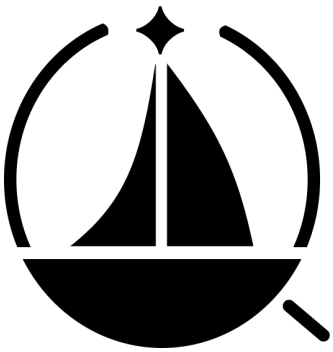

quayside.app
Final Report

Team Quaybit: Mya Schroder, Cam Dolan, Kai Iverson
CSCE A470 Capstone

Table of Content

Abstract

quayside.app is a web application designed to automate project management. It uses ChatGPT to generate tasks for a project and helps manage those tasks as the project progresses. In Spring 2024, as part of CSCE A470 Capstone, Team Quaybit successfully refactored quayside.app into the Django framework and added additional features including a REST API, updated graph visuals, a kanban board, and more. Positive feedback from quayside.app's users deemed this project a success.

1. Introduction

74% of projects fail to meet project objectives (PMI, 2024). Our goal is to reduce that number by developing quayside.app, a project management web application designed to get teams started on projects quicker (see Figure 1) and answer the question "What's next?". quayside.app uses ChatGPT to break a project into deliverables and help teams automate their project management.

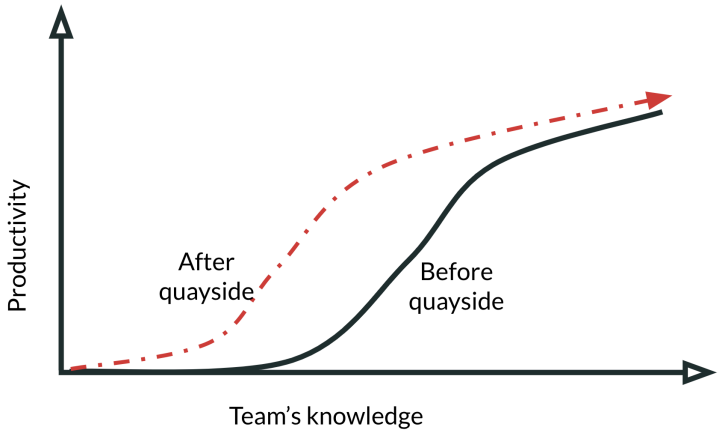

Figure 1: quayside.app's impact. (Williams, 2023)

Our community partner, Erik Williams has been a project manager for 7 years and is now the COO of Beadedstream. In the summer of 2023, he came up with the vision for quayside.app. Since then, quayside.app has been under development as an open source web application.

Over the course of the 2024 Spring semester in CSCE A470, Team Quaybit contributed to quayside.app by converting the web application framework from next.js to Django to speed up the development process and adding additional functionalities featured in the Requirements section. Erik Williams remained our community partner throughout the semester. Additionally, from time to time, developers outside of Team Quaybit worked on quayside.app (as this is an open source project). However, the features we discuss in this report are solely the ones developed by us during this semester.

2. Requirements

Refactoring this app meant we had to maintain already existing functionalities:

- Storage/view of users.
- Storage/view of projects.
- Task generation into tree with ChatGPT.
- Deletion of projects/tasks.
- Oath Sign in.
- Cloud hosting.
- Documentation with in-code comments/readme files and a GitHub wiki.

This meant we needed to translate all functions from the old next.js framework used in the previous version of quayside.app to the new Django framework. These requirements had to be fulfilled before moving on to meeting the requirements set for the new functionality to be implemented this semester.

To get an idea of what kind of additional requirements to create for this project we conducted an interview with Erik Williams. Using what we gathered from the interview we came up with the following functionality requirements:

- REST API that can be accessible separate from the web application.
- Additional graph manipulation functions such as rearranging tasks, adding tasks, and tree feedback.
- Kanban Board view for tasks.
    - Drag and drop task into different boards
- ChatGPT interacting regarding the project such as specific questions to keep the user on track with tasks.
- Sharing capabilities such as invite by email.
- Settings page.
- Getting-started screen that the user can view without logging in.

3. Design

quayside.app is a containerized Django application hosted on Google Cloud Run. It uses MongoDB to store data and interfaces with the ChatGPT3.5 API. The architecture of quayside.app is broken into two distinct parts: the API and the APP. The API listens for a request over the internet and responds with corresponding information. The APP also listens for a request over the internet but it responds with a web page to view. The APP uses API requests to get the information it uses to create the web page. A big advantage of separating our server into two parts is that the API is independent of the website. We can use the API to connect to quayside.app from any medium we want in the future.

