Team 4

Potatimer MVP Project Pitch

Potatimer Project on GitHub

Professor:
Thomas A. Powell

Team Advisor:
Deepak Karki

Members:
Alexis Chen —— Team Lead/Manager
Elizabeth Cho — QA
Kevin Jang ——- UI, Docs
Marco Kuan —— Backend
Ahmad Milad —- Frontend
Rohan Patel ——-Backend
Miaoqiu Sun —— Frontend
Jessie Zuo ———- Team Lead/Manager

Introduction

Our team is planning to develop a Pomodoro Timer in the theme of a task to cultivate and harvest potatoes as they grow over time. The end-goal is to allow a user to customize the work and break times to accommodate their needs; change the theme of our timer to minimalist light, dark, and potato themes; and various options to mute sound or reset the pomodoro cycle.

In order to meet this goal, we will create the timer in three phases that partition the remainder of the quarter. Phase 1 is about building a fully-functional MVP timer through a focus on planning, research, and backend work. We expect to have this product complete by the end of the sixth week. Phase 2 would add optional features such as a settings menu to contain most of the new functions that accommodate a wider consumer base. This would require more frontend work, along with coordinating functionality with the backend team and the QA. We would permit at most three weeks to the end of week nine for this phase. Finally, the final phase would finalize our timer so it works without bugs and consists of proper documentation.

We will be using HTML, CSS, and JavaScript as required by restrictions. HTML lets us create a website application for our timer. The programming language's primary use is to create containers for many of the interactive elements, such as buttons and formatting elements next to each other. With CSS we can style the layout of the website so users can read large text and notice where container boundaries, such as button outlines, are located with ease. Integrating JavaScript into the website application through the Document Object Model (DOM) gives the timer its functionality. Since the timer is primarily an on-click interaction, the user would click on elements and trigger JS functions to execute what the user desires. These three programming languages would create a desirable pomodoro timer for the project specifications.

Phase 1: Minimum Viable Product (W4-6)

The Timer Problem

Based upon our Week 4 research, we have identified the following components of a basic pomodoro timer: an interactive website application, start and stop buttons, and a visual timer that counts down. The pomodoro cycle consists of four intervals which follow a longer break at the end, usually three times as long as the short breaks.

