

Team Member Full Name	NetID
Megan Cater	mcater
Coleen Gillilan	cgillila
Charles Korndorffer	ckorndor

Chosen Technology Stack

☒ Python + Django

☐ Python + TkInter

Persistent Storage Design

We are using the SQLite database to persist data. The database consists of two tables, `Assignment` and `Timer`, that are shown in Figure 1. The `Assignment` table has the primary key of `id` and attributes `name`, `due_date`, `class_name`, `description`, and `completed`, and the `Timer` table has the primary key of `id`, foreign key of `assignment_id_fk` and attributes `begin` and `end`.

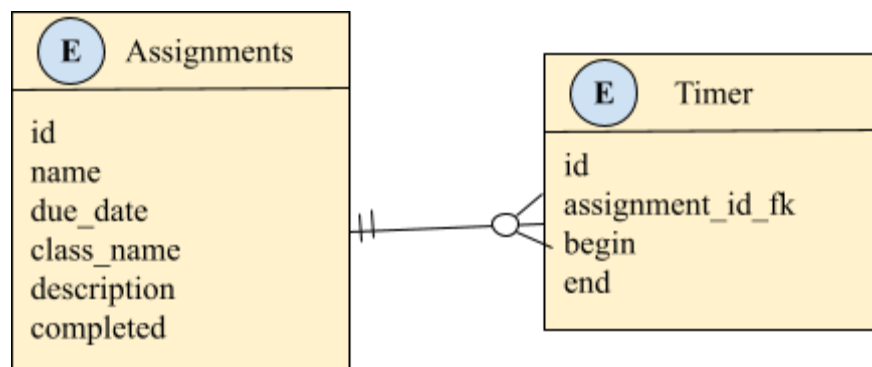


Figure 1 database schema

Demonstration of the Features

Homepage

Figure 2 shows a screenshot for the main page when the project is being run. This includes a list of all the homework assignments that currently exist in the database. Each assignment has the option to edit, delete, and start/stop a timer for the assignment. When the button is clicked, then the page is redirected to the appropriate action. The add assignment button redirects to the create page, the time dashboard button redirects to the time dashboard, and the average dashboard button redirects to the average dashboard page. There is also the option to edit and delete each of the timers for the assignments.

Homework Assignments

[Add Assignment](#)[Time Dashboard](#)[Average Dashboard](#)

1. Reading 01

[Edit](#)[Delete](#)[Timer](#)

- Class: Operation System Principles
- Description: Test
- Due Date: Sept. 23, 2021
- Completed: False
 - [Edit](#)[Delete](#) Timer 1: Dec. 4, 2021, 2:14 p.m. - Dec. 4, 2021, 2:19 p.m.
 - [Edit](#)[Delete](#) Timer 2: Dec. 9, 2021, 5:49 p.m. - Dec. 9, 2021, 6:52 p.m.
 - [Edit](#)[Delete](#) Timer 3: Dec. 5, 2021, 5:51 p.m. - Dec. 5, 2021, 5:51 p.m.
 - [Edit](#)[Delete](#) Timer 4: Dec. 6, 2021, 10:50 a.m. - Dec. 6, 2021, 10:50 a.m.
 - [Edit](#)[Delete](#) Timer 5: Dec. 6, 2021, 10:50 a.m. - Dec. 6, 2021, 10:50 a.m.
 - [Edit](#)[Delete](#) Timer 6: Dec. 6, 2021, 10:55 a.m. - Dec. 6, 2021, 11:49 a.m.

2. Project 01

[Edit](#)[Delete](#)[Timer](#)

- Class: Operation System Principles
- Description: Proect
- Due Date: Sept. 30, 2021
- Completed: False
 - [Edit](#)[Delete](#) Timer 1: Dec. 5, 2021, 12:09 p.m. - Dec. 5, 2021, 12:32 p.m.
 - [Edit](#)[Delete](#) Timer 2: Dec. 6, 2021, 10:50 a.m. - Dec. 6, 2021, 10:50 a.m.