3.1 API Design

Here are a few traits of a well designed API. The first trait is that it needs to give valuable information while simultaneously protecting other sensitive information. For example, user A should have access to their own projects, but should not have access to user B's projects. Another example is that a user that is not logged in should only be able to login, and nothing else.

The second trait is that the information an API gives should be as usable as possible. In other words, the API should perform as much logic and computation on the information as it can before sending the info off. There are three main reasons for this. Number 1 is that this makes the API very easy to use. Number 2 is that this leaves the computation to the server and not the user. Number 3 is that the API can be used in a consistent way across many mediums.

quayside.app's API has both of these traits. We have systems in place to ensure that the API is protected and we have also designed the API to give information in its most usable form. And now, here is a breakdown of quayside.app's API's features.

There is a protected route to get, create, edit, and delete a list of tasks or a single task, filtered by anything we want. Many of the other routes also deal with tasks. The flexibility and control we gained through the tasks route made creating the other routes simpler.

A protected route called projects gets, creates, edits, and deletes projects. In quayside.app, a project is a collection of tasks, along with other information such as the owners. We are also able to get multiple projects at a time, filtered by anything we want using this route. This is used, for example, to get the name of all of the projects that a user owns.

Along with the project route, we have a route to generate a project. This route is behind the most important feature of quayside.app, breaking a project down into tasks. We discuss this route in detail in **section 3.5**.

The kanban route is a protected route that makes it easy to view the tasks of a project, grouped by their completion status. This route is discussed in the **sections 3.3** and **section 3.4**.

Finally, a protected route exists to get, create, and update a user. This route is important for our session management and for project ownership. It is used to seamlessly create an account when a user logs in for the first time.

3.2 User Interface Design

All of the web pages of quayside.app, besides the welcome page, contain the same sidebar and header. The header has buttons to navigate to the home page, logout modal, and settings page. It also displays the user's name, to visualize that they are logged in. The sidebar contains a dropdown tab that displays all of the projects that a user owns. At the top of the sidebar is a big button labeled "+ Project" that is used to open up the create project modal. In the future, as we add more features, there will be access to more from the sidebar. Through the sidebar and header, a user has a button to get to most of quayside.app from most places within quayside.app .

As stated earlier, the welcome page is an exception. It has no sidebar since the user is not logged in yet. The welcome page exclaims "quayside.app Ignite Collaborative Productivity" before showing off some of the features of quayside.app . The header prompts the user to login.

When a user navigates to one of their projects, a new header appears: the project header. It contains the name of the project, a button to select the project visualization (more in **section 3.3**), a button to create a new standalone task, a button to open the share project modal, and a delete project button. When the delete button is pressed, the user is asked to confirm if they really want to delete the project.

Figure 2 shows the task modal. When a user goes to create, edit or delete a task, they are brought to the task modal. There is a field to add/edit the task name, completion status, data range, time estimation and description. At the bottom left, a button to delete the task. At the bottom right, a button to save the changes made.