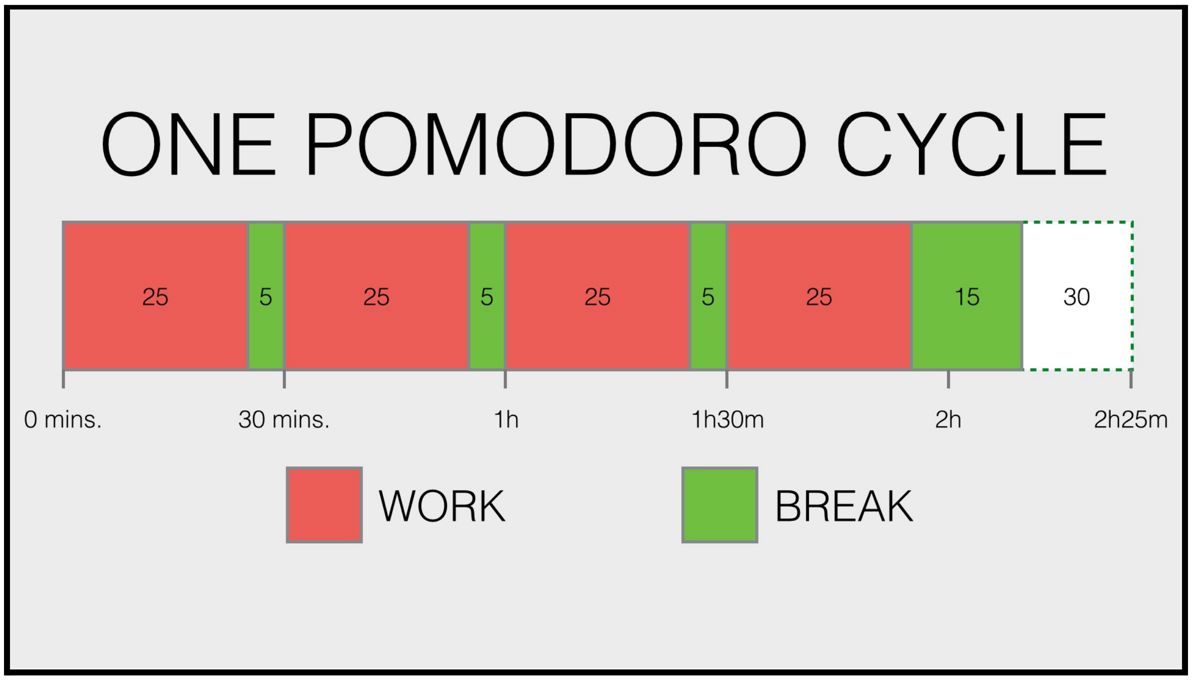


Figure 1: The Pomodoro Timer Cycle

On page-load, the timer is already set to the default work length. The start button is meant to initiate the timer and the stop button is automatically pressed once the timer reaches zero. Most timers show that at the end of each work and break interval, the timer stops and the user must manually start the timer again. We think this is counterintuitive because it leads to the rigid break time being longer than it should be. Therefore we plan to make our potato timer a running timer that doesn’t stop unless the user presses the stop button.

User Stories

Our main audience is students, but the general public should be capable enough to use the timer. The user interface won't be too cluttered and the buttons would be noticeable for people to see without resizing the page. In addition, students who study in the evening should keep their eyes healthy by having less blue-light exposure. Therefore we will accommodate nighttime users with a dark theme that doesn't irritate users at night. Both the default light and dark themes will have minimal colors so colorblind users may use the website application without any visual issues.

Time Allocation

With less than six weeks in the remainder of the quarter to build the timer, we will prioritize a functional MVP over additional features in this phase. The website application shouldn't take much time to create the buttons and displays for the timer, so we anticipate that this part should take a few hours at most. However, the JavaScript scripts that interact and change the HTML, such as button responsiveness and counting down the timer, should take more time to build. This would take a few days to design and create, so an exploratory testing approach would be ideal. Because this basic timer is very minimalist, the QA should be reviewing the webapp for about one or two days. The difference between the light and dark themes is a visual change in CSS colors, so this shouldn't take more than two hours to do.

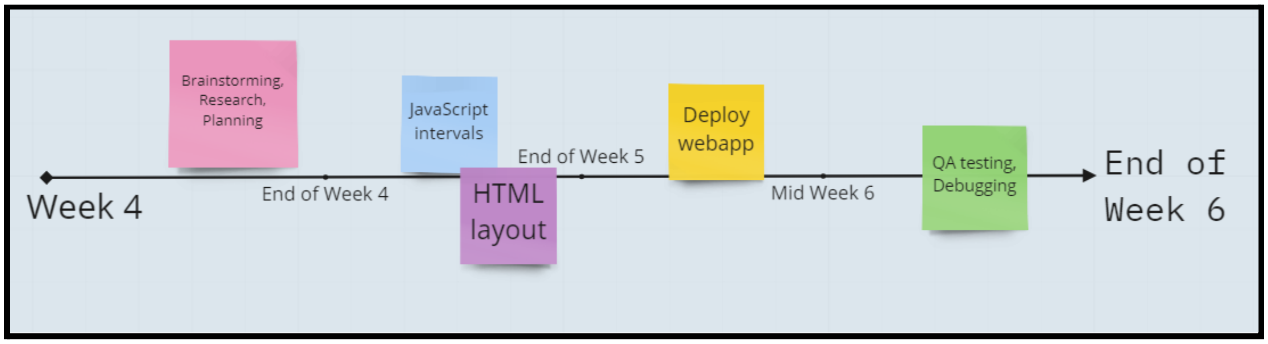


Figure 2: Timeline of Development

We will be using [GitHub Issues] and [GitHub Project Board] to document our tasks and progress. The project board is automated so whenever someone creates a new issue it will be placed into a TODO section of the board. This structure relieves the disarray of trying to have real-time communication and multiple chat threads in Slack. In addition, we will use a checklist of requirements when submitting a pull request to the master branch. This would help prevent push mistakes because each version has to be verified by the managers. We think this is an important requirement to have that its trade-off is better than having quick, unverified pushes onto master.