3. Reading 3

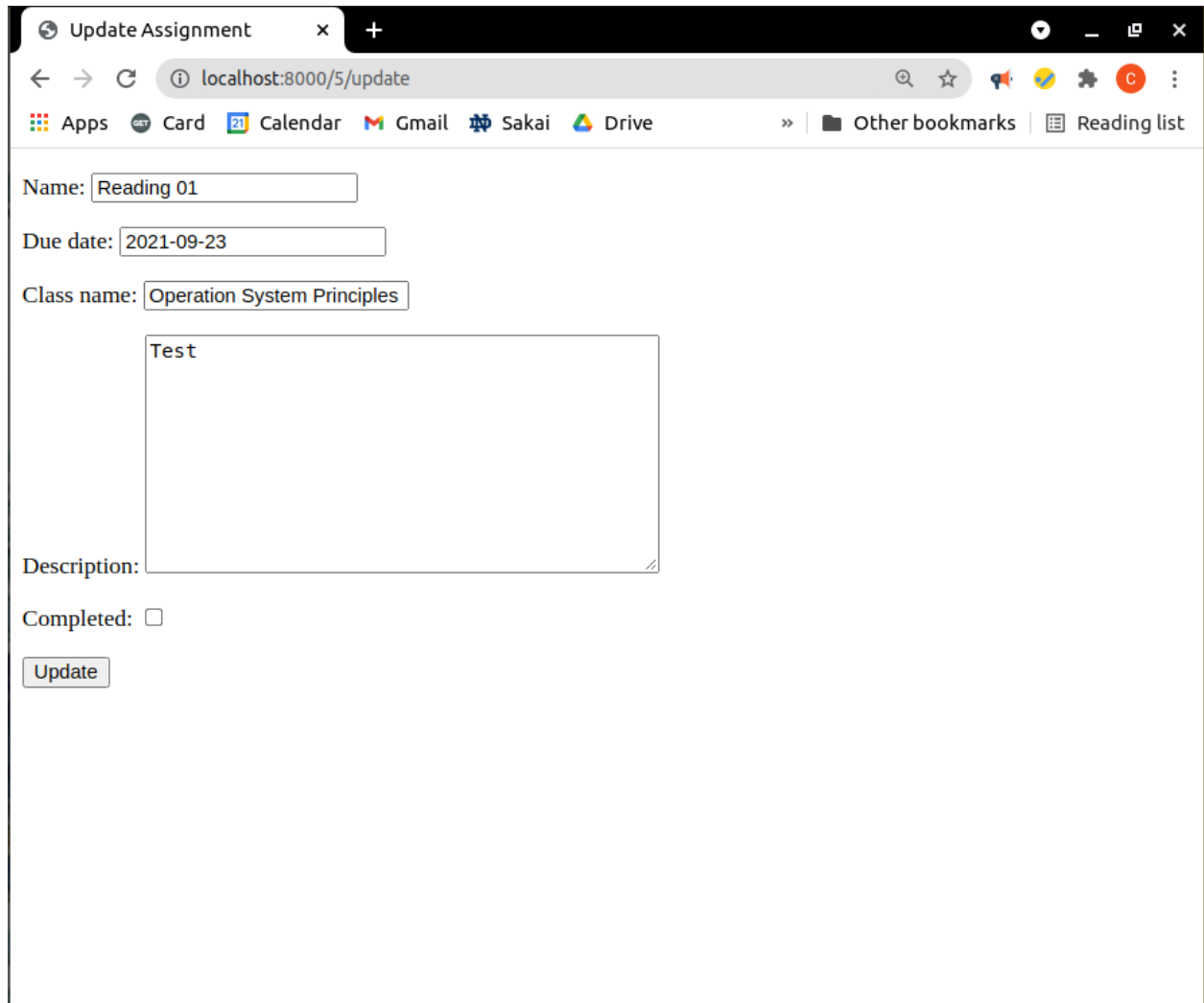
[Edit](#)[Delete](#)[Timer](#)

- Class: Operation System Principles
- Description: some description
- Due Date: Oct. 8, 2021
- Completed: True
 - [Edit](#)[Delete](#) Timer 1: Dec. 5, 2021, 5:55 p.m. - Dec. 5, 2021, 5:56 p.m.

Figure 2 Screenshot showing the home page

Update a task

Figure 3 shows a screenshot for the update page. In this page, the data in the database is pre-loaded into the fields and can be edited. Once changes are made, then the update button is hit. This brings the user back to the home page and the changes saved. If any change is invalid, it will not allow the user to save.



