Submission Worksheet

Submission Data

Course: IT202-450-M2025
Assignment: IT202 - Milestone 3
Student: Mark Y. (may23)
Status: Submitted | **Worksheet Progress:** 100%
Potential Grade: 10.00/10.00 (100.00%)
Received Grade: 0.00/10.00 (0.00%)
Started: 8/2/2025 10:49:40 PM
Updated: 8/3/2025 3:48:15 AM
Grading Link: https://learn.ethereallab.app/assignment/v3/IT202-450-M2025/it202-milestone-3/grading/may23
View Link: https://learn.ethereallab.app/assignment/v3/IT202-450-M2025/it202-milestone-3/view/may23

Instructions

1. Refer to Milestone3 of this doc:
   https://docs.google.com/document/d/1XE96a8DQ52Vp49XACBDTNCq0xYDt3kF29cO88EWVwfo/view
2. Ensure you read all instructions and objectives before starting.
3. Ensure you've gone through each lesson related to this Milestone
4. Switch to the Milestone3 branch
   1. `git checkout Milestone3` (ensure proper starting branch)
   2. `git pull origin Milestone3` (ensure history is up to date)
5. Fill out the below worksheet
   - Ensure there's a comment with your UCID, date, and brief summary of the snippet in each screenshot
   - Ensure proper styling is applied to each page
   - Ensure there are no visible technical errors; only user-friendly messages are allowed
6. Once finished, click "Submit and Export"
7. Locally add the generated PDF to a folder of your choosing inside your repository folder and move it to Github
   1. `git add .`
   2. `git commit -m "adding PDF"`
   3. `git push origin Milestone3`
   4. On Github merge the pull request from `Milestone3` to `dev`
   5. On Github create a pull request from `dev` to `prod` and immediately merge. (This will trigger the `prod` deploy to make the heroku prod links work)
8. Upload the same PDF to Canvas
9. Sync Local
10. `git checkout dev`
11. `git pull origin dev`

Section #1: (3 pts.) Api Data

Progress: 100%

Task #1 (1 pt.) - Concept of Data Association

Progress: 100%

Details:

- What's the concept of your data association to users? (examples: favorites, wish list, purchases, assignment, etc)
- Describe with a few sentences

Your Response:

The concept of my data association to users is a "Tracker" system, which allows users to select and monitor specific stocks from the available list.Those selections are saved in a separate Tracker table that links each user to their tracked stocks. This association enables users to easily revisit and manage only the stocks they care about.

Saved: 8/2/2025 10:50:45 PM

Task #2 (1 pt.) - Data Updates

Progress: 100%

Details:

- When an associated entity is updated (manually or API) how is the association affected?
    - Does the user see the old version of the data?
    - Does the user see the new version of the data?
    - Does the user need to have data re-associated or remapped?
- Explain why.

Your Response:

When an associated stock is updated, whether manually or via the API, the user sees the new version of the data immediately without needing to re-associate or remap anything. This is because the Tracker table stores only a foreign key to the stock's id, not a copy of the stock's data. Since the tracking relationship remains tied to the stock's unique ID, any updates to the stock's properties are reflected automatically wherever the stock is displayed — including in the user's tracker list. This ensures data consistency and reduces redundancy.

Saved: 8/2/2025 10:56:20 PM

Task #3 (1 pt.) - Handling Association

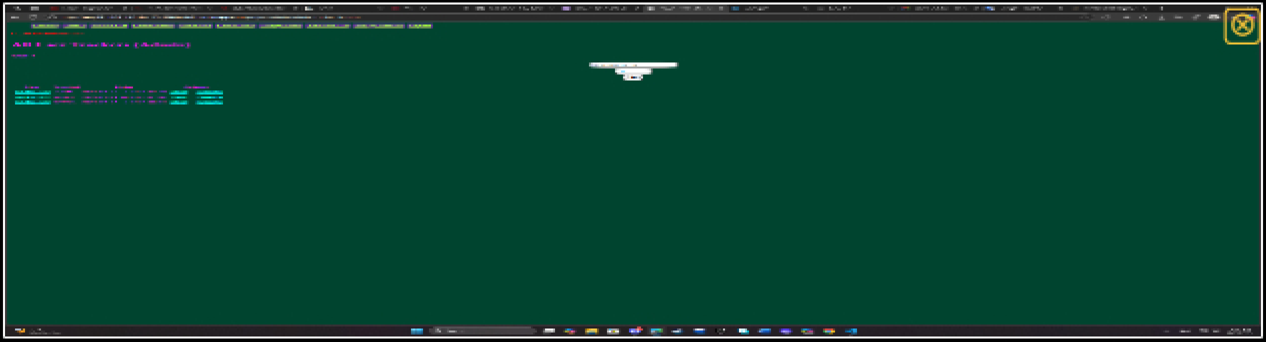
Progress: 100%

Part 1:

Progress: 100%

Details:

- Show an example page of where a user can get data associated with them
- Ensure heroku dev url is visible
- Caption if this is a user-facing page or admin page



admin page

Saved: 8/2/2025 11:14:46 PM

Part 2:

Progress: 100%

Details:

- Describe the process of associating data with the user.
- Can it be toggled, or is it applied once?

Your Response:

Data association between users and stocks is implemented through a "Track" feature. When a user clicks the "Track" button on a stock, a POST request is sent to create_tracker.php, which associates the current user's ID with the stock ID in the Tracker table. This action is not toggleable, once a stock is tracked, it stays tracked unless the user untracks it through a separate delete action. If the user attempts to track an already-tracked stock, a message is shown to prevent duplicates. This design ensures a controlled data association.

