

# 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 Zou — Team Lead/Manager

\*Update pitch as more diagrams, layouts, and references are created

## **Table of Contents**

<b>Introduction</b>	<b>4</b>
<b>Phase 1: Basic Timer (W4-6)</b>	<b>5</b>
<b>The Timer Problem</b>	<b>5</b>
<b>Time Allocation</b>	<b>6</b>
<b>Proposed Solution</b>	<b>7</b>
<b>Phase 2: Additional Features (W6-~9)</b>	<b>9</b>
<b>Planned Features</b>	<b>9</b>
<b>Time Allocation</b>	<b>11</b>
<b>Proposed Solutions</b>	<b>11</b>
<b>Phase 3: Finalize Timer (W~9-10)</b>	<b>13</b>
<b>Potential Problems</b>	<b>13</b>
<b>What We're Not Doing</b>	<b>13</b>
<b>Appendix</b>	<b>14</b>

## 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; the theme of our timer as minimalist light, dark, and potato mode; and various options to mute sound or reset the cycle.

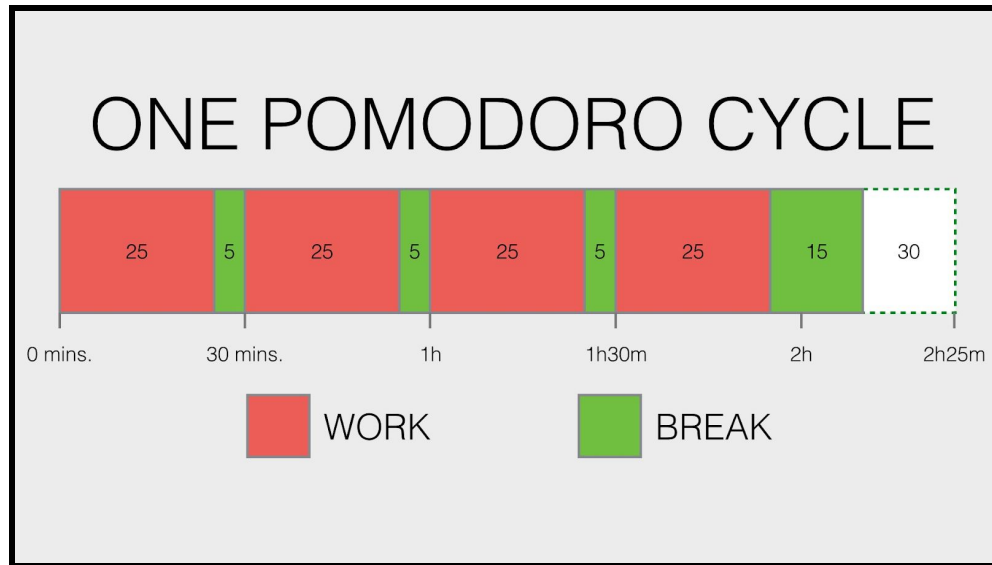
In order to meet this goal, we will create the timer in three phases that partition the remainder of the quarter. Phase 1 is about building a fully-functional MVP timer through a focus on planning, research, and backend work. We expect to have this product complete by the end of the sixth week. Phase 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.

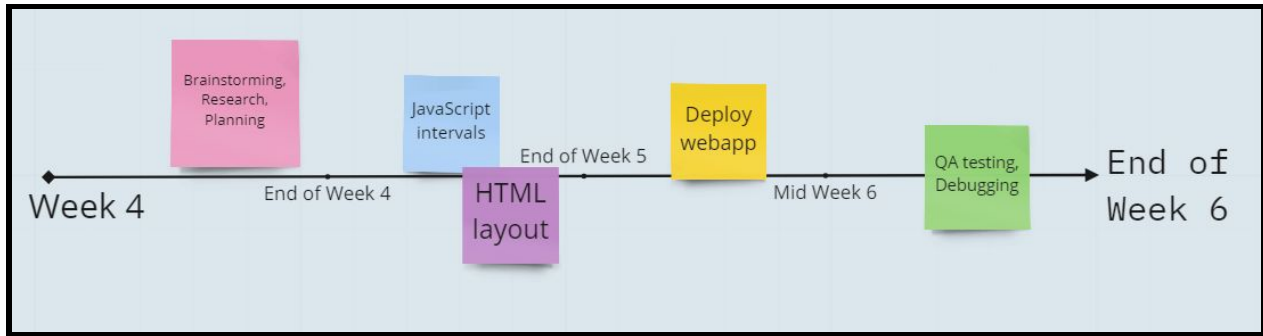
### User Stories

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.