Proposed Solution

To start off, we plan to hard-code the length[1] of the work, break, and long break intervals as 25, 5, and 15 minutes respectively. This implementation would help the developers on our team test to see if there are any mathematical operations that must be done on the scripted time in order to simulate these exact minutes. Once this is done, we would understand how timing is emulated between the frontend and backend. In the next phase we will shift to replacing these values with user-inputted lengths stored in variables.

The website application for our timer would consist of a button for the start and stop buttons with a paragraph element identified by an id as the timer. By marking what element is the timer, we specify what element is being modified without accidentally modifying all elements of that type. When the start button is clicked, a JavaScript function is executed to begin modifying the timer element's value. This value is changed to simulate a countdown by seconds. We would be keeping track of what interval and phase the timer is in so that the pomodoro cycle property is met. Changing the intervals and phases would happen by conditionals and variable assignments in JavaScript. For example, on page load the interval is 1 of 4 and the phase is set to "work". Once the timer reaches 00:00, the time is reset to the break time and the phase is set to "break". After the timer reaches 00:00, the interval increments and the phase is set to "work". This process repeats until it is the fourth interval. On the fourth interval a conditional statement would read the interval and then set the phase to a "long break". When this break is over, the interval resets to one and the stage goes back to "work".

We want to make the timer visually appealing and accessible for the users, so we have sketched out a rough design of what the website would look like [See Figure 3]. The user would see the project name, the timer, and the start and stop buttons in the center of the page. We have decided to implement a developer display[2] that shows the current interval and phase for debugging the cycle. This developer mode allows the team to debug the code using passive indicators and it can be hidden by adjusting the code internally, so users won't be affected. In order to accommodate the visually impaired and users that leave the timer running in the background, a sound is played to notify the user that the current phase is over. We plan to make the sound adjustable in the next phase.