Update Assignment

localhost:8000/5/update

Name:

Due date:

Class name:

Description:

Test

Completed: ☐

Figure 3 screenshot for updating a task

Create a task

Figure 4 shows a screenshot for creating an assignment. It is similar to the update page except no fields are pre-filled. The user has to fill all fields with valid inputs for the changes to be saved. Once submit is hit, then the values are saved to the database and will show up on the home page.

The screenshot displays a web browser window with the title 'Create Assignment'. The address bar shows the URL 'localhost:8000/create'. The browser's bookmark bar includes links to 'Apps', 'Card', 'Calendar', 'Gmail', 'Sakai', 'Drive', 'Other bookmarks', and 'Reading list'. The form itself contains the following elements:

- Name:** A text input field.
- Due date:** A date input field.
- Class name:** A text input field.
- Description:** A large text area for entering details.
- Completed:** A checkbox.
- Submit:** A button to save the assignment.

Figure 4 screenshot for creating a task

Delete a task

Figure 5 shows a screenshot for deleting an assignment. The page confirms that the user wants to delete the assignment they have selected. If so, the assignment is deleted from the SQLite database. If not, the user is returned to the home page.

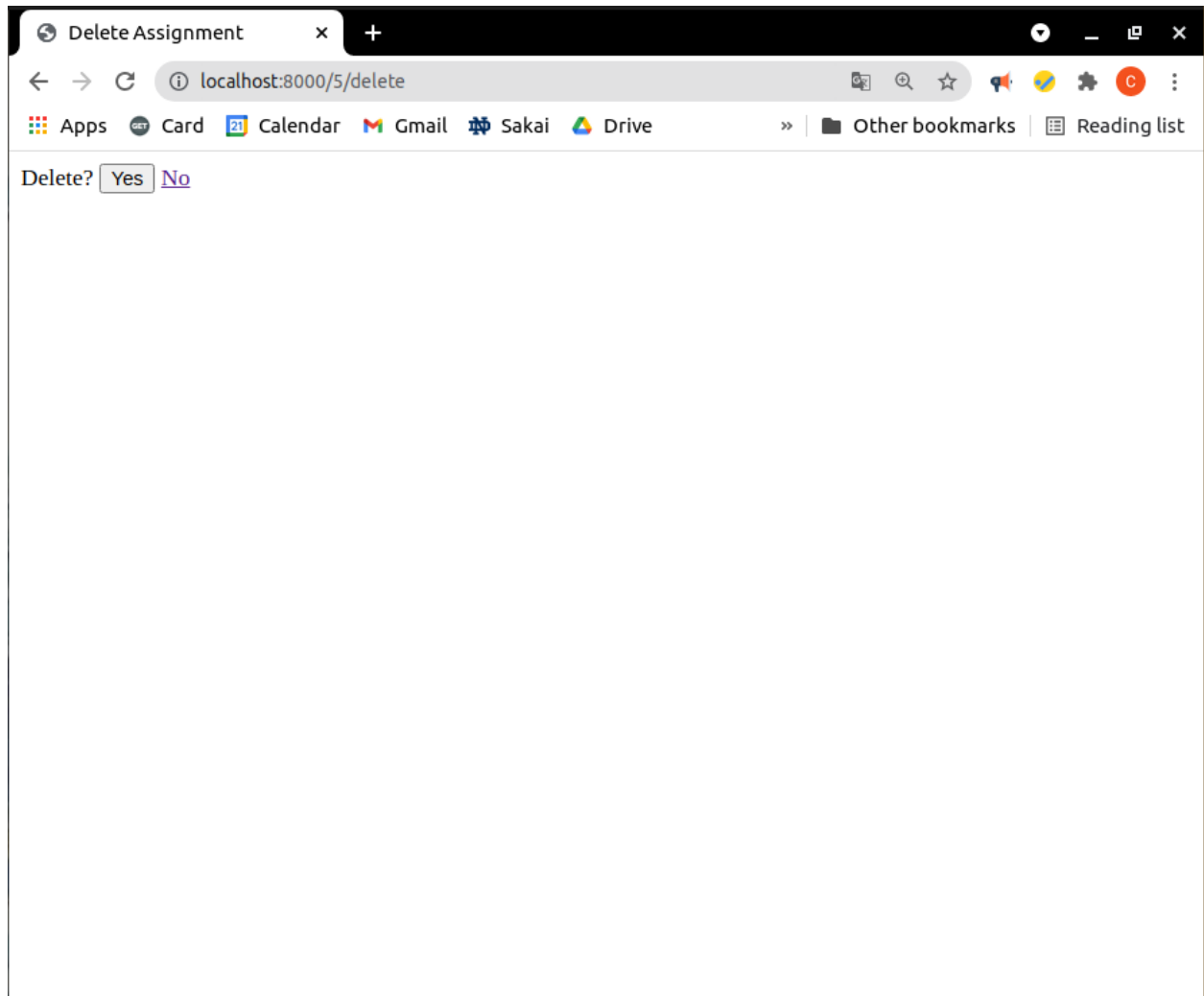


Figure 5 screenshot for deleting a task

Create a Timer

Figure 6 shows a screenshot for creating a timer. The timer is attached to a homework assignment, whose id is in the url. The “Start Timer” button creates the timer and then updates the **begin** attribute for the assignment, and the “Stop Timer” button updates the **end** attribute for the assignment. Upon pressing “Stop Timer,” the user is redirected to the assignments page.