Rachel is a twenty-year old computer engineering undergraduate at MIT. She's taking six classes this semester, so all the readings and homework are exhausting her ability to complete her academics and sleep for a proper time. Sometimes she would take naps from trying to pull all-nighters to catch up on homework, so Rachel is looking for a way to study more efficiently. A week ago inside the Starbucks at the corner of Ames St., Rachel overheard several peers talk about a web app called Potatimer. Instantly, she opened a new tab on her Macbook and visited the Potatimer app. Rachel was drawn to the simple design of the Potatimer that she felt all the white space around the timer was calming in contrast to the dozens of pages she had to read that week. She read the instructions and instantly knew how to operate the timer and its purpose in her life. With a single click, Rachel started the accessible timer and reluctantly went back to studying. She struggled to think of how to design an algorithm to solve her graph problem, but then she heard a beeping noise coming from the Potatimer. It told her to take a short five-minute break from studying. Rachel agreed with the suggestion and started to order a latte. In that short break, Rachel felt that she could understand the solution by the way the customer line was a queue and the person at the cashier popped off the queue! The idea was fresh in her mind and she was able to order her drink, so she went back to her laptop and quickly wrote up a sweet solution to her graph problem. Unbeknownst to Rachel, the timer would turn her life around with the order it brought to her overwhelmed life of studying. A month later, she was able to fix her sleep schedule and have time to hang out with her friends. With the help of Potatimer, Rachel was able to pass all six of her classes and she would continue using the timer throughout her college experience, whether it be in a cafe, on the go, or even in a remote setting.

## **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<sup>1</sup> 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.

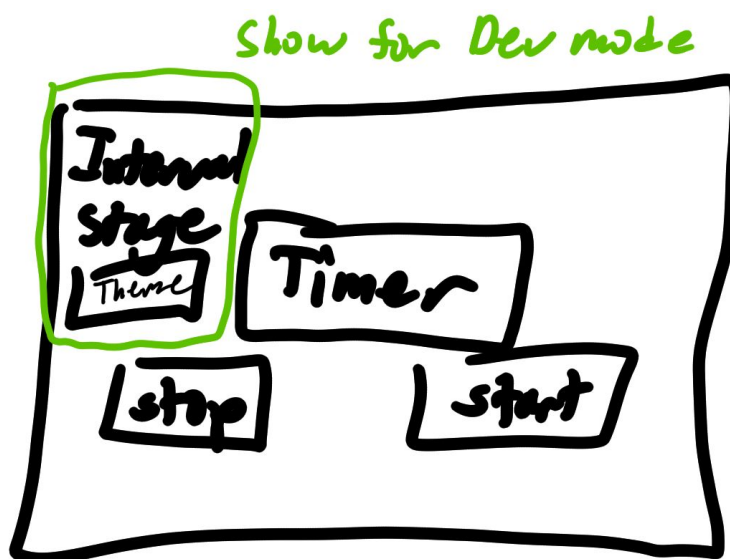
The [website application for our timer](#) would consist of a button for the start and stop buttons with a paragraph element identified by an id as the timer. By marking what element is the timer, we specify what element is being modified without accidentally modifying all elements of that type. When 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

---

<sup>1</sup> [These overarching decisions can be found in specs.](#)

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<sup>2</sup> 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.



**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.

---

<sup>2</sup> Ibid

## Phase 2: Additional Features (W6-~9)

Many of these features are things we would like to do after we create an MVP timer by the end of the sixth week, thus these ideas aren't concrete. We are still brainstorming more ideas, so this pitch will be updated throughout the quarter.