[1] These overarching decisions can be found in specs.
[2] Ibid

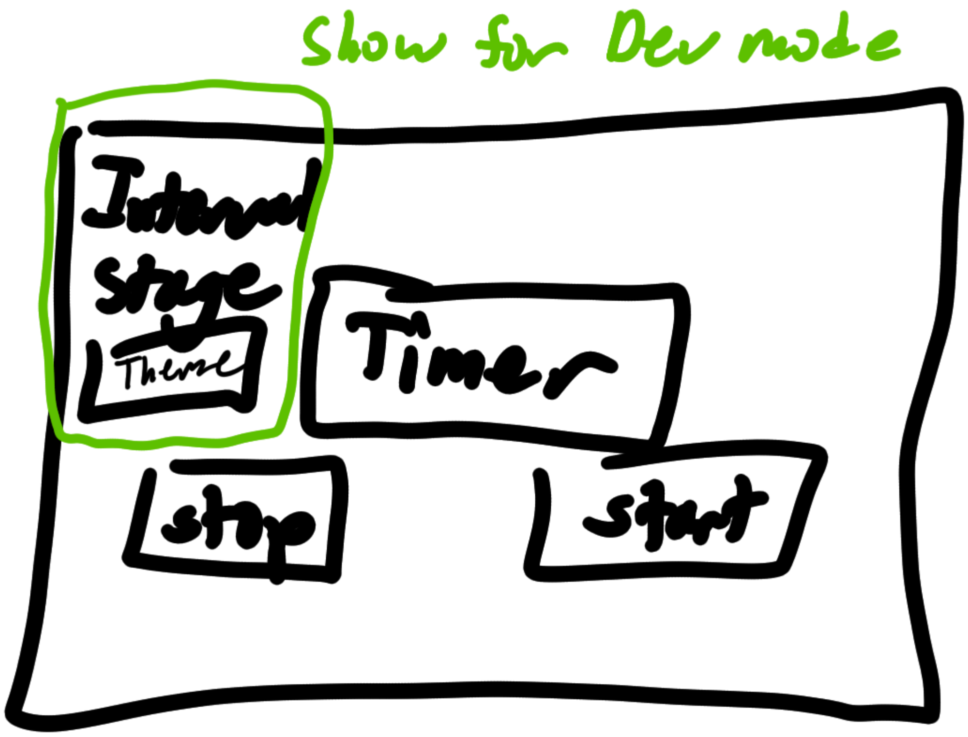


Figure 3: Fat-Marker Sketch for Basic UI Layout

The light and dark themes would be toggled by a button that calls a function to change colors. We are going to keep it as a button because when we add additional features we would like to be time efficient by reusing code. These two themes should be the most accessible for the general public, such as allowing colorblind people to distinguish between black and white on both minimalist themes.

Phase 2: Additional Features (W7-~9)

Ideally, the MVP timer will be finished by the end of the sixth week. We have brainstormed several basic settings to include as well as making the timer more aesthetic to attract users. Our timer is called Potatimer because we plan to create a potato theme in addition to the minimalistic ones. With the MVP we built, these features will be added onto what we already have so that we won't affect the correctness of the MVP. For example, to keep the minimalistic theme of the timer we will add a settings icon that opens a menu box with extra features such as changing theme, adjusting default times and volume, and more. Some of these features, such as tasks, create dependencies on the basic functions of our timer [See **Appendix A**]. All of these features are covered in the second pitch, which can be found in the same repository as this one.

Phase 3: Finalize Timer (W~9-10)

By week eight or nine, we will have to make sure the timer is presentable without any bugs. Due to the new features we would have implemented, we have to run QA tests on our timer in order to validate its functionality. It would be ideal for us to finalize the timer by the end of week nine so that we can all have breathing room to study for all of our finals. We don't know what is expected of us this late in the quarter besides a MVP timer, so we shall plan accordingly as we get more assignments about the project.

Potential Problems

We anticipate that there will be issues coordinating the JavaScript scripts with the website's HTML and CSS. These issues would be debugged and fixed at the highest priority because it should be a recurrent issue that would be solvable once we understand the frameworks over time. For example, when the timer reaches zero it has to update the timer display, play a sound, and update the interval and state. There are many attributes that have to be adjusted in the HTML, so we have to check each one to see if the program did what it was supposed to.