Saved: 8/2/2025 11:14:46 PM

Section #2: (6 pts.) Associations

Progress: 100%

Task #1 (1.50 pts.) - Logged-in User's Associated Entities

Progress: 100%

Details:

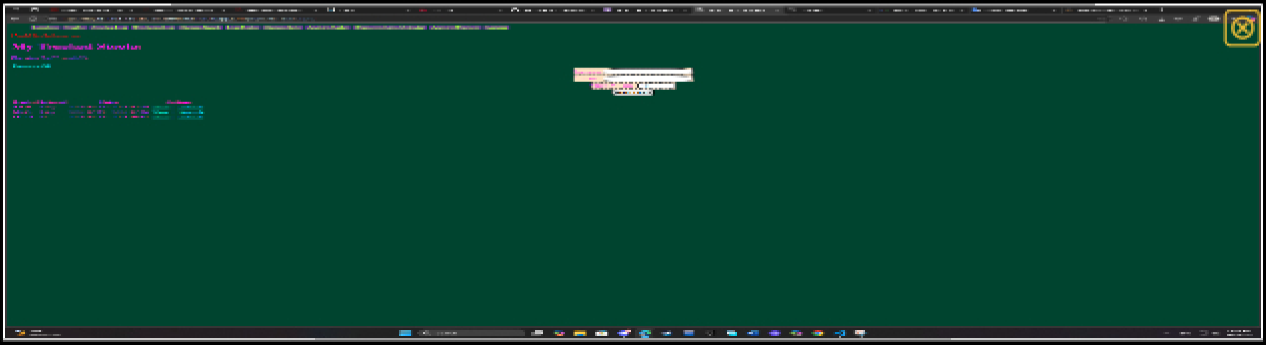
- Each line item should have a logical summary
- Each line item should have a link/button to a single view page of the entity
- Each line item should have a link/button to a delete action of the relationship (doesn't delete the entity or user, just the relationship)
- The page should have a link/button to remove all associations for the particular user
- The page should have a section for stats (number of results and total number possible based on the query filters)
- The page should have logical options for filtering/sorting
    - A limit should be applied between 1 and 100 and controlled by the user (server-side enforces rules)
    - A filter with no matching records should show "no results available" or equivalent

Part 1:

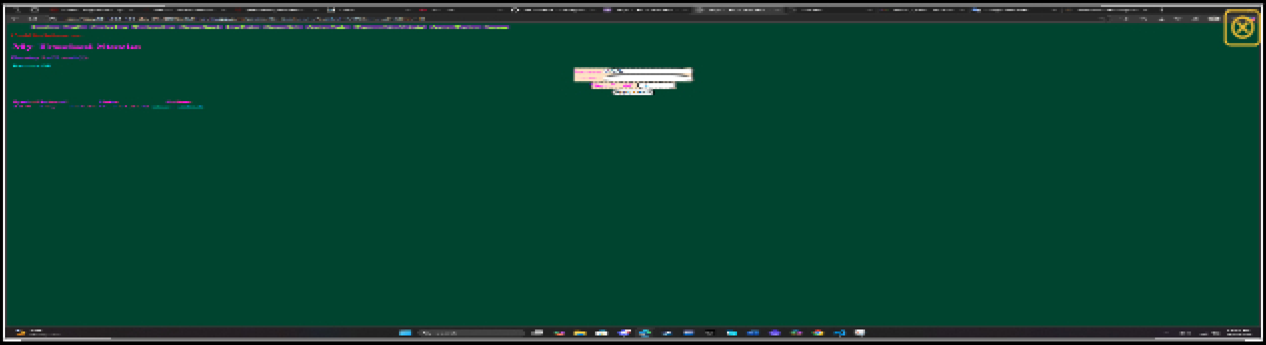
Progress: 100%

Details:

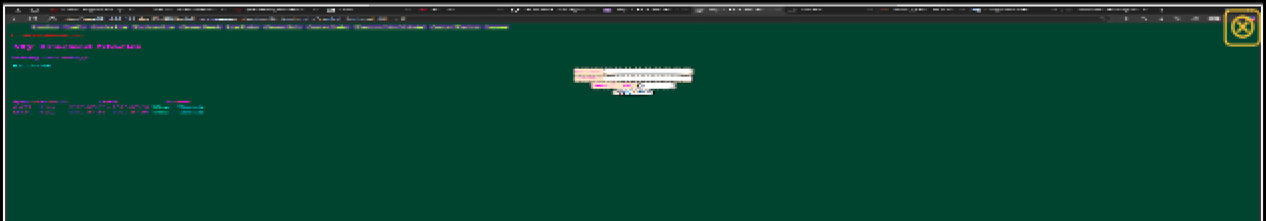
- Show a few examples of this page from heroku dev with various filters applied
- Ensure heroku dev url is visible
- Ensure each requirement is visible



