Team Member Full Name (Alphabetically)	NetID
Nicholas Newton	nnewton2
Parker Scanlon	pscanlo2
Benjamin Swiderski	bswiders

Chosen Technology Stack

Python + Django

Features Implemented for Phase 2

- Create a new task
- Update/Edit an existing task
- Delete an existing task
- Start/Stop a timer for a task
- Update a time entry's begin/end timestamp
- Delete a time entry
- Time Management Dashboard
- Task Management Dashboard

Persistent Storage Design

We are using SQLite database to persist our data. Our database includes tables shown in Figure 1. It contains 1 table. The task table identifies each tuple by a unique identifier id as the primary key. The table also contains the name of the task, its description, the project it is used on, and the status of whether it is open or closed. This information is all received from the user input and they are currently set to NOT NULL. The description is a text field and the other three fields are varchars. Currently the values must be filled out to create a new task. The webpage will let the user know if they have an invalid response. The task table also has a timer state and begin time which are detected when the time is started or ended. There Is also a timestamp table to handle the data recorded for a task run by the timer. This contains the begin and end time, a unique identifier as the primary key, and the task id. The task id is indexed for quick relations and joins with the dask table data.

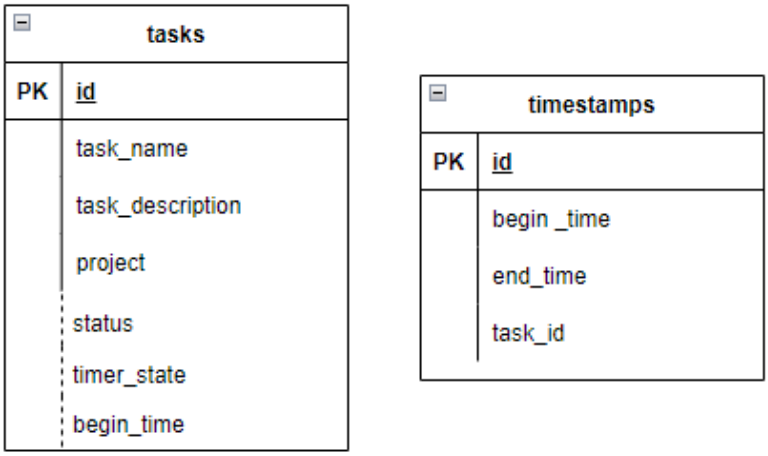


Figure 1. Database schema.

Demonstration of the Features

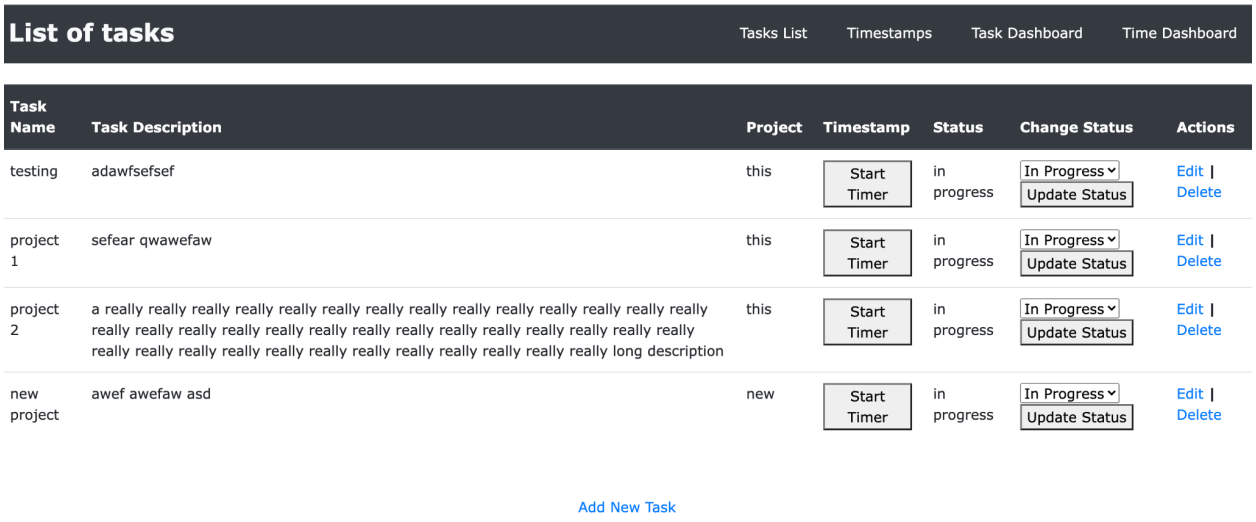


Figure 2. Task Page on 127.0.0.1:8000/ webpage showing tasks.