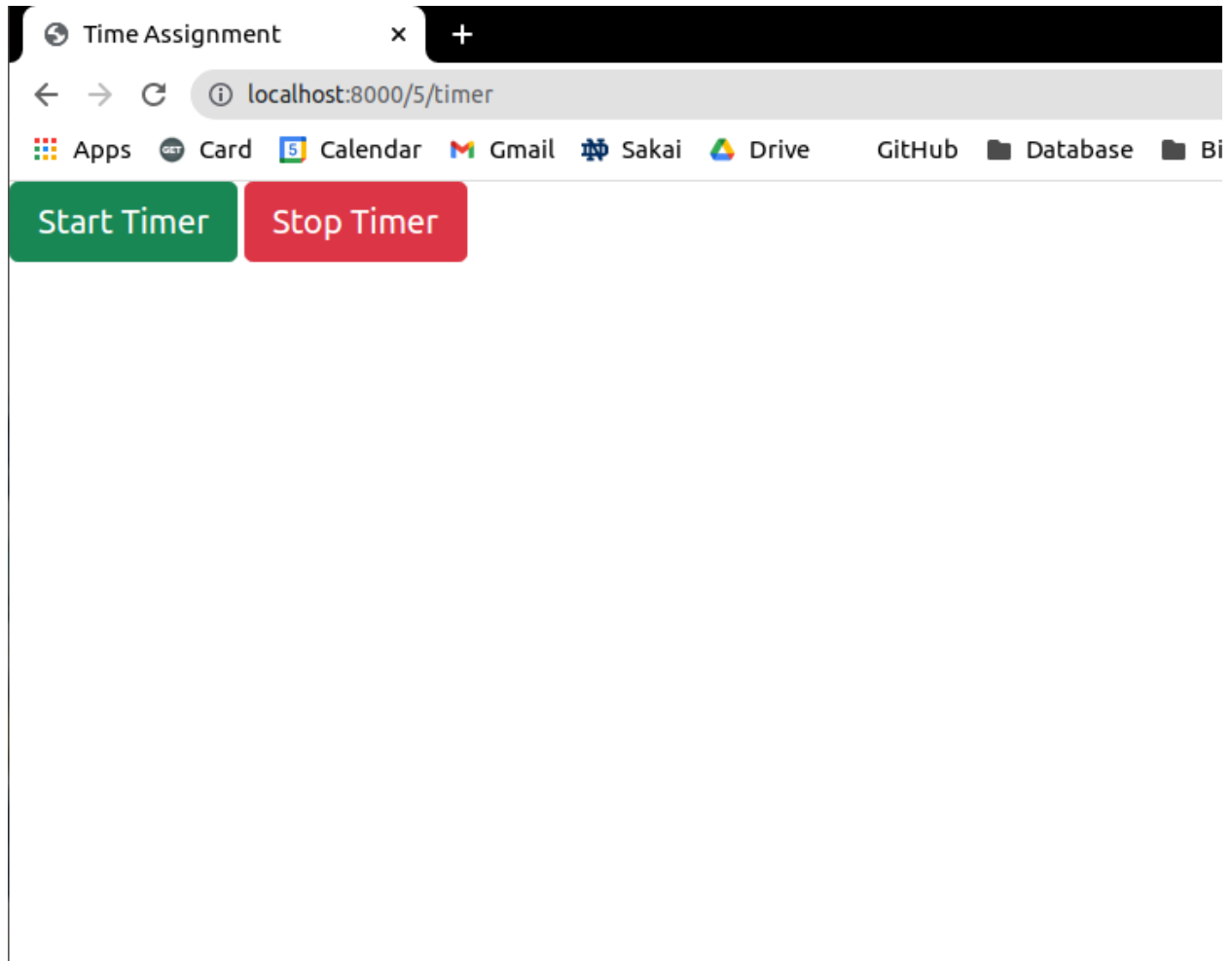