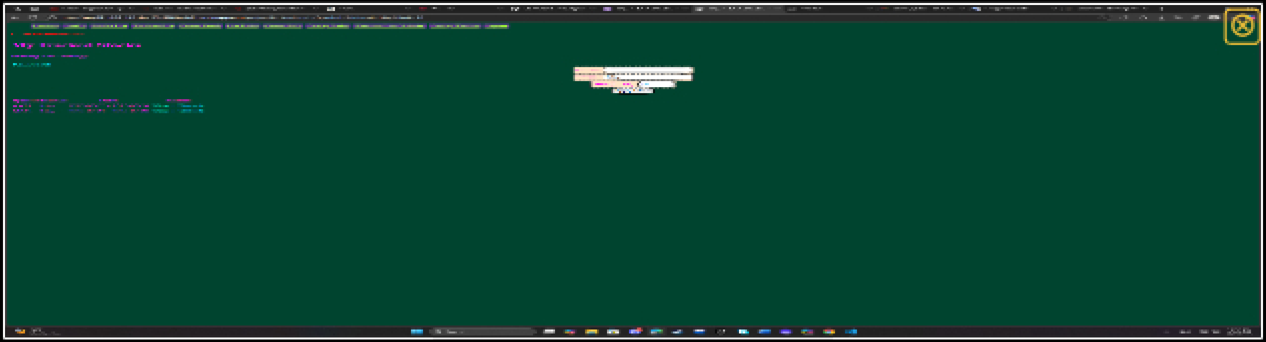
logical summary per line item& view button& untrack/delete the relationship button& remove all and everything else



symbol filter



limit filter



interval filter

Saved: 8/2/2025 11:38:54 PM

Part 2:

Progress: 100%

Details:

- Describe how you solved showing the particular association output
- Describe how you solved the various items required for this page (i.e., line item requirements, stats, filter/sort, etc)

Your Response:

To show the user's specific stock associations in the tracker, I joined the Tracker and Stocks tables using a SQL query filtered by the logged-in user's ID, ensuring only relevant data appears. Each line item displays a summary of the stock (symbol and date range) along with links to view or untrack it, fulfilling the line item requirements. For stats, I used PHP to count and display the total number of results dynamically. Filtering and sorting were implemented using query parameters with appropriate SQL conditions for symbol and interval, and the user can control the number of displayed results with a validated limit input between 1 and 100. If no matches are found, a clear message like "No results available" is shown.

Saved: 8/2/2025 11:38:54 PM

Task #2 (1.50 pts.) - All Users Association Page

Progress: 100%

Details:

- Each line item should have a logical summary
- Each line item should include the username this entity is associated with
    - Clicking the username should redirect to that user's public profile

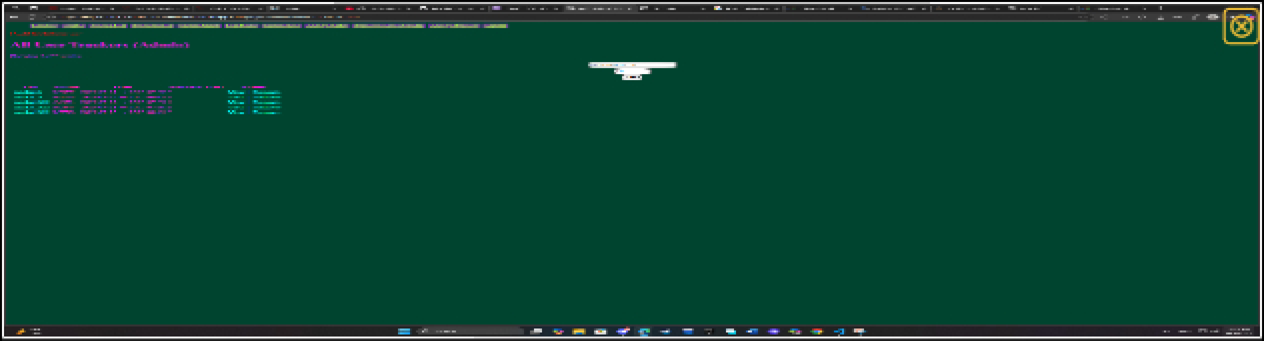
- Each line item should include a column that shows the total number of users the entity is associated with
- Each line item should have a link/button to a single view page of the entity
- Each line item should have a link/button to a delete action of the relationship (doesn't delete the entity or user, just the relationship)
- The page should have a section for stats (number of results and total number possible based on the query filters)
- The page should have logical options for filtering/sorting
    - A limit should be applied between 1 and 100 and controlled by the user (server-side enforces rules)
    - A filter with no matching records should show "no results available" or equivalent

Part 1:

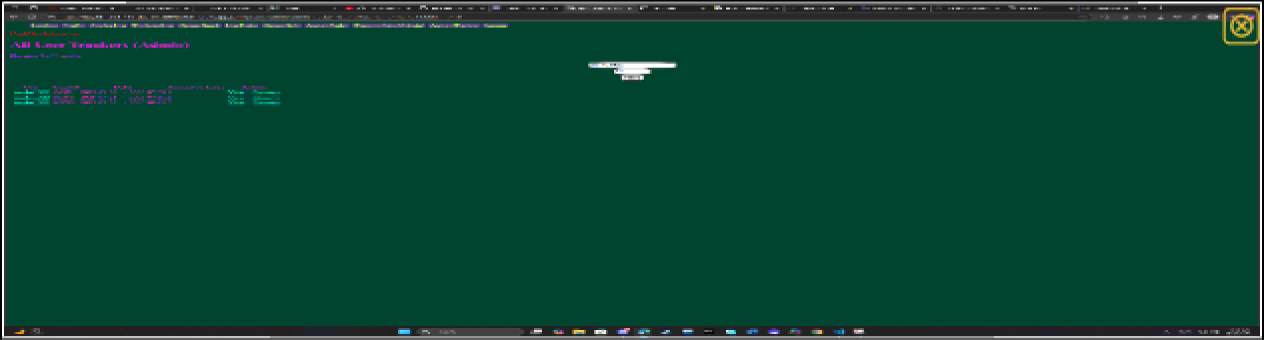
Progress: 100%

Details:

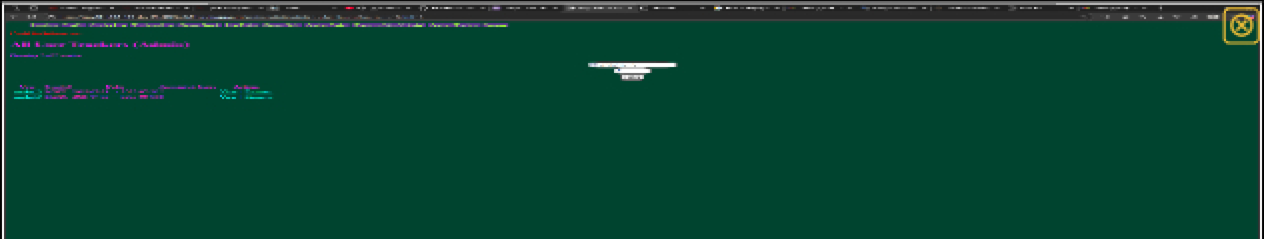
- Show a few examples of this page from heroku dev with various filters applied
- Ensure heroku dev url is visible
- Ensure each requirement is visible



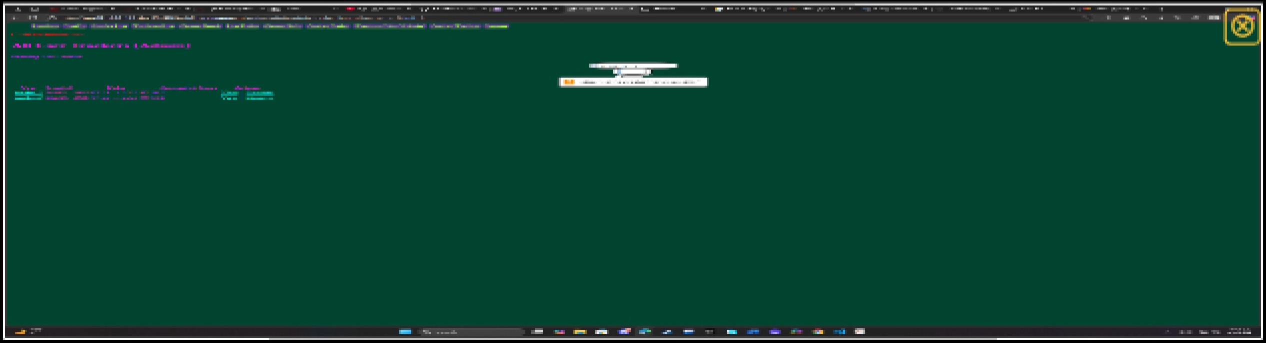
default settings



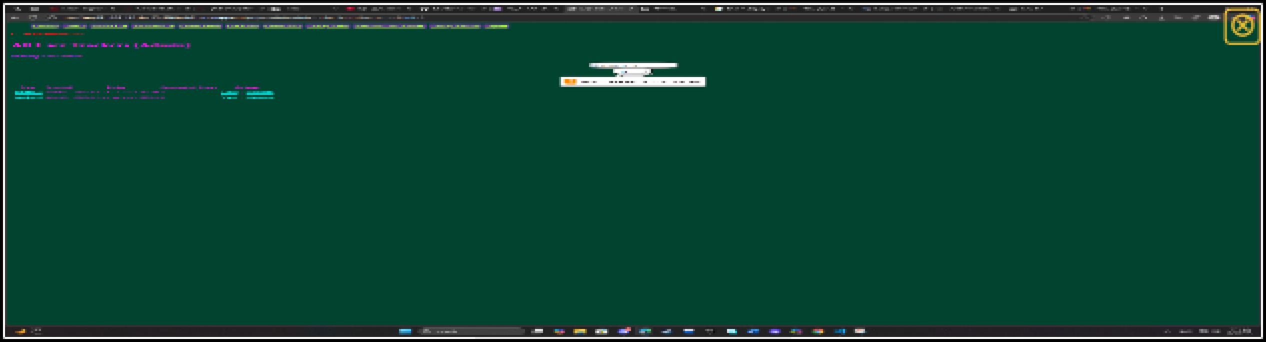
filter by username



filter by limit



limit over 1



limit below 100

Saved: 8/3/2025 2:02:57 AM

Part 2:

Progress: 100%

Details:

- Describe how you solved showing the particular association output
- Describe how you solved the various items required for this page (i.e., line item requirements, stats, filter/sort, etc)

Your Response:

To show the association output between users and stocks, I created a Tracker table to record which users track which stocks, then built a dynamic page that joins the Tracker, Users, and stocks tables to fetch and display this relationship. Each row displays a clear summary including the username, stock symbol, and date range, along with buttons for viewing the stock or removing the tracking association. I added a strict username filter to only show exact matches, a record limit input, and real-time statistics showing how many results are returned out of the total possible. If no results match the filters, a “No results available” message is shown. I also included the total number of users tracking each stock using a subquery, and ensured that usernames link to the user's profile for easy navigation.