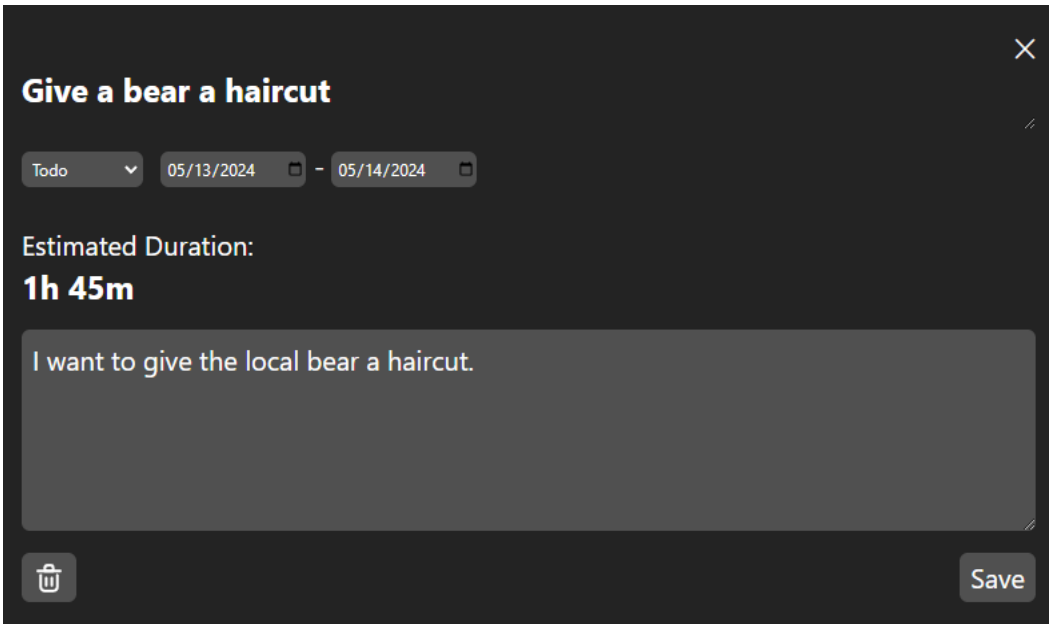

Figure 2: Task Modal.

A settings page is currently in the works and will be completed in the near future. It will contain the following tabs: Account, Team, Theme, Projects, Email, Notifications. The theme settings are exciting because they allow the user to customize the look and feel of quayside.app . Users can select their favorite background image, completion status colors, website color and font color. quayside.app will keep the users theme when they logout and log back in.

3.3 Project Visualization

There are two ways to visualize a project in quayside.app: the project tree (Figure 3) and the kanban board (Figure 4).

We designed the project tree visualization because it emphasizes larger tasks being broken down into smaller tasks. At the root of the tree is the project itself. The first layer down represents the tasks needed to complete the project. The next layer down represents subtasks needed to complete the task above. The task is rendered the color corresponding to its completion status to help the user visualize what they have done and what they need to do next. “+” buttons are on the right of each task that allow a user to add new tasks and customize their project to their liking.

The tree is rendered using the D3 svg javascript library which generates nodes in a tree shape given parent-child data relations. At each node, we add an svg colored box with each task’s text and a “+” symbol. The box and “+” symbol is associated with the corresponding task modal url so when the user clicks on those portions of the graph, it routes the user to the modal to edit or create a new task.

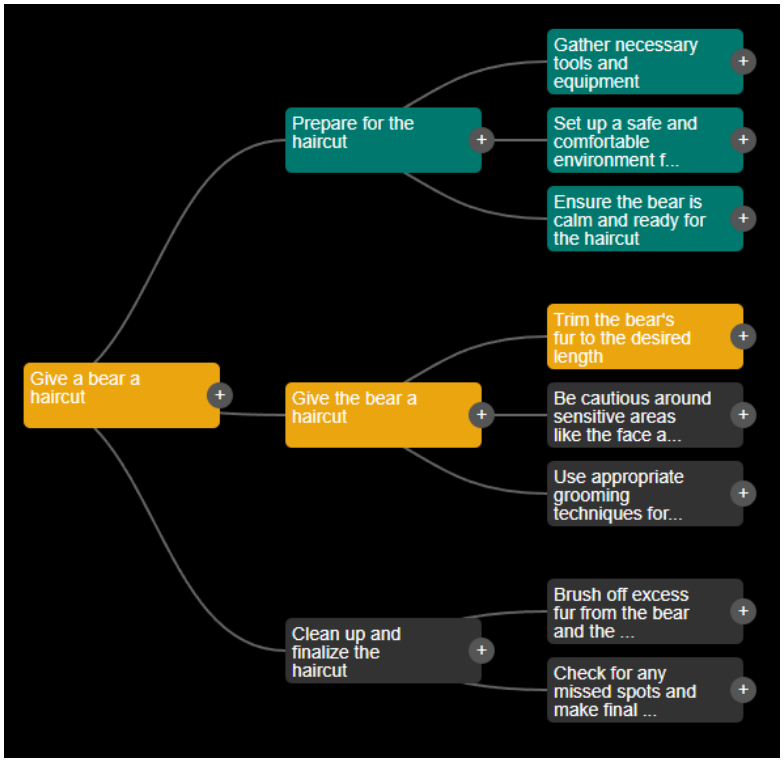

Figure 3: Task tree.

The second project visualization we created is called a kanban board (Figure 4). A kanban board shows each task, grouped by their completion status. A user can drag a task from one status to another and the task will update to have its new status. quayside.app also maintains each task's priority within the status. This means that when a user updates the kanban board, then leaves and comes back, each task will have the correct status and be sorted in the same order within the status.