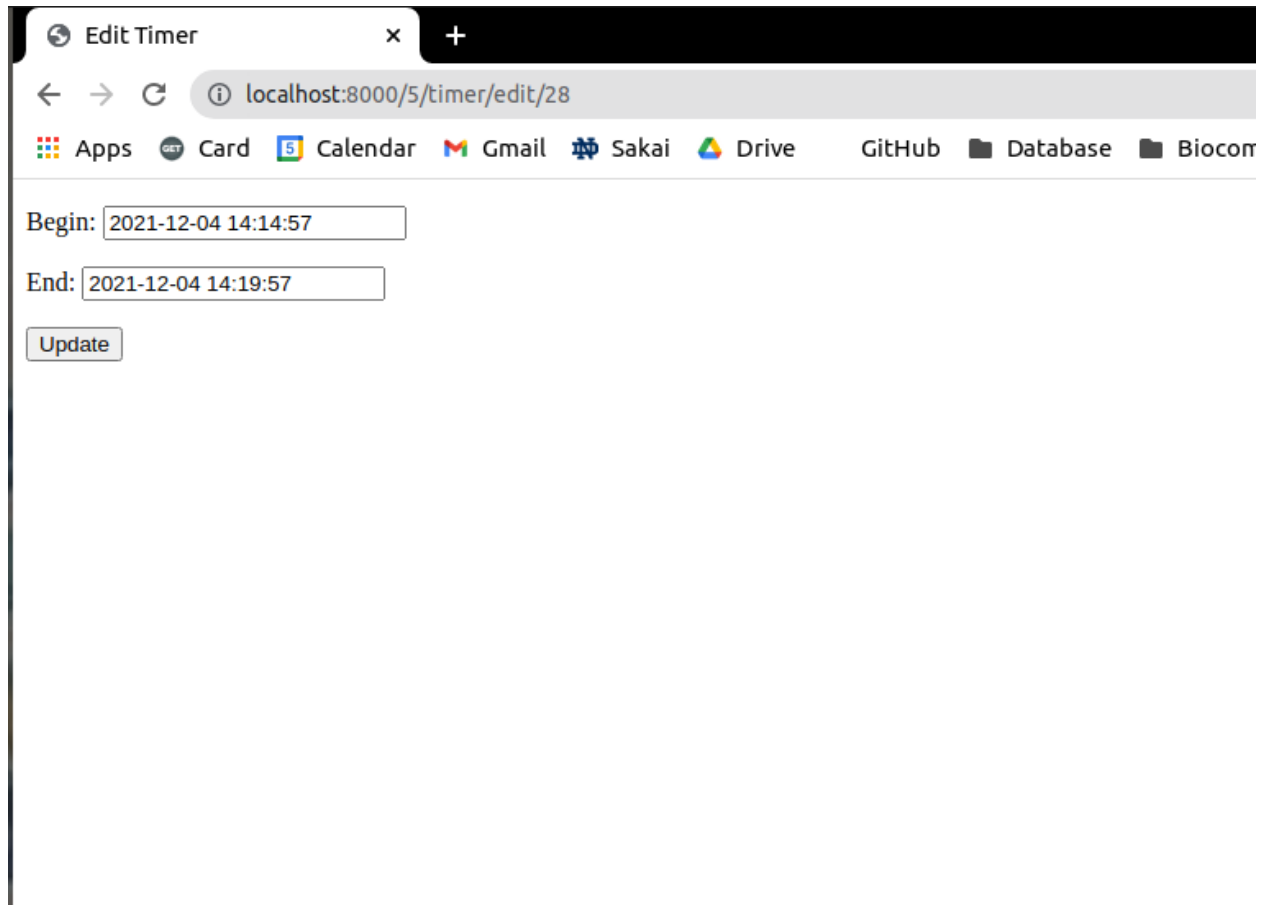