Saved: 8/3/2025 2:02:57 AM

Task #3 (1.50 pts.) - Unassociated Page

Progress: 100%

Details:

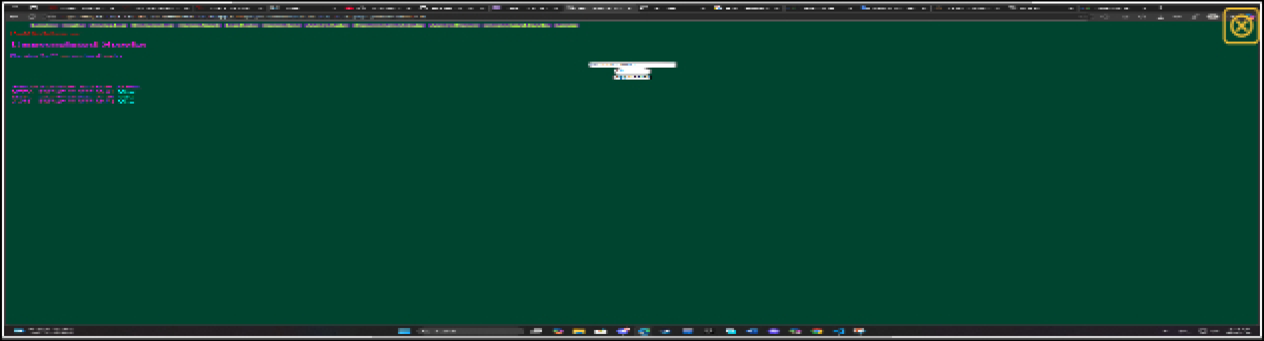
- Each line item should have a logical summary
- Each line item should have a link/button to a single view page of the entity
- The page should have a section for stats (number of results and total number possible based on the query filters)
- The page should have logical options for filtering/sorting
  - A limit should be applied between 1 and 100 and controlled by the user (server-side enforces rules)
  - A filter with no matching records should show "no results available" or equivalent

Part 1:

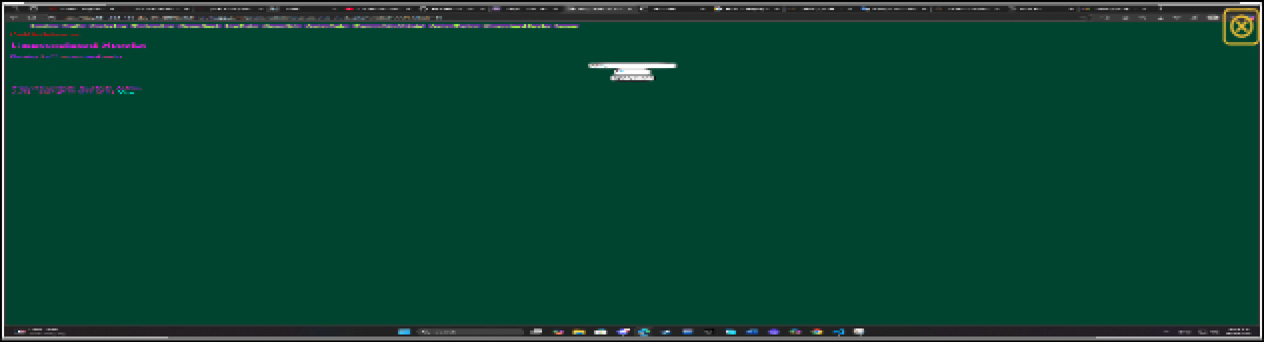
Progress: 100%

Details:

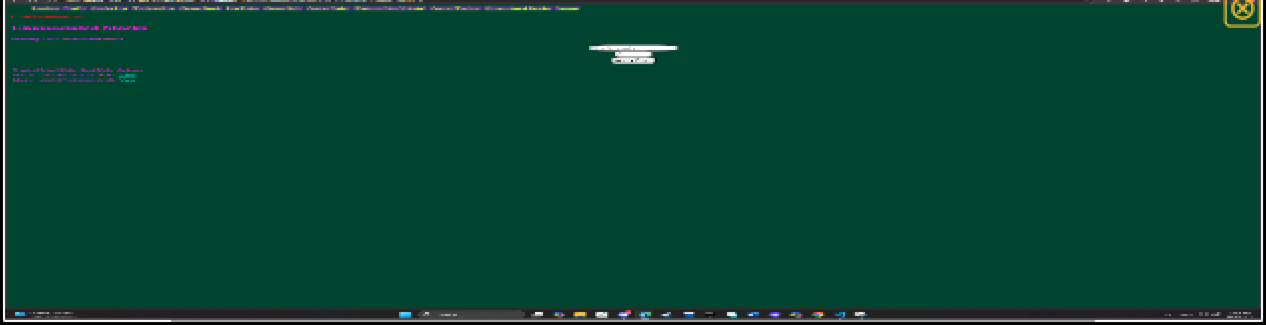
- Show a few examples of this page from heroku dev with various filters applied
- Ensure heroku dev url is visible
- Ensure each requirement is visible



no filters



filter by symbol



filter by limit

Saved: 8/3/2025 2:46:00 AM

Part 2:

Progress: 100%

Details:

- Describe how you solved showing the particular association output
- Describe how you solved the various items required for this page (i.e., line item requirements, stats, filter/sort, etc)

Your Response:

I joined the Tracker, Users, and Stocks tables to fetch relevant stock data associated with each user. Each line item includes a summary (symbol and date), the username it's linked to, the total number of users tracking that stock using a subquery, and actions like view and untrack. I added filters for strict username matching and a record limit, both enforced server-side. The stats section shows the result count, and a message appears if no results match the filter.