### Planned Features

Our timer is called potato timer for a reason. Therefore we will create a third theme that is about growing potatoes as the user studies. The point of this theme is to make it a satisfying experience for users to stay motivated by watching potatoes grow throughout one pomodoro cycle, so it reinforces a user's long-term focus. The default pomodoro cycle is two and a half hours long, so growing and harvesting potatoes would be a practical way of modeling the amount of time a user invested in working on their assignments!

As seen from our simple system architecture [See Appendix A], an intermediate function we would like to implement are tasks. This gives our timer semantic meaning by reminding the user what they have to do during their session. When they input a task, it is displayed on the website until they remove it.

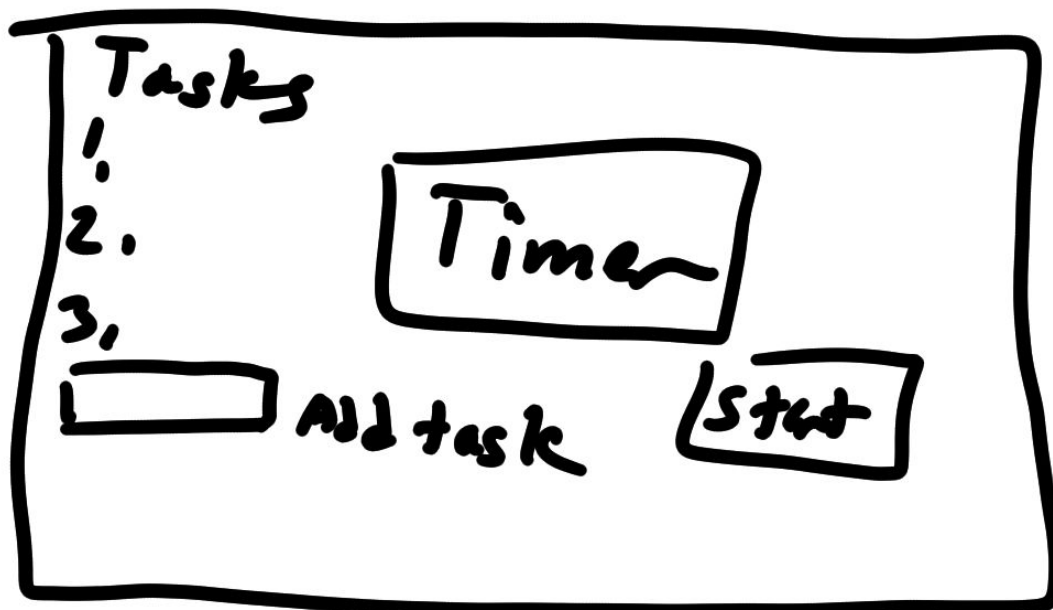
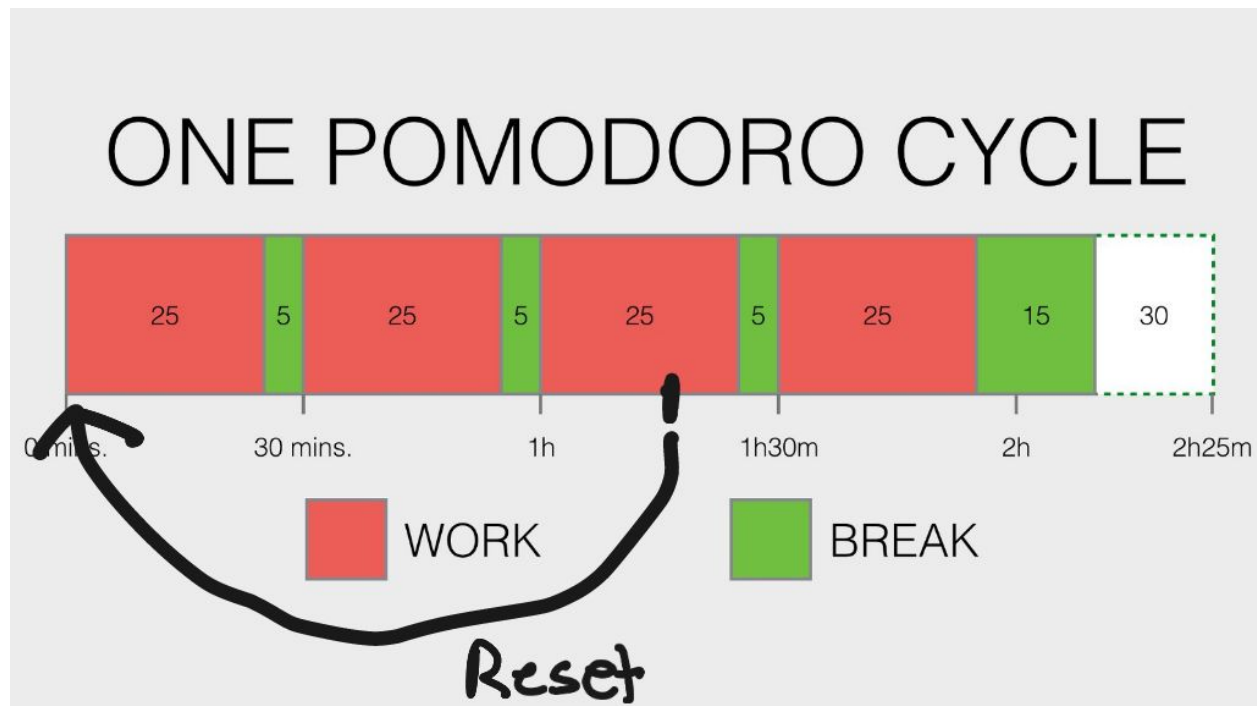