Figure 6 screenshot for creating a timer

Edit a Timer

Figure 7 shows a screenshot for editing a timer. In this page, the data in the database is pre-loaded into the fields and can be edited. Once changes are made, then the update button is hit. This brings the user back to the home page and the changes saved. If any change is invalid, it will not allow the user to save.



Edit Timer

localhost:8000/5/timer/edit/28

Apps Card Calendar Gmail Sakai Drive GitHub Database Biocon

Begin: 2021-12-04 14:14:57

End: 2021-12-04 14:19:57

Update

Figure 7 screenshot for editing a timer

Delete a Timer

Figure 8 shows a screenshot for deleting a timer. The page confirms that the user wants to delete the timer they have selected. If so, the timer is deleted from the SQLite database. If not, the user is returned to the home page.

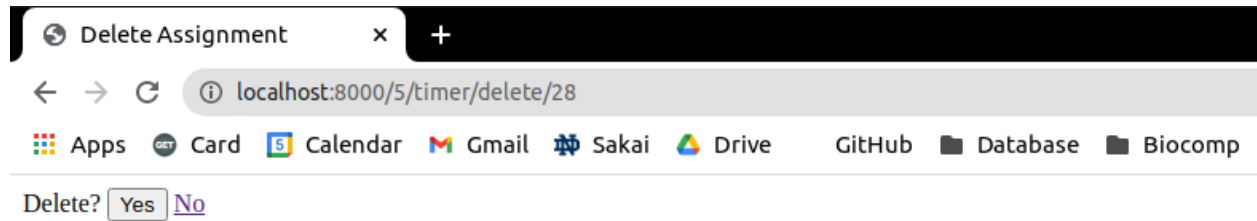


Figure 8 screenshot for deleting a timer

Time Management Dashboard

Figure 9 shows a screenshot for the time management dashboard. On this page, the user can see how many total minutes they have spent per assignment and per class on a given week or month. The assignments are listed on the left and classes on the right, each with the metric calculating total time per assignment/class. When the dashboard loads, it defaults the month and week to the current month and week, but a form in the far left corner allows the user to select which month/week they would like to see metrics for. This form only allows the user to select months and weeks that have timers associated with them.