Saved: 8/3/2025 2:46:00 AM

Task #4 (1.50 pts.) - Admin Association Page (Like User Roles)

Progress: 100%

Details:

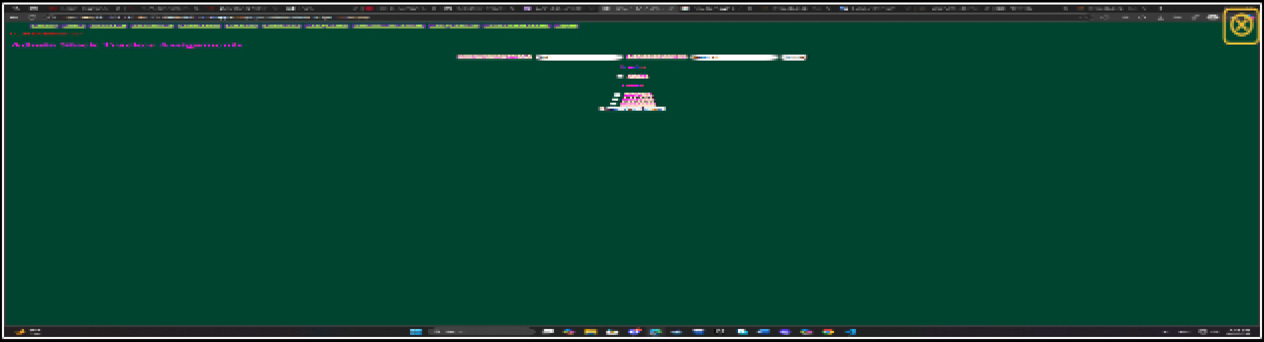
- The page should have a form with two fields
  - Partial match for username
  - Partial match for entity reference (name or something user-friendly)
- Submitting the form should give up to 25 matches of each
  - Likely best to show as two separate columns
- Each entity and user will have a checkbox next to them
- Submitting the checked associations should apply the association if it doesn't exist; otherwise it should remove the association
  - A filter with no matching records should show "no results available" or equivalent

Part 1:

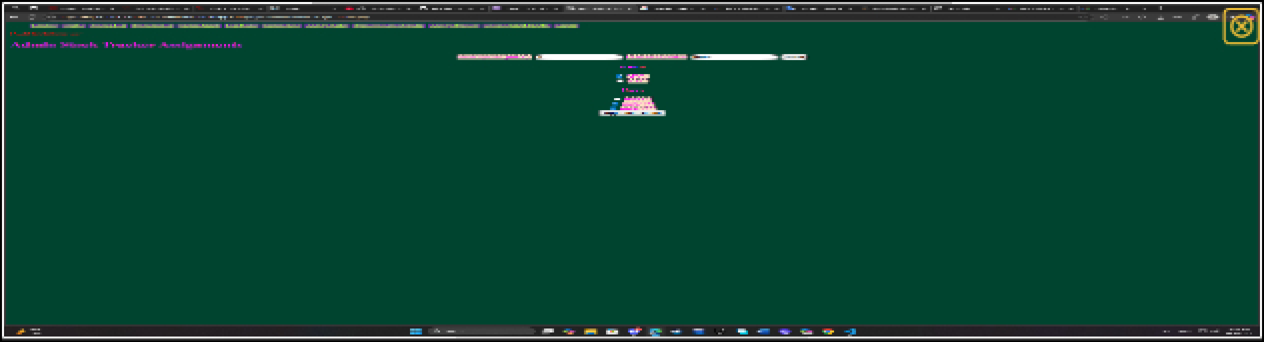
Progress: 100%

Details:

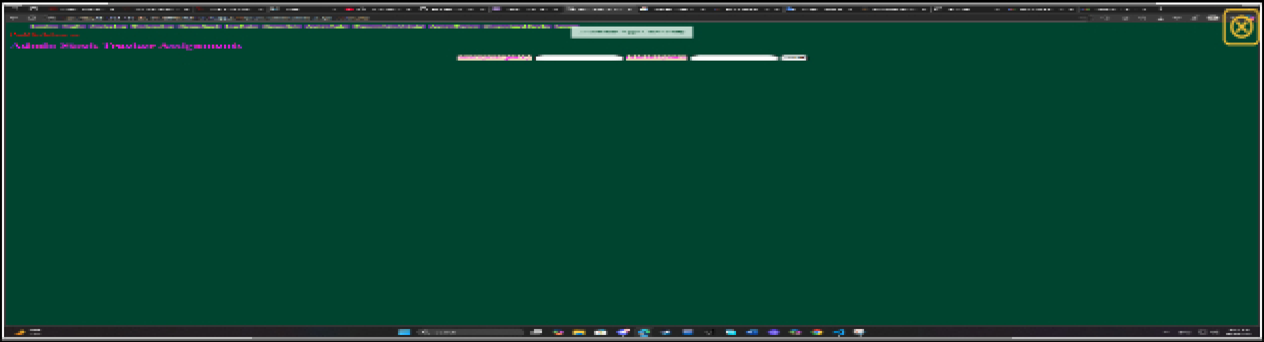
- Show a few examples of this page from heroku dev with various selections having been submitted
- Ensure heroku dev url is visible
- Ensure each requirement is visible