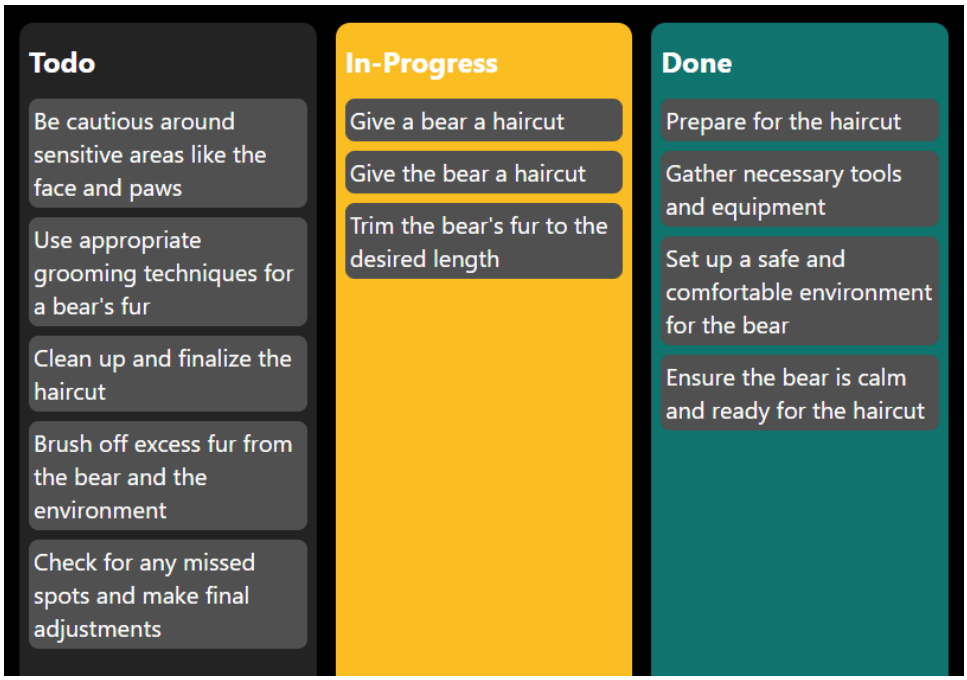

Figure 4: Kanban Board.

3.4 Algorithms

Correctly updating and maintaining the kanban board proved to be a difficult challenge with a clever solution. It all stems from an innocent question: how do we maintain the order of tasks within a status? We start with a field in the tasks table, called priority, to store order within status. Now, how do we initialize this value for each task in a project? The solution is to group each task by status and then assign priority, starting from 0, counting up by 1. What if we want to change a task's priority, how does that affect all of the other tasks priorities? What happens to the priority and surrounding priorities when we want to move a task from one status to another? What happens when we create a new task? What happens when we delete a task? What happens when some priorities are null, some are integers, and the ones that are integers are arbitrarily spaced apart?

The solution is contained in the updateKanban endpoint, and has three steps:

Step1: Normalize the priority of the tasks (Figure 5).
The goal of this step is to ensure every task in a status has a valid priority. Within a status, there should be a task with 0 priority, 1 priority, 2 priority, and so on. We also want to maintain order during this process. The trick is to sort tasks by priority, from smallest to largest, while letting tasks with null priority goto the end of the list. Next we reassign all of the priorities, in the sorted order, starting from 0 and counting up by 1.

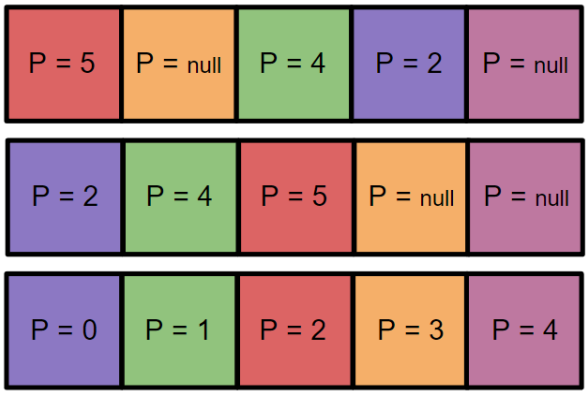

Figure 5: Normalize Priority.

Step 2: Remove task from status list.
To correctly remove a task from the status list, we need to adjust the priority of other tasks within the status list. For example, consider 5 tasks, sorted by priority, with the TODO status (0, 1, 2, 3, 4). If we remove the task with a priority of 2, we are left with a gap in priority (0, 1, 3, 4). To resolve this, we decrement the priority of all tasks in the list with a priority greater than 2 (0, 1, 2, 3).