Figure 4: Sketch of Task List



We expect that users won't stay glued to the potato timer website when they study or do their work. If they leave the timer running in a tab to do other work, the user should know without navigating back to the timer tab how much time they have left. We want to make the timer convenient for our users, so displaying the countdown timer in the tab would be a great feature to have.

Say a user wants to reset the progress for whatever reason. Instead of refreshing the page and losing all of their settings, they should be able to reset the timer cycle instead. A reset button that can be displayed from the settings menu would accomplish this user's need.



**Figure 5: Reset Button Behavior**

## Time Allocation

Due to the various features we have brainstormed, we will be implementing them in terms of time and usefulness priority. The backend and frontend developers would have standup meetings through Zoom and coordinate things together whenever it is convenient.

- Settings menu: Will adjust the layout of the website to fit this, must coordinate with all elements inside the menu; long and maybe problematic.
  - Adjust the volume: simple JS function with an HTML property; a few minutes.
    - Muting the volume is the same as adjusting volume to zero.
  - Potato theme: Heavy HTML and JavaScript changes in background; long.
- 
- Have the countdown in tab title: modify timer countdown function to change the <title> data; very short.
  - Tasks: Add and remove HTML; short-medium.
  - Reset button: Resets the entire cycle; a few minutes.

## Proposed Solutions

These are additional features, so we should be adding these features onto our existing code. This would reduce the amount of risk involved in restructuring things that already worked in Phase 1 QA testing. For example, we can reuse the HTML and CSS layout of the light theme to be a base for the potato theme if we implement it. In addition, user-modifiable times for work, break, and long break periods can be achieved by replacing the hard-coded numbers with a variable that holds the default. When the user changes the default number in the entry field, it would change the number assigned to the variable without changing the rest of the functions.

We anticipate that there will be several features that will be added, so the best way to display all of these would be in a menu popup. These settings can be indicated by a gear icon button or a text button. When this button is clicked, it would show all of the semantically-named settings and clicking on them will trigger the corresponding feature. The menu would include the themes, the user-modifiable times, and the volume to name a few planned features. To close the menu, a user would click the familiar X at the corner of the menu. We could also use an event listener where clicking out of the menu box closes it. A sketch of this menu is shown in [**See Appendix B**].