What We're Not Doing

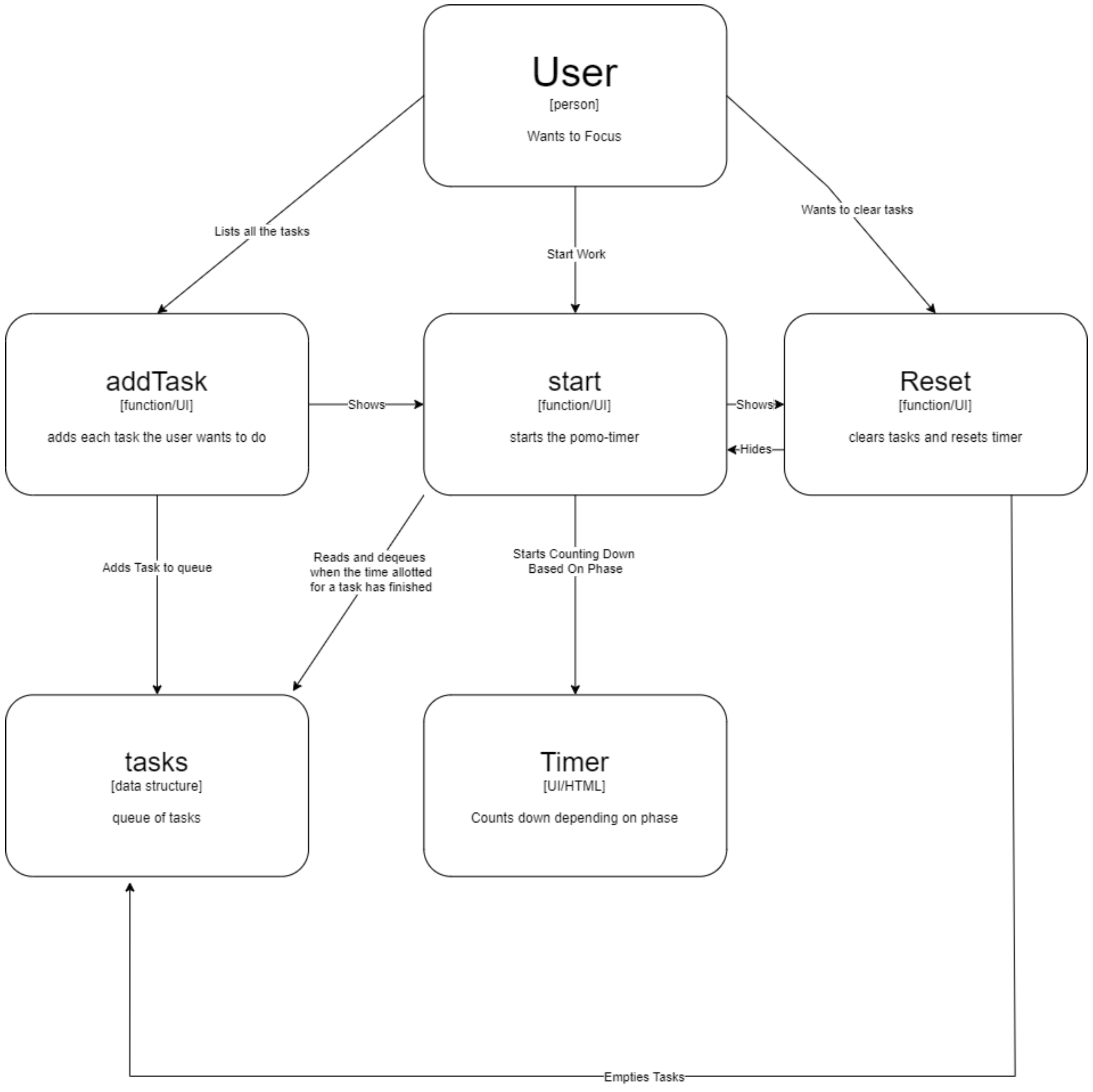
Given the time constraints for the end of the quarter, we will not be able to accommodate a larger audience and many accessibility edge cases. Because we are focusing on a browser-first approach to the frontend, we won't be making the website responsive. The timer targets a browser audience, so mobile users will be put at a disadvantage and may not be able to use our timer.

For more engaged audiences such as children that need external motivation to use our timer, we won't make the potato-themed timer like a collectible game. We are storing information on local servers through GitHub Pages, so it isn't possible to store client information. Therefore, we won't create account-based logins that count the number of potato harvests for a user.

Nevertheless, the main goal for the start of this project is to make sure that we achieve a fully-functional pomodoro timer. We will avoid using third party tools to ensure that we do not overwhelm ourselves with unfamiliar environments. Therefore, we will currently restrict ourselves to using HTML, CSS, and JavaScript for our web app.

Appendix

[Appendix A] C4 Diagram of the MVP


User
[person]
Wants to Focus
Lists all the tasks
Start Work
Wants to clear tasks
addTask
[function/UI]
adds each task the user wants to do
Shows
start
[function/UI]
starts the pomo-timer
Shows
Hides
Reset
[function/UI]
clears tasks and resets timer
Adds Task to queue
Reads and deqeues when the time allotted for a task has finished
Starts Counting Down Based On Phase
tasks
[data structure]
queue of tasks
Timer
[UI/HTML]
Counts down depending on phase
Empties Tasks