Create

Figure 3 shows the form to create a new task. It gives textboxes for the task name, description, and project name as well as a choice for status, which defaults to “in progress” rather than “completed”. The form error checks and requires an entry for each field. The form will prompt the user for an entry if none is provided. This form is accessed by clicking the “Add New Task” link from the Task home screen. Once the submit button is clicked, the user is taken back to the Task home screen and the new task is displayed.

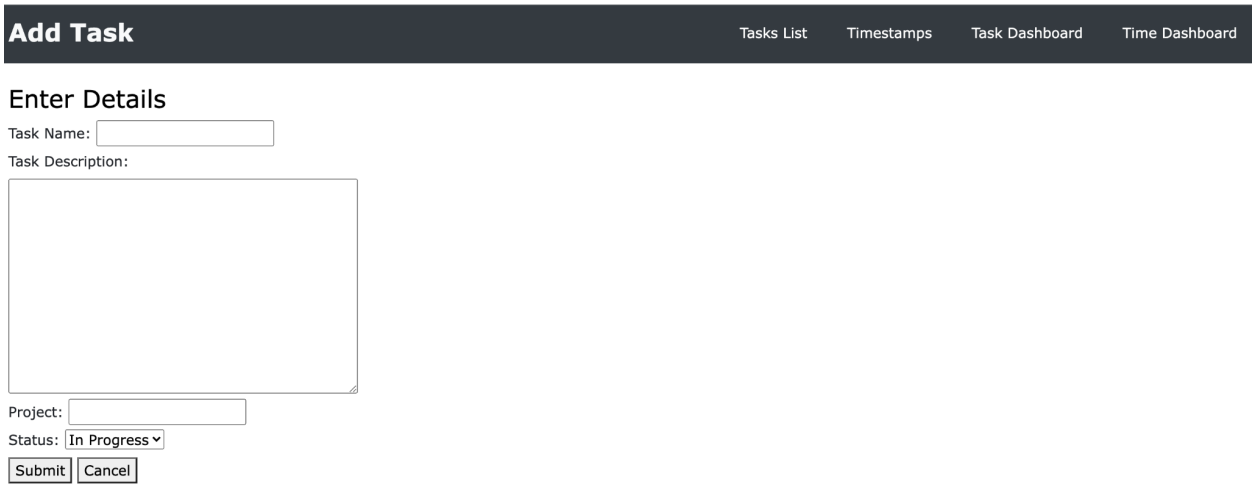


Figure 3. Screenshot for create feature.

Update/Edit

Figure 4 shows the update task form. It gives options to update the task name, description and project name. The task update form can be accessed by clicking on the edit option under the Actions header in the row of the corresponding task. The task status can be changed directly from the Task List home screen by using the selector and clicking “Update Status”, which can be seen in Figure 5.