From the Week 5 lab, we should be able to create an adjustable volume button for the timer, so the developers should be following a similar strategy to adjust the sound volumes.

The potato theme is the main goal of the potato timer, so we are going to design several layouts and animations to simulate potatoes growing on the application. An early potato theme was proposed around week 3 and gives us an idea of how to create a potato theme with animations [**See Appendix C**]. We have yet to decide a consensus vote if we want to go through with a potato theme.

The minor features should be simple to implement. For displaying the countdown in the title of the page, we would adapt the timer countdown function to also modify the data in the <title> element. The <title> element is what is displayed in the tabs of websites, so modifying the element would achieve our desired goal. To implement a reset button, we would create a new visible button to do this and then create a new function that stops and resets the intervals and phase to their default values.

## **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**

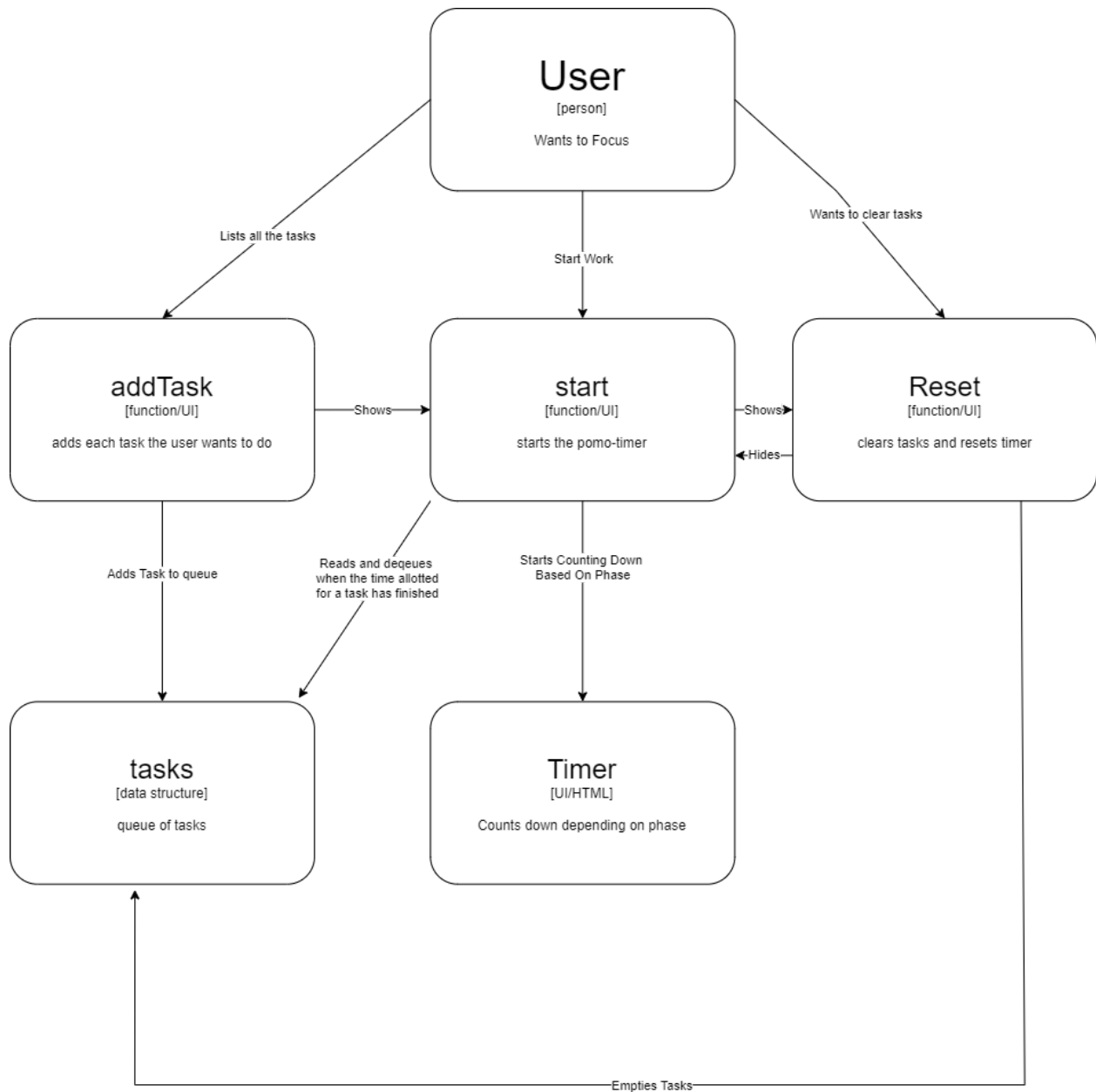
Given the time constraints for the end of the quarter, we will not be able to accommodate a larger audience and many accessibility edge cases. Because we are focusing on a browser-first approach to the frontend, we won't be making the website responsive. The timer targets a browser audience, so mobile 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 deploying GitHub Pages, so it isn't possible to save client information such as customized times if they close the page. Therefore, we won't create account-based logins that count the number of potato harvests for a user.

