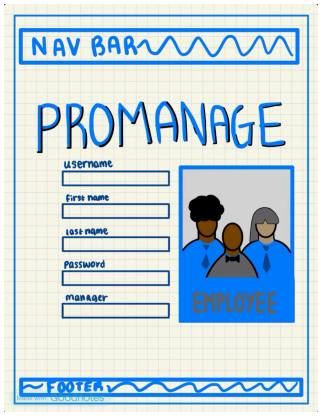
includes the pop up modal. I also created the footer as well as writing our test cases. Additionally, I did the design, trying to make the website pretty.

5. Case Diagram

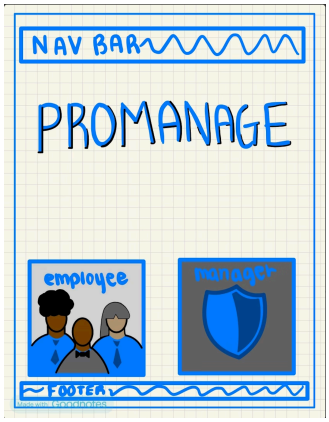


6. Wireframes

6.1 Register Page for Employees

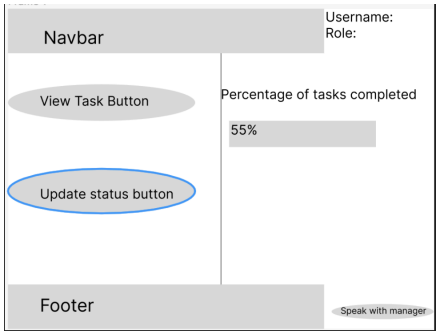


6.2 Page to Choose Manager or Employee

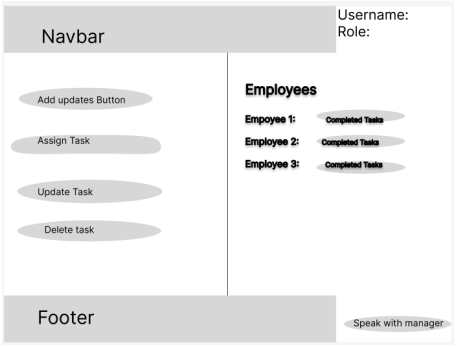


6.3 Register Page for Managers

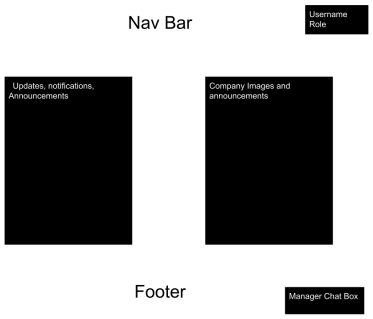
6.4 Employee Tasks page



6.5 Manager Tasks page

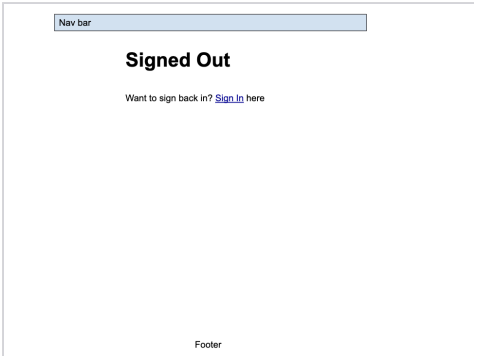


6.6 Homepage



6.7 Sign-in Page

6.8 Sign-out Page



7. Test Results

Feature 1: Assign an employee a task

Test Cases:

1. Task must have all required boxes filled by the manager.
2. All required boxes must be valid like date/time ect.
3. Tasks must show up in the employee task list.
4. When an employee updates the status this must reflect on the manager's end.

Test Data:

1. Task validated and added to employee DB.
2. Updated task but be updated in task DB.

Feature 2: Live Chat

Test Cases:

1. Employees must be able to access and send messages in the chat box.
2. Manager Must be able to receive messages from the employee.
3. Manager must be able to respond to the employee.

Test Data:

1. Message must be added to message DB by both employee and manager.
2. Both employee and manager must be able to see updated messages.

Feature 3: Completed Task

Test Cases:

1. Once a task is marked as complete by the employee the manager must be notified
2. The manager must be able to see a list of completed and uncompleted tasks by each employee.