Step 3: Add the task to the new status list.
To make space for the task within this list, we need to do the opposite of what we did in step 2. Here is the example: consider 6 sorted tasks in the DONE status (0, 1, 2, 3, 4, 5). We want the inserted task to have a priority of 3. What we do is increment the priority of all tasks in the list with a priority of 3 or greater (0, 1, 2, 4, 5, 6). We now have enough space to insert a task with a priority of 3.

3.5 Breaking a Project into Tasks.

The most important feature of quayside.app: Describe your project to quayside.app, quayside.app breaks your project into tasks and tells you need to do next. Here is how we pull this off.

It starts with the user's input. Our project prompt user interface (Figure 6) is very simple, generalizable, and powerful. The user enters the name of the project, and a description as simple or complex as they want, and quayside.app does the rest.

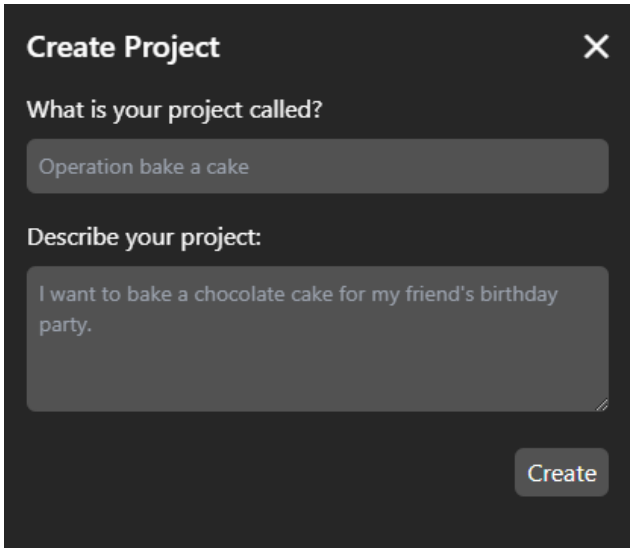

Figure 6: Project Prompt.

This information is passed to the generateProject route. From here we connect to openAIs GPT API. We define the role of the model to be a project management assistant. We say that it needs to turn the project name and description into tasks and subtasks. We instruct it to strictly respond in the following format:

1 Task1
1.1 Subtask 1
1.2 Subtask 2
1.3 Subtask 3
2 Task 2
2.1 Subtask 1
2.2 Subtask 2
and so on…

We insert the project name and description into the prompt and let the large language model do its thing. We have found that it reliably sticks to the format we tell it to respond in.

The next step is to parse the LLMs response. We split the response by '\n' and use a regular expression to determine if each line is a task or subtask, the parent task of each task, and the name of each task. From here, we can simply use our existing project and task routes to create a project.

4. Software development process

4.1 Development Lifecycle and Work Schedules

The development process occurred in two week iterations (shown in Figure 7). We usually met with the community partner on a weekly basis to discuss progress or any problems. We also met on our own time when deemed necessary. We usually worked individually with frequent communication and collaboration through discord and GitHub. As mentioned earlier we followed a two week iteration schedule similar to Agile. This means at the start of each iteration the we would divide out the next scheduled tasks with the expectation that tasks would be finished at the end of iteration. However some tasks could not be completed in their iteration either due to either challenges impeding tasks completion or team members being occupied with other responsibilities.