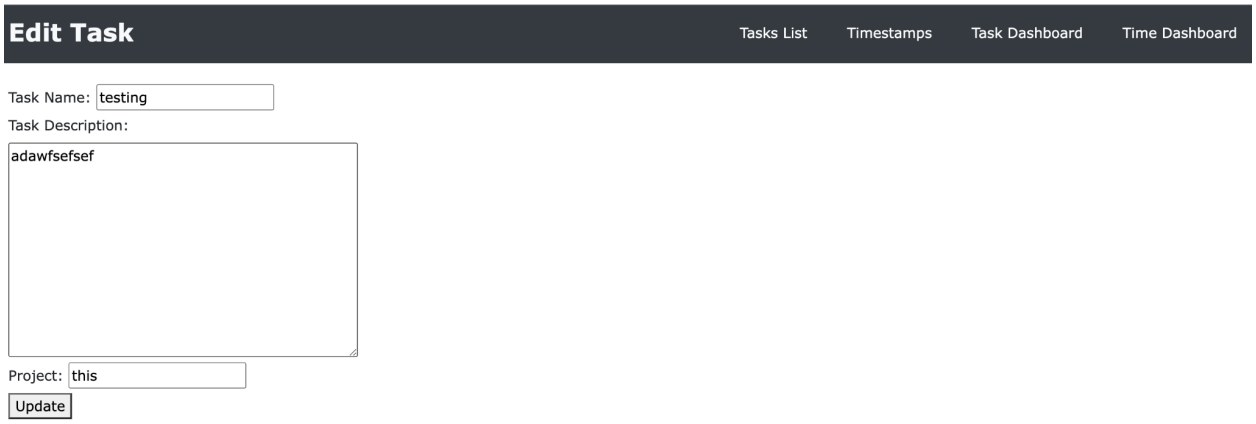
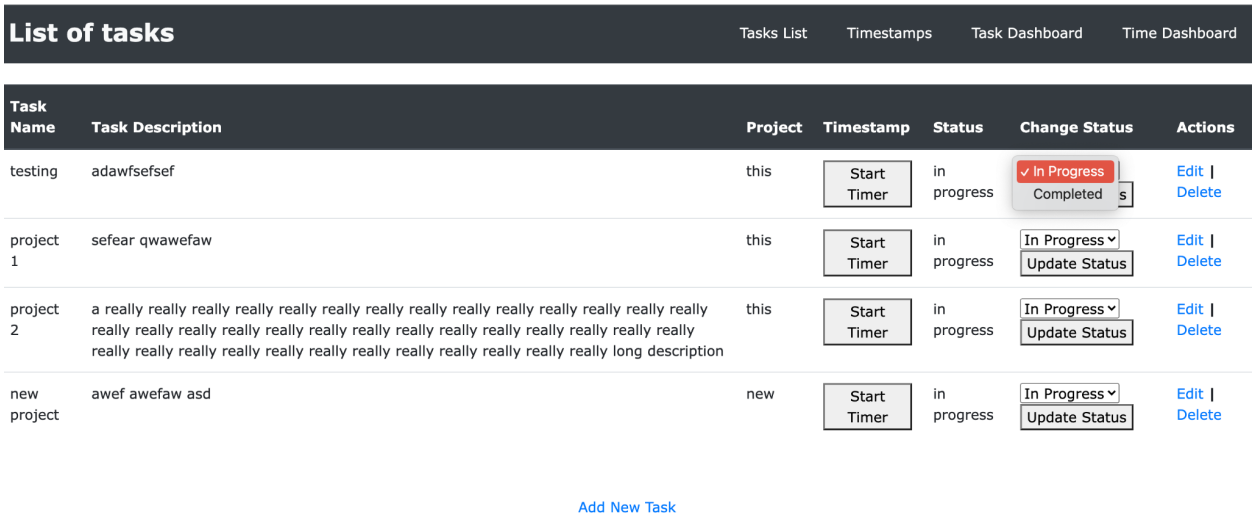


Figure 4. Screenshot for update fields.


List of tasks
Tasks List
Timestamps
Task Dashboard
Time Dashboard
Task Name
Task Description
Project
Timestamp
Status
Change Status
Actions
testing
adawfsefsef
this
Start Timer
in progress
In Progress
Completed
Edit | Delete
project 1
sefear qwawefaw
this
Start Timer
in progress
In Progress
Update Status
Edit | Delete
project 2
a really really really

Timer' and when clicked again, it will return to the 'Start Timer' state and create a timestamp object. These timestamps are all saved to the SQLite database.