Test Data:

1. Manager must have access to complete task DB and in Progress DB for the employee with the organization

- What are the users doing?

  The user was creating an employee and manager account. This is valid because that is a key dependency in the application. The user was then seeing if on the manger side they could assign tasks and this is a key functionality

- What is the user's reasoning for their actions?

  The reasoning behind this is to ensure the application is functional level.

- Is their behavior consistent with the use case?

  Yes.

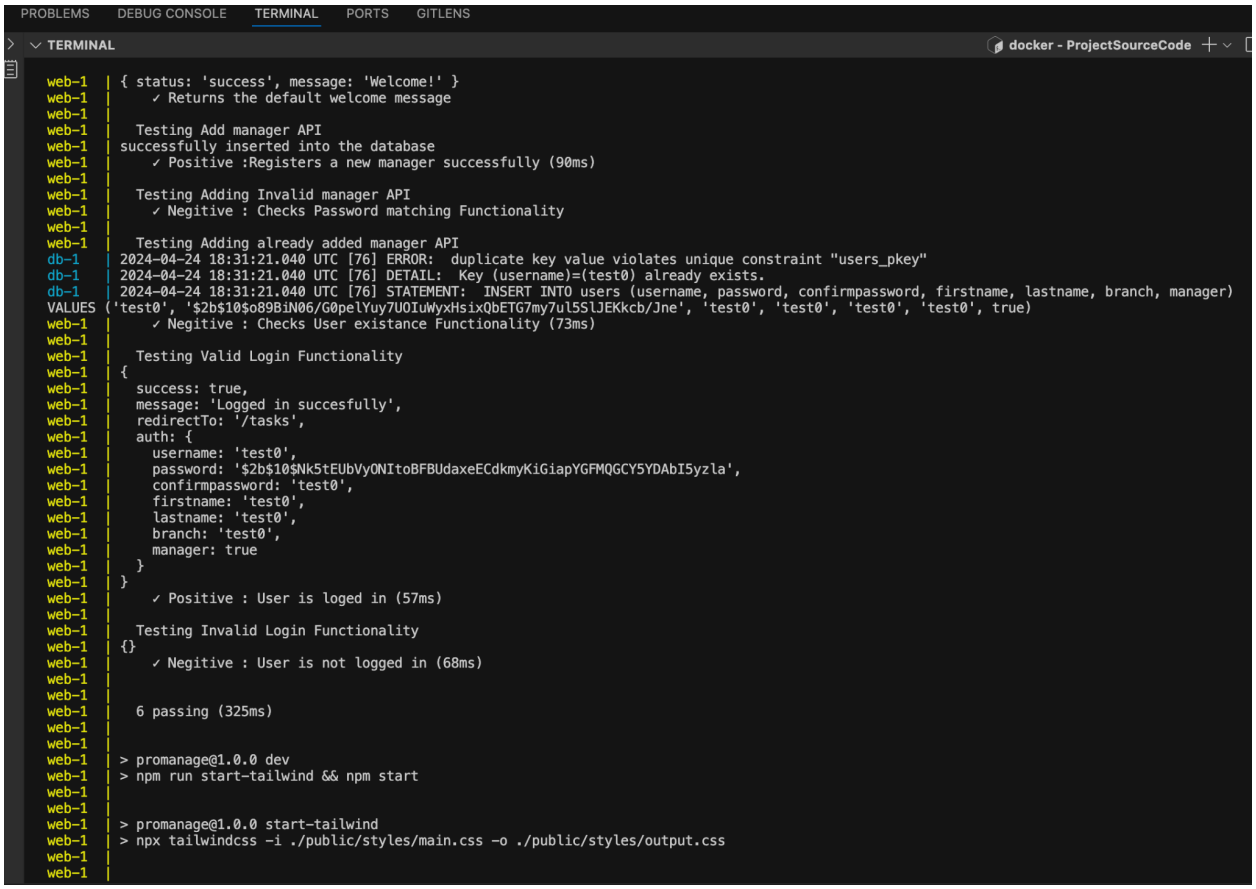
- If there is a deviation from the expected actions, what is the reason for that?

  There was only deviation when the user already existed in the SQL DB because that would not allow the test cases to pass because the username already exists.

- Did you use that to make changes to your application? If so, what changes did you make?

The changes were already implemented; we were essentially just testing the functionality to ensure it works as expected.

Below is a screenshot of the results:



8. App Deployment