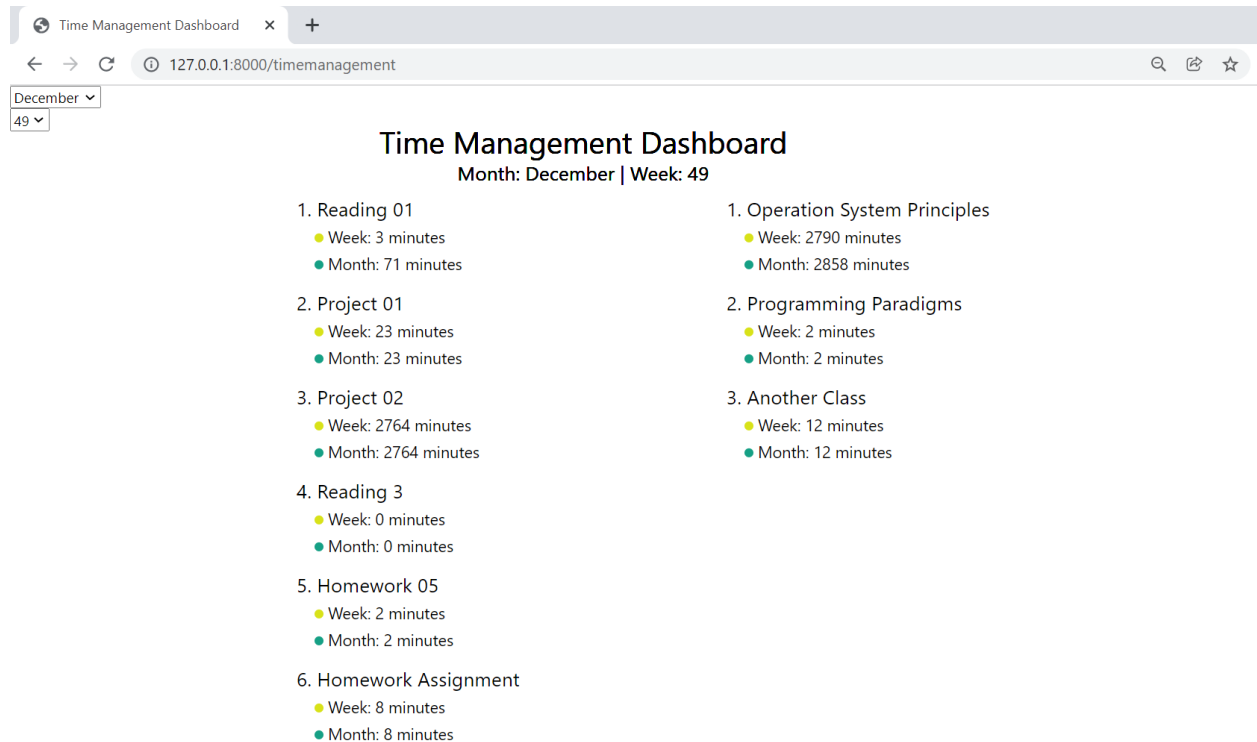


Figure 9 screenshot for time management dashboard

Average Time Management Dashboard

Figure 10 shows a screenshot for the time management dashboard. On this page, the user can see the average time spent per assignment on a given week or month. The assignments are listed with their metrics. When the dashboard loads, it defaults the month and week to the current month and week, but a form in the far left corner allows the user to select which month/week they would like to see metrics for. This form only allows the user to select months and weeks that have timers associated with them.