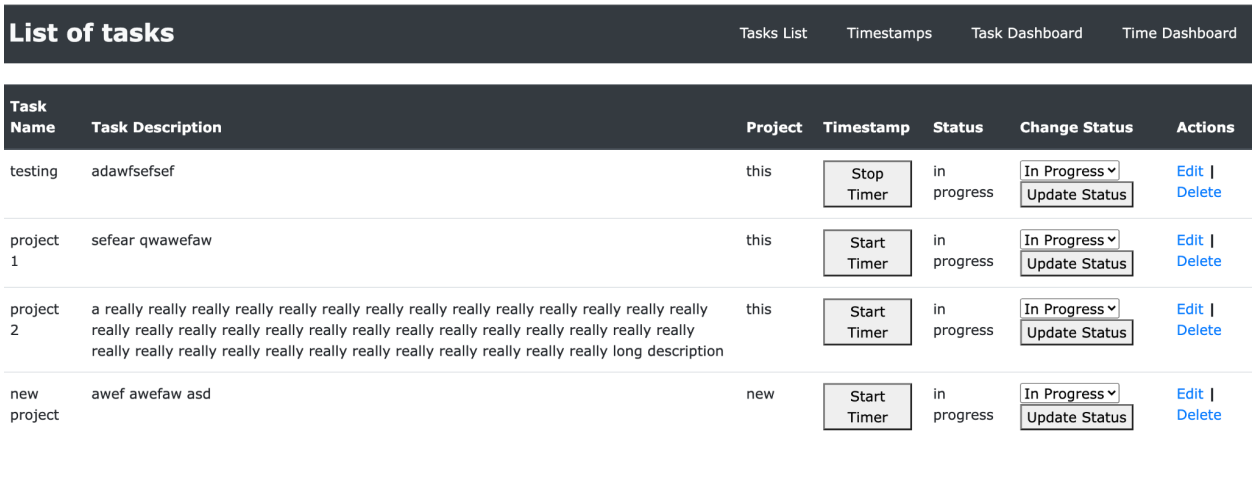


Figure 7. Screenshot Start/Stop Timer.

Update a Time Entry's Begin/End Timestamp

Figure 8 and 9 show the features that are available when a timer is started/ended. Figure 8 displays the list of timestamps in the database by task name. This includes the start/end time, elapsed time, and actions that can be performed on the timestamps. These timestamps are also searchable by task. Figure 9 shows the form to edit a timestamp. This form can be canceled with the cancel button or updated once the calendar and time fields are selected.

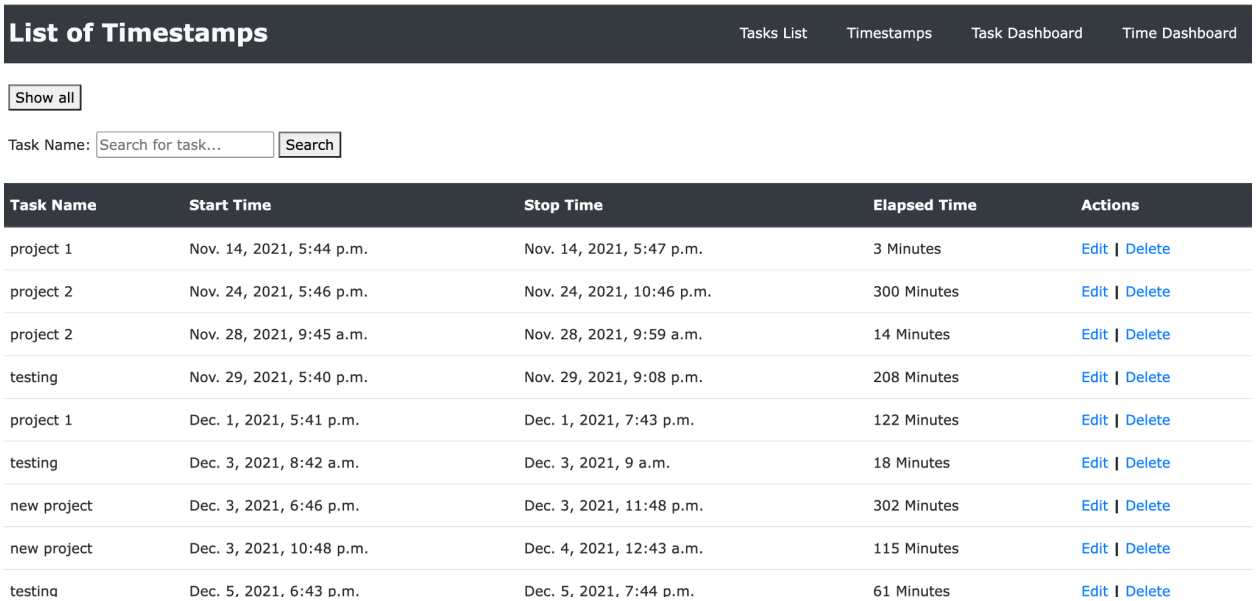


Figure 8. Screenshot of Timestamp Dashboard.