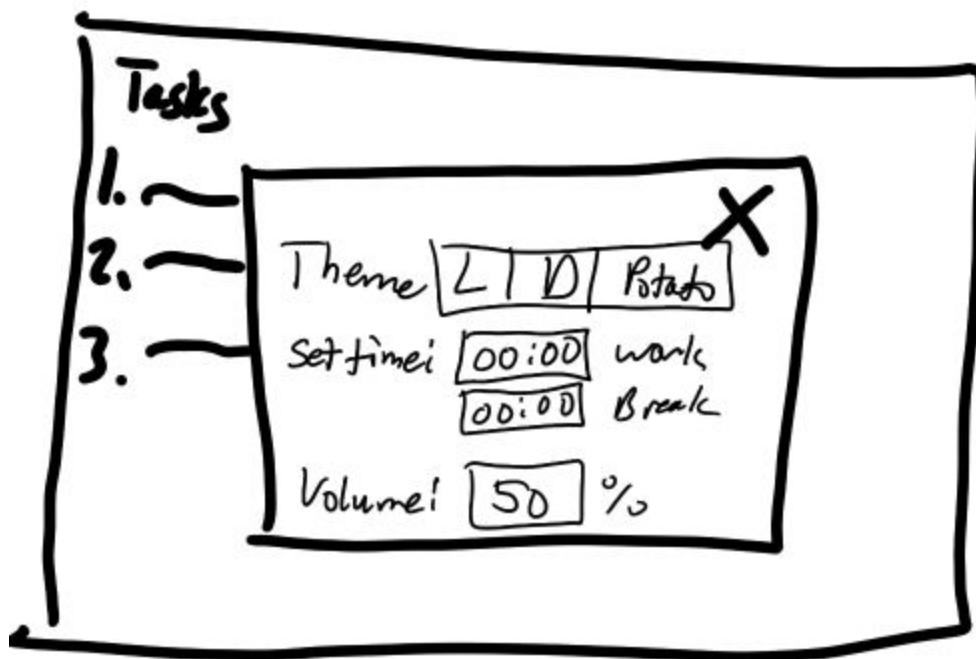
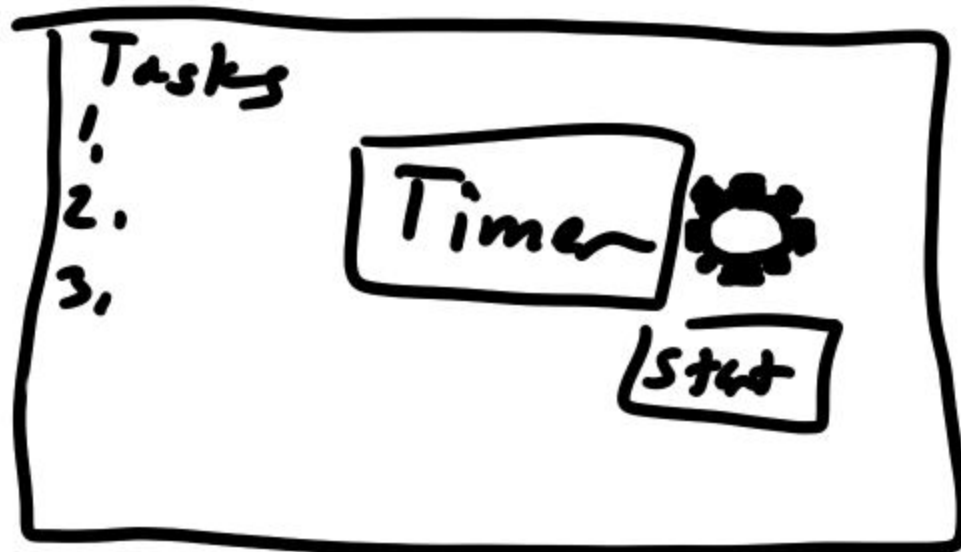
Nevertheless, the main goal for the start of this project is to make sure that we achieve a fully-functional pomodoro timer. We will avoid using third party tools to ensure that we do not overwhelm ourselves with unfamiliar environments. Therefore, we will currently restrict ourselves to using HTML,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