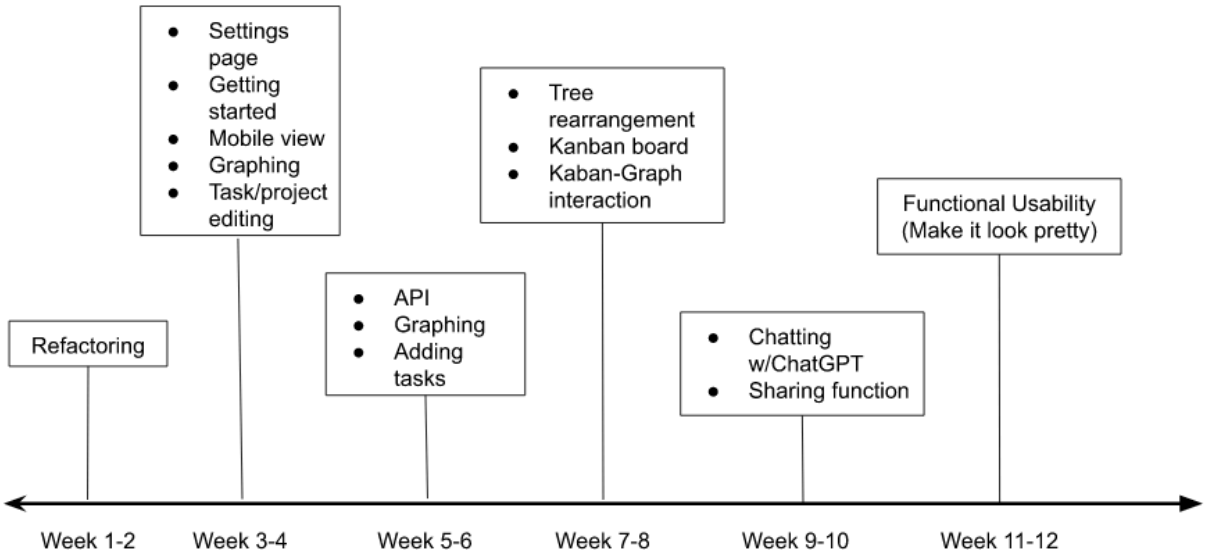

Figure 7: Project planning.

4.2 Challenges

Due to using a noSQL database with Django, which prefers SQL, many of Django's built-in functionalities that we could have used had to be manually implemented. This is where a large chunk of the development process was spent for members tasked with those implementations. Another challenge was that two members of the team weren't familiar with JavaScript and HTML prior to this project, so they had to spend more time on their frontend task.

4.3 Debugging and Evaluations

Debugging was done mostly through user testing and some implemented test cases. User testing came mostly from heuristic evaluation done by our peers, friends, colleagues or ourselves. A table of the bugs/critiques from the evaluation and the status of the bug can be seen below in Table 1.

Heuristic Issue (1-10)	**Severity (1-4)**	**Description of Problem**	**Status**
7	2	No scroll bar -- if mouse wheel doesn't work	Considering
		no scrolling limit	Fixed
8	1	When I click outside of the sign in box, the box does not disappear	Considering
5	3	User can enter more than 4 numbers for year date project	Considering
8	2	When entering and exiting the create task menu, the app lags a bit and removes the project tree for a moment	Fixed
2	1	Can't login without a github or google account	Considering
5	1	Title box resize for new task	Fixed
H1	1	For hi-res displays, tasks disappear off screen too soon (also for small project trees)	Fixed
9	2	Exception if login is canceled (MultiValueDictKeyError at /callback/)	In-Progress
H2,H5	3	invalid date range is allowed (start date after end date); Feb 31	Fixed
H3	1	Dragging too far can make it difficult to find the list again	Considering
H4	3-4	New tasks either disappear or have a small canvas	Fixed
H2	1	Not possible to modify layout lines/task box placement	Considering
H1	1	When you're using different date format like dd.mm.yyyy, there is a comma in front of the date	Considering
H1	2	Sidebar is not expandable so you can't see your project's full names	Considering
1	1	Add task + not centered	Fixed
2	1	Grabbing the name bar will let you drag it down for the entire page	Intentional
10	2	A "help" button or more of a description/instructions of how quayside works	Considering
H3	2	GitHub sign in wants full read & write access to personal user data, Google sign in only reads	Investigating
H5	3	(More of a suggestion) Since dealing with user accounts, something like this might be helpful for finding security vulnerabilities if not already considered	Appreciated
h5	3	sign-in w/ GitHub = error on FF	Considering

H7	1	Keyboard shortcut -- Ctrl + Enter = Save for entering a new task?	Considering

Table 1: Heuristic Review.

4.4 Testing Results

Toward the end of the semester we spent a considerable amount of time testing and running through the general flow of this project's final version of the app to ensure everything worked properly. To further test the results one member of the team started using the app regularly.

5. Results

5.1 Final Application

Our project is currently live at quayside.app (also shown in Figure 8). See **Appendix A** for a user manual for our application. We also have some blog-style documentation featured at quayside-app.github.io/Wiki/ that talks about more architectural and up-front decisions we made. For code specific documentation, our github repo contains a lot of valuable information. The readme contains information about how to set up and run the web app locally, and in the program files, all the functions contain a function block comment that explains each function.