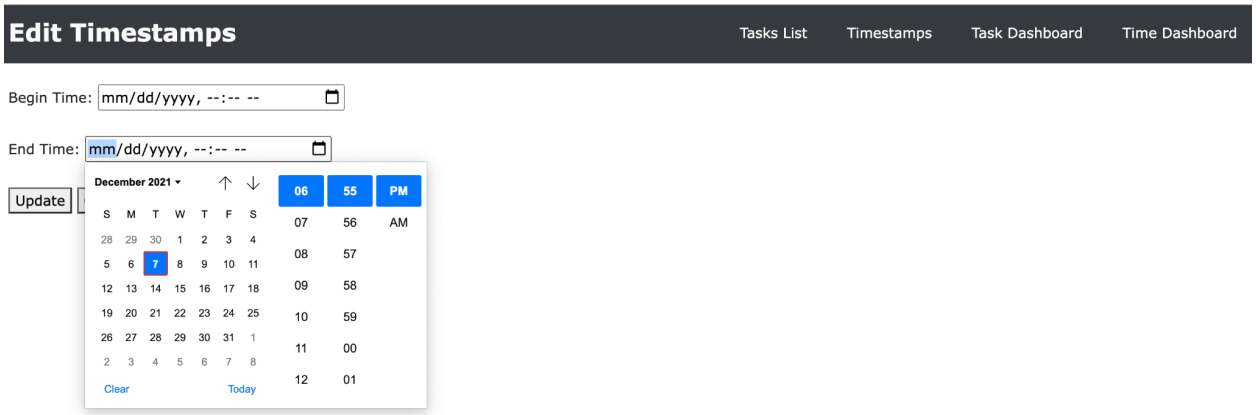


Figure 9. Screenshot for updating start and end timestamps.

Delete a Time Entry

Figure 10 shows the delete function, which is found on the timestamp page. The user can delete a timestamp by clicking the button to the far right of the task under the Actions header.

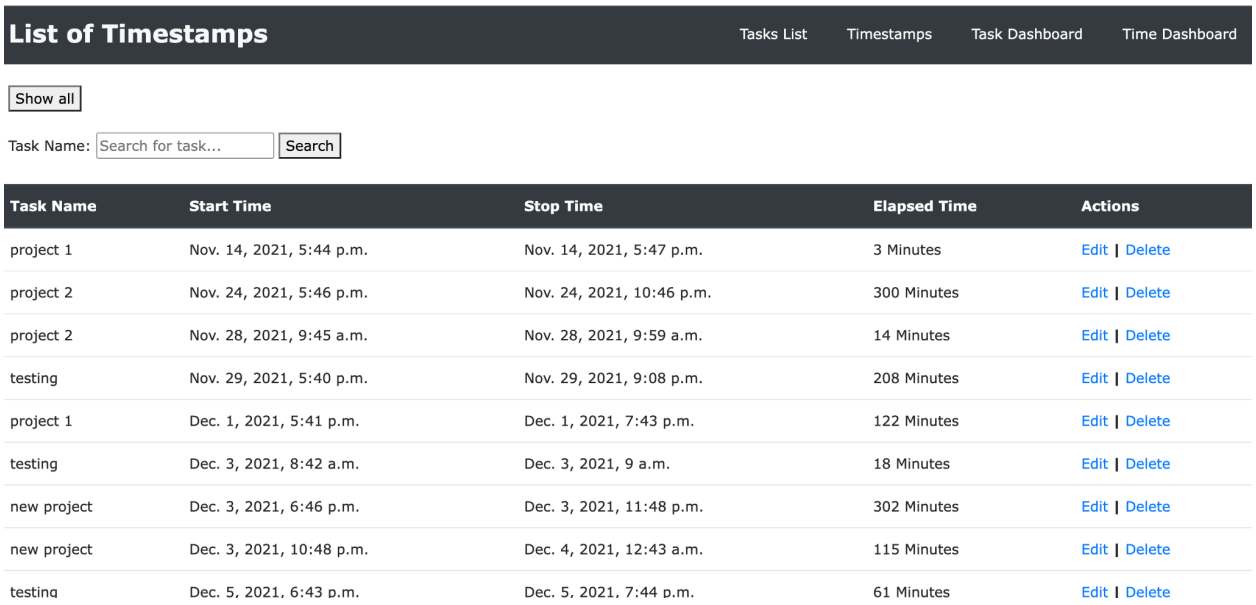


Figure 10. Screenshot for delete timestamp feature (same as Figure 8).