## Appendix

### [Appendix A] Potatimer's C4-Modeled System Architecture

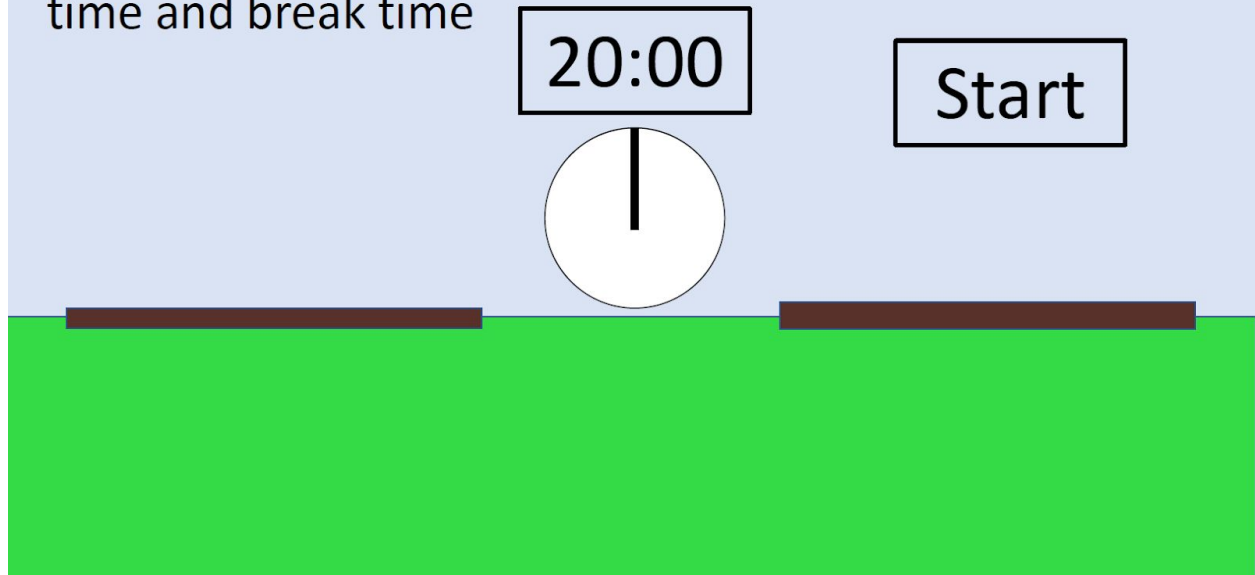


[Appendix B] Sketch of Settings Icon and Menu Popup



### [Appendix C] An Early Layout for the Potato Theme

Page load: no plants, start button, options to change time and break time



Break stage potato display (5:00 on timer, 0/2)