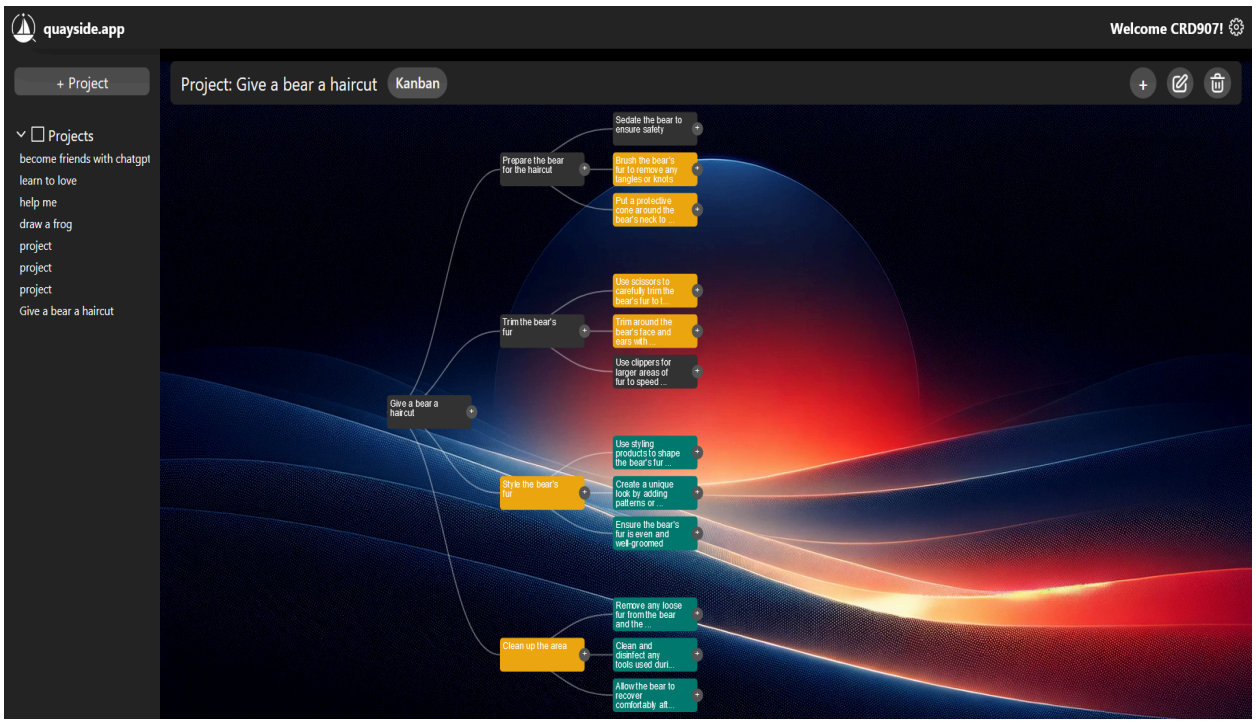

Figure 8: Application Main Page.

Overall, we fulfilled almost all the requirements we specified at the beginning of the semester. The only exception to this was the “Chatting with ChatGPT” feature because towards the end of the semester, before we started the chatting sprint, Erik Williams decided it would not add enough value to the application at this time in proportion to the amount of time it would require. It would also raise our ChatGPT API cost. Additionally, we implemented the front end of our settings page but did not the backend as we were prioritizing more impactful features.

During our final Capstone Poster Presentation, we received lots of positive feedback for quayside.app, with some people we interacted with telling us they would definitely use the web application to manage their project. Additionally, we feel that we have developed quayside.app enough so that we can start managing quayside.app with quayside.app (it is a bit meta). We mark this as quite a big milestone.

5.2 Contribution Breakdown

Contribution was split up by feature as follows:

- Cloud hosting. - Cam 100%
- Oath Sign in. - Cam 100%
- Getting-started screen that the user can view without logging in. - Cam 70%, Mya 30%
- REST API that can be accessible separate from the web application. - Mya 70%, Kai 30%
- API Authentication and Authorization with API Key. - Mya 100%
- Task generation with ChatGPT. - Mya 70%, Kai 30%
- Project Data Editing (Name, Date). - Mya 100%
- Task Editing (Name, Description, Date, Status). - Mya 100%
- Project Graph manipulation. - Mya 100%
- Kanban Board Frontend. - Cam 100%
- Kanban Board Backend. - Kai 100%
- Sharing via email. - Mya 100%
- Settings page Frontend. - Kai 100%
- Documentation with in-code comments/readme files and a GitHub wiki. - Mya 33%, Cam 33%, Kai 33%

Specific file contributions can be viewed in our Github repository, using the “git blame” feature. Additionally, Figure 9 shows Github contributions over the course of the semester (Mya is “schromya”, Cam is “CRD907”, and Kai is “kaiverson”).