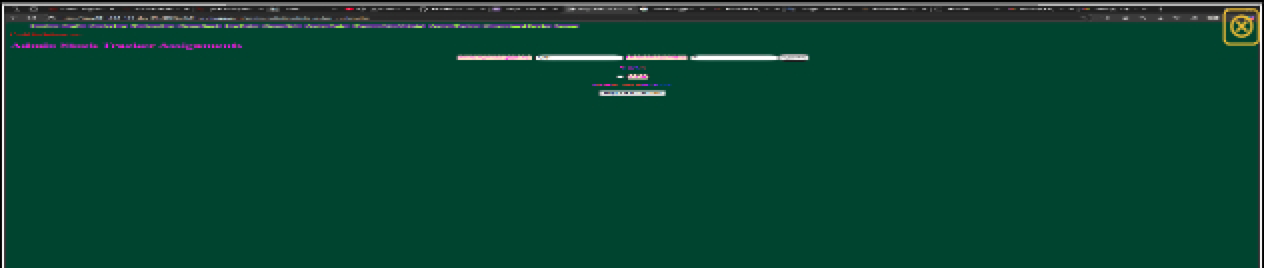
example 1



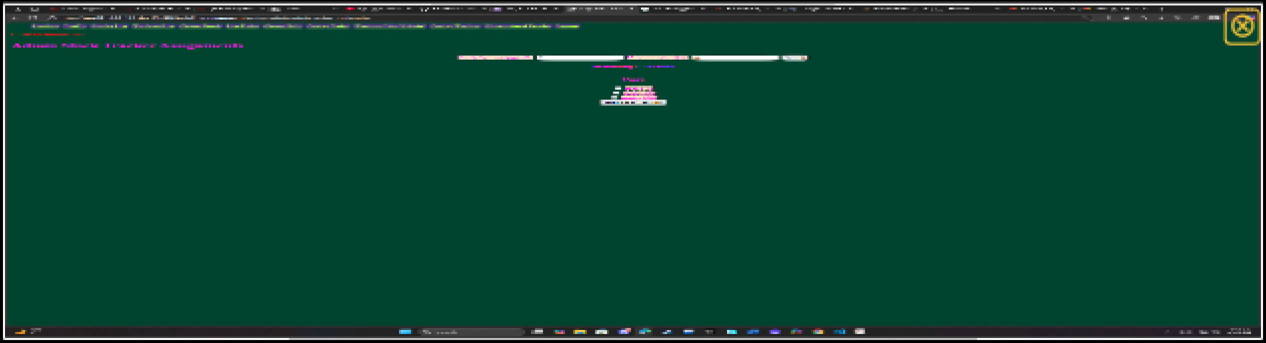
example 2



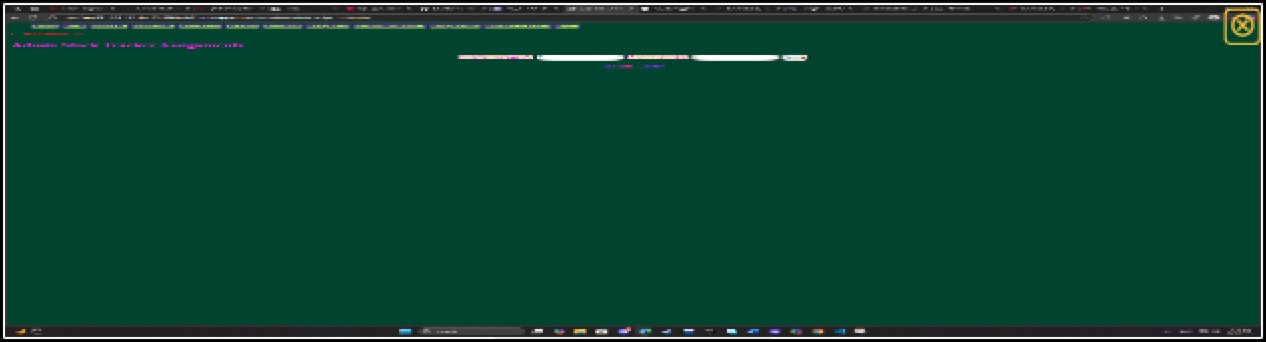
successful toggle message



no matching usernames



no matching symbols



none matching

Saved: 8/3/2025 3:41:31 AM

Part 2:

Progress: 100%

Details:

- Describe how your code solves the requirements; be clear

Your Response:

This code solves the Admin Association Page requirements by implementing a form with two partial match search fields—one for usernames and one for stock symbols. When the admin submits these inputs, the page queries the database and displays up to 25 results each in two separate sections (users and stocks), each with checkboxes. When the admin selects one or more users and stocks and clicks "Toggle Associations", the code checks if each selected (user_id, stock_id) pair exists in the Tracker table. If it does, it deletes the association (removes tracking); if it does not, it inserts the new association. This ensures a true toggle effect. The page also shows "No matching users found", "No matching stocks found", or "No results available" if nothing matches the filters, ensuring feedback for all user actions.

Saved: 8/3/2025 3:41:31 AM

Section #3: (1 pt.) Misc

Progress: 100%

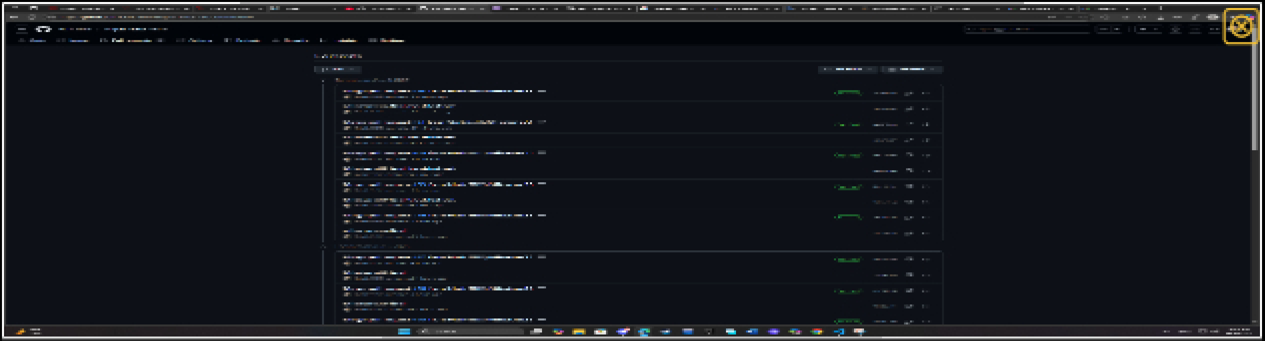
Task #1 (0.33 pts.) - Github Details

Progress: 100%

Part 1:

Progress: 100%

Details:
From the Commits tab of the Pull Request screenshot the commit history



commits history

Saved: 8/3/2025 3:43:31 AM

Part 2:

Progress: 100%

Details:
Include the link to the Pull Request for Milestone3 to dev (should end in `/pull/#`)

URL #1
https://github.com/Mark-5555/may23-IT202-450/pull/55

URL
https://github.com/Mark-5555/ma

Saved: 8/3/2025 3:43:31 AM

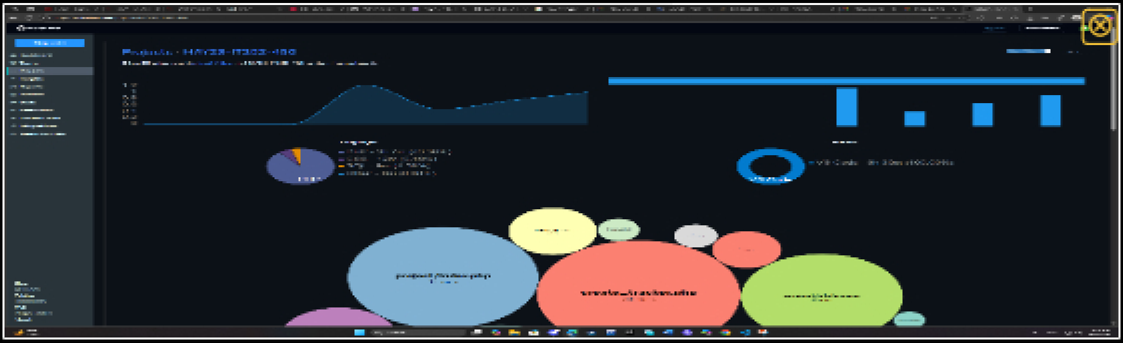
Task #2 (0.33 pts.) - WakaTime - Activity

Progress: 100%

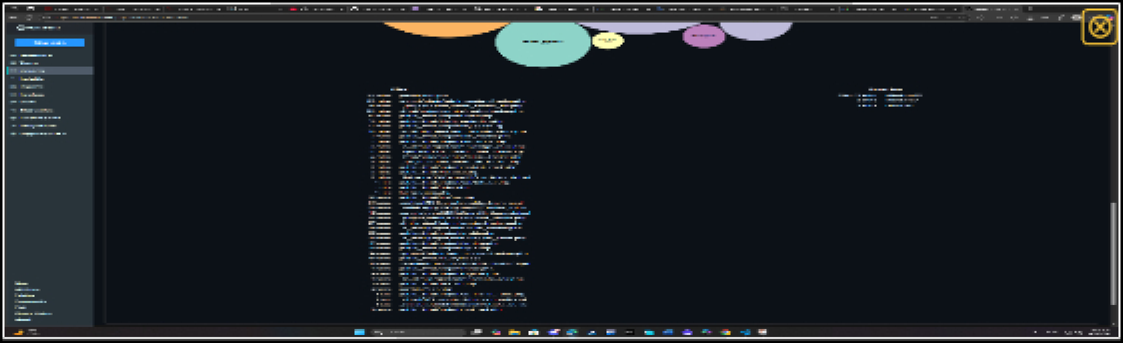
Details:

- Visit the WakaTime.com Dashboard
- Click `Projects` and find your repository
- Capture the overall time at the top that includes the repository name

- Capture the individual time at the bottom that includes the file time
- Note: The duration isn't relevant for the grade and the visual graphs aren't necessary



wakatime top



wakatime bottom

Saved: 8/3/2025 3:45:31 AM

Task #3 (0.33 pts.) - Reflection

Progress: 100%

Task #1 (0.33 pts.) - What did you learn?

Progress: 100%

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

I learned how to manage many-to-many relationships using an association table like Tracker to link users with specific entities such as stocks. I also gained experience in implementing features such as filtering, sorting, toggling associations, and handling edge cases like duplicate associations or deletion constraints. Additionally, I became more comfortable with writing efficient SQL queries, handling user input securely, and ensuring admin-level control over relationships in a web application.

Saved: 8/3/2025 3:46:23 AM

Task #2 (0.33 pts.) - What was the easiest part of the assignment?

Progress: 100%

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

The easiest part of the assignment was setting up the user interface for filtering and listing the tracked entities. Reusing existing page layouts and logic from previous milestones made this part relatively straightforward. Creating forms, displaying results in tables, and using checkboxes for selecting multiple users or entities felt familiar and manageable.

Saved: 8/3/2025 3:47:19 AM

Task #3 (0.33 pts.) - What was the hardest part of the assignment?

Progress: 100%

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

The hardest part was implementing the toggle functionality for associations. It required careful SQL logic to check if a relationship existed and then either insert or delete based on that condition. Debugging toggle errors and ensuring database constraints like UNIQUE(user_id, stock_id) didn't cause issues during insertion also took time. Ensuring the logic didn't silently fail and gave correct user feedback made this part challenging.

Saved: 8/3/2025 3:48:15 AM