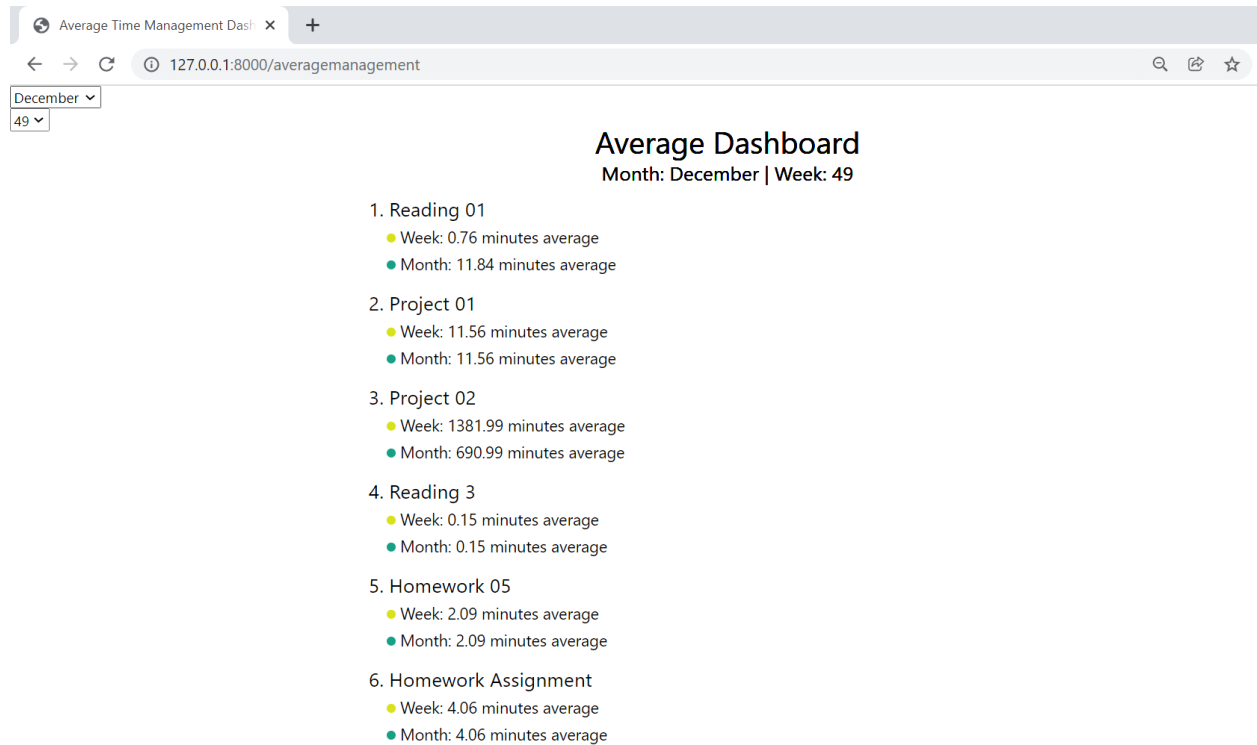


Figure 10 screenshot for average time management dashboard

Project's Learned Lessons

In this section, you will reflect on the group's activities and performance through the entire project, and capture your reflections, observations and thoughts. It should address the following items, but feel free to expand on these in light of your group's unique experiences.

- 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?***

We chose to use Django to implement the project, which itself is Python-based and uses the model-template-views pattern. Python is a multi-paradigm language that supports object-oriented programming, structured programming, and functional programming. We chose to use Django and its associated programming paradigms because the combined features and the ease of Python allow for the creation of a web application and user interface with helpful tools while also being able to expand upon the base and implement more advanced features (such as the dashboards). Furthermore, we approached the interface as function-based rather than class-based due to the explicit and straightforward nature in creating the views as function-based.

- 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 include in its stead?***

In planning the project, we decided to implement the form feature of the dashboard last. However, the way we designed our dashboard template did not allow us to make use of a form; we would have needed to refresh a div tag inside of dashboard's html or pass parameters to the html page, but Django would not allow us to do either due to the way we structured the html. Because of this, if we were to plan the release of the project's features differently, we would have had both the display of the metrics and the form feature of the dashboard planned to be implemented at the same time. This way, they would have been worked on in conjunction and the problem we faced with the dashboard would have been realized sooner and avoided. Otherwise, we would keep the release of the project's features the same.

- 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?***

In developing our Django web application, we deviated from the way we were taught to use Django as we were thinking about how to implement it, which inevitably led to the risks of trial and lots of error. For example, we first attempted to use the REST framework with ReactJS, but since none of us were familiar with ReactJS and using the JavaScript library was more complicated than we had anticipated, this approach had to be dropped. Eventually, we were able to apply function-based views in order to implement the first several features of the application, which worked as we had intended it to. However, as mentioned previously, we did not expect to have as much trouble with the dashboard as we did; as much as we tried manipulating the templates, our plan to create the dashboard was initially unsuccessful, and since we were hesitant to scrape much of our work entirely, we had to devise a new plan. In future projects, we would want to deal with risk by having a clearer understanding of the frameworks and software we are planning to use, as this lack of understanding is what resulted in hitting deadends. We would also

do more small-scale testing to make sure our ideas were feasible instead of attempting to write a whole feature only to realize it doesn't work the way we want it to.

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

The most challenging aspect of the project was working around the quirks of Django. Some things that could be done easily in php, for example, require much more workaround in Django. This was the case with form feature, which took up the largest amount of time to implement. Ultimately, we chose to use the D3 JavaScript library to help us implement this, as one of our group members was familiar with it. Figuring out how to create the web application in general was also very challenging. As none of us had used Django before this class, we were still very new to it and lacked understanding of its intricacies. This resulted in us trying many different things that were unsuccessful, and we had to keep rethinking how to approach the problem. There were many, many Google searches.

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

Our ability to bounce ideas off of each other allowed us to overcome problems when we faced them, while our splitting up of the work helped us implement the project's features in a timely manner. We would meet as a group to discuss our current approach to the project, work together on some of the features, and then assign roles to handle the rest of the features. This aspect of our process/organization made sure that everyone was on the same page and did equal work while also giving us time to discuss where we were struggling and how we could fix these problems. Our use of GitHub also allowed us to coordinate and keep an eye on the others' progress. After one group member pushed, another would take a look at the work so far and fix any issues if needed, or help to clean up the code. All of these aspects combined resulted in us completing the project to the best of our abilities.