Time Management Dashboard

Figure 11 shows the Time Management Dashboard. This can be filtered by calendar week and month. The week begins on the Monday of the respective day selected. The dashboard results display the task with its description, project, status, total minutes,

minutes per week, and minutes per month spent on the task. In addition, each project has these three metrics displayed below the task results.

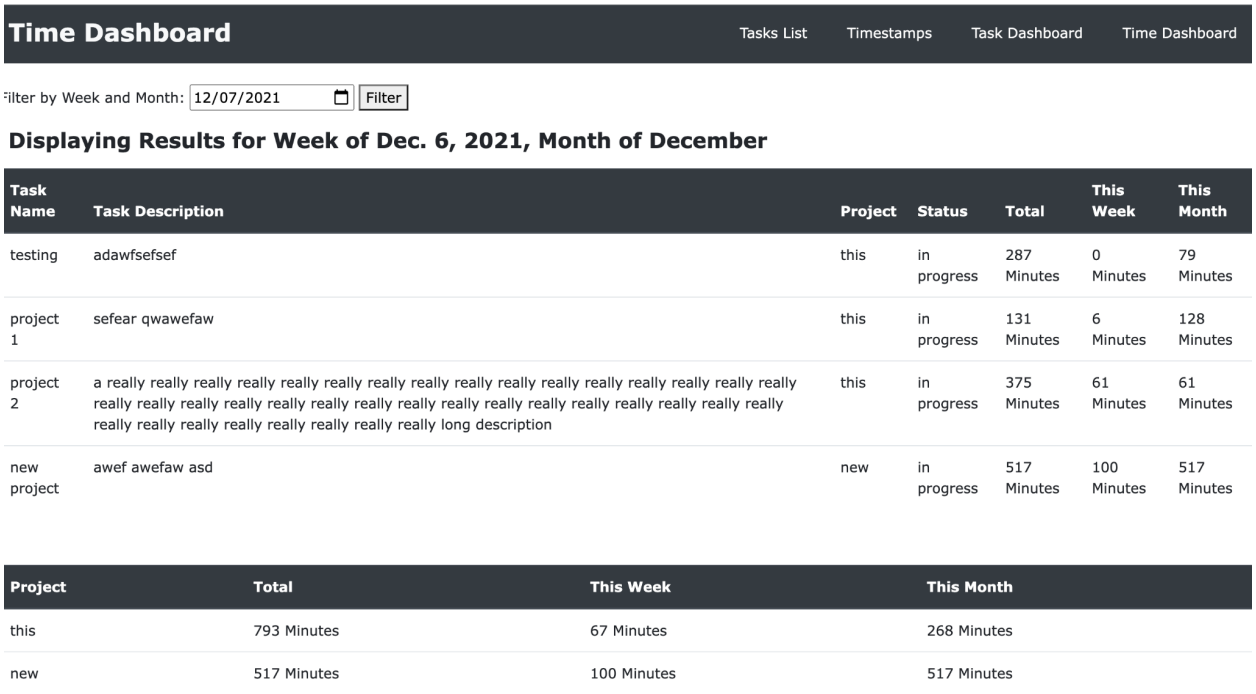


Figure 11. Screenshot for Time Management Dashboard.

Task Management Dashboard

Figure 12 shows the TaskManagement Dashboard. This can be filtered by calendar week and month. The week begins on the Monday of the respective day selected. The dashboard results display the task with its description, project, average total timestamp duration, average weekly timestamp duration, and average monthly timestamp duration. It also displays the percentage of total, weekly and monthly time spent working on each task.