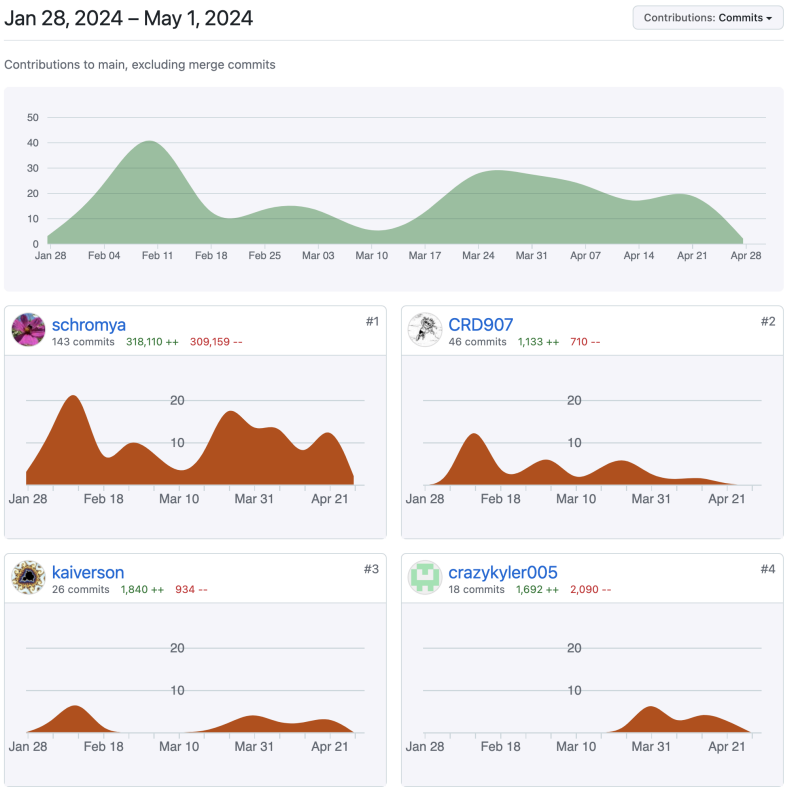

Figure 8: Github Contributions.

5.3 Next Steps

There are many new features we want to implement to add on to quayside.app. These include accurate task duration estimation, integration with google/github to track tasks, and email/text notifications to keep users on track with their tasks.

6. Conclusions and Lessons Learned

The idea for quayside.app was created in the summer of 2023. In the fall of 2023, a group of computer science students brought quayside.app to life using next.js. In the Spring of 2024, we refactored quayside.app in python and built upon the lessons learned from those who worked on it before us. We are proud of our creation and the experience it provides users. The project generator gives valuable insight into projects and the tree and kanban visualizations do a great job of keeping track of everything. That being said, here are some of the biggest lessons we learned.

All of us, in hindsight, think that it was a questionable decision for us to use mongoDB, a noSQL database, for our Django framework backend. Django comes with many pre-made tables and functions that are only compatible with SQL databases. We had to put a lot of effort into

mongoDB to work at all. A big lesson we learned is to cooperate with your framework. Though noSQL has its advantages, it was not worth the trouble.

As for personal things that we learned-
Mya says: I learned a lot about authentication and authorization.
Cam says: I learned a lot about frontend development and web page styling.
Kai says: I gained a lot of experience documenting code and using good practices.

Going into the future, we hope that the lessons we learned help the next group of developers create an even better version of quayside.app .

7. References

PMI. 2024.
https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pmi-pulse-of-the-profession-2024-report.pdf?rev=8d24f62b2b044e00ae21e0388ea9c8bf

Williams, Erik. Summer 2023 quayside.app presentation graphic.

Note: Our Github Repository is featured at https://github.com/quayside-app/quayside

Appendix A: User Manual

1. Go to https://quayside.app/ on a desktop or mobile device.
2. Sign in with Github or Google.
3. Click the “+ Project” button and enter a project name and description to generate a project. You should be brought to your generated project in a few seconds.
4. Edit tasks by clicking on them in a tree. Add new tasks by clicking the “+” icon.
5. View your tasks on a kanban board by clicking the “kanban” icon the project bar. Update the status of your tasks by dragging them to different boards.
6. Edit your project or share it with collaborators by clicking the edit project icon in the project bar.