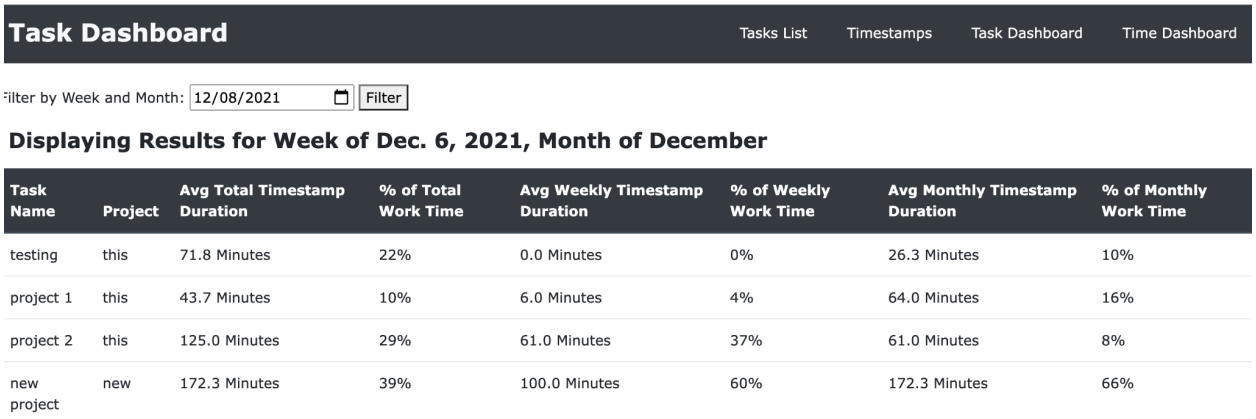


Figure 12. Screenshot for Task Management Dashboard.

Project's Learned Lessons

1. What programming paradigm(s) have you chosen to use and why? If you were to start the project from scratch now, would you make different choices? Do you think the paradigm(s) chosen helped (or not) in developing the project?

Our group used a Django framework with a SQLite database and html/css for the frontend. This worked very well for our project. In other classes some of us had used a LAMP stack with PHP and MySQL and found the tech stack we used for Paradigms to be a much more programmer friendly experience. Using Django also organized the project files well.

2. Would you plan the release of the project's features differently? That is, did your plan result in releasing something too early, when another feature could just as well have been included in its stead?

We believe that the order we implemented the features from Phase 1 and throughout Phase 2 was optimized so there was not an issue releasing a feature too early. We knew that we needed task CRUD to be completed before any timer or timestamps could be made so the timer and timestamps could be connected to their respective tasks. We also knew that the dashboards would need to be implemented last. They both required calculations on data involving features that created the timestamps and tasks.

3. Identify risks you encountered, those that never occurred, and problems that arose that you didn't expect or plan for. How would you want to deal with risk in future projects?

There were some small quality of life features that we could not figure out how to implement with Django. When editing a timestamp, the code always returns to the full timestamp page. In other words, if a user searched a task and then edited the timestamp, upon returning to the list of timestamps, the page shows the full table again instead of just the previously searched tasks. We were able to make this work for the delete timestamp feature. Another unforeseen problem had to do with the communication between HTML and Django for timestamps. We had to do extra work to allow HTML's DateTime picker and Django's DateTime field to talk to one another. Initially we also wanted to make the week and month filter an HTML week and month picker, not a general date picker. However, there was little to no support from django for those input types so we changed to a date input. As we work on future projects it is important to not get too attached to the initial idea when a serviceable, much easier to implement alternative exists.

4. What were the most challenging aspects of the project?

The most challenging aspect of the project was getting a handle of the Django framework and nuances in the MVT code. It took a lot of time to get a base understanding how models, forms, views, and templates all interact. As a result, the most time was spent implementing the CRUD features for creating and displaying tasks. As we wrote more code, we became more familiar with the Django development pattern and the timer and dashboard features became easier to implement.

5. What aspects of your process or your group's organization had the largest positive effect on the project's outcome?

Utilizing GitHub with individual branches as well as a Dev branch was definitely one of the most helpful organizational aspects for our group. It allowed us to each work independently of each other and commit different features on the same files concurrently. We also were fortunate to have good communication throughout the project so deliverables were completed comfortably on time and there was no confusion between division of work.