Project Evaluation Phase 2

CSE-112-team8: Cre8tors

Table of Contents:

Overview

The Evaluation Process

- Create a roadmap of what goals we want to push for. Compare this to the current status of the repositories.
- look into the current UI and see which system has a more flexible base design in that which can you envision having a better UI.
- Rip up the Code bases and find all the possible cons associated with the code.
- Set up the repos locally and make changes to the code to figure out how brittle the codebase is.
- Get code coverage and test results of each repo if they exist.
- List the pros and cons of the repo from the perspective of the values we possess (Simplicity, Organization, User Centered Design, and Transparency).

Roadmap (Priority list)

1. Implement a more in-depth testing suite so that bugs can be found effectively.
    a. Higher code coverage for unit testing.
    b. More considerations and edge cases for E2E testing.
    c. Codeclimate
    d. Set up User Reviews from 3rd party people (other teams with pomodoro timer projects/ Other team in general/ people in general)
2. Refactor the Code (JS/HTML/CSS)
    a. Create more semantic HTML
    b. “Untangle” CSS
    c. Create comprehensive Javascript
3. Allow phone screen accessibility
4. Move deployment from Github pages to a new host.
    a. Will need to update pipeline by adding in a continuous deployment which supports the new host.
5. Add accessibility features such as keybinds, colorblind accessibility(low priority) and internationalization (multiple languages, prioritize just supporting non-English characters).
6. () Set up a back-end to give continuity across machines for users in terms of stats, tasks, and even possibly sessions.
7. Set up the website to be a PWA or a chrome extension.
8. Enhancing task manager to stand out as an app (i.e. task templates, being streamlined, what have you done)

Documentation Comparison

Chad's Repo

- Onboarding Documents: Sort-of
  - Documentation on code structure exist (C4 Diagrams), and documents for getting started on the pipeline and testing but not so much on getting the project set up locally.
- ADRs: 9
- C4 Diagram: Yes
- Issues: 3 open, 29 closed
- SCRUM Board: Yes
- Flow Charts: Yes
- User Stories: 4
- Personas: 8
- Wireframe/High Fidelity: Yes
- Roadmap: Yes

James' Repo

- Onboarding Documents: Mostly No
  - In the wiki there are rough function documentation and roughly formatted build pipeline documentation
- ADRs: 3
- C4 Diagram: No
- Issues: 2 open, 25 closed
- SCRUM Board: No
- Flow Charts: Yes
- User Stories: 8
- Personas: 8
- Wireframe/High Fidelity: Yes
- Roadmap: No

Dependencies and Tools Comparison

Chad's Repo

- Dependencies
  - Bootstrap
- Tools
  - Testing
    - Cypress
    - Jest
  - Git Commits
    - Husky
  - Documentation
    - JSDoc
  - Linting
    - ESLint
    - Prettier

James' Repo

- Dependencies
  - N/A
- Tools
  - Testing
    - Jest
  - Documentation
    - JSDoc
  - Linting
    - ESLint
    - Prettify

Usability Comparisons

Chad's Repo

- Multi Page Website with three separate pages.
  - Would be more suited to creating a backend/login system for people to maintain progress between sessions
- Instructions included.
  - Depending on how we set it up, it might need popups or tutorial prompts to teach people how to access the timer
- Functionality hidden in different pages
- Non responsive design.
- Un-intuitive main menu, does not signal its purpose.

James' Repo

- Single page, everything is on the one page
  - Would be more suited to a very simple timer for people who are easily distracted, don't want more overhead.
- Non responsive design
- Settings menu doesn't display options properly
- Unused Stop button on timer
- No instructions
  - Also, can easily be designed such that no instructions are needed
- Easy to see all functionality
- Easy to navigate

Codebase Scrutiny

Chad’s Codebase (HTML)

- Very strong div-itis going on with the modals system. Large dependency on id and classes and a lack of semantic HTML.
- Modals could have been generalized web components or even templates which change based on their current role instead of just having three div modals constantly in the DOM.
- The structure of the HTML is generally unorganized in a semantic sense. Header, Main, and Footer tags aren't used even though those structures are used.
- In General, the HTML is more focused on catering to the style and the JS instead of setting up the content.
- There are many instances of content in the HTML existing only to set up some styling, which contradicts the purpose of HTML, which is to act as the provider of content.

Chad’s Codebase (CSS)

- CSS file main.css is only used in a single HTML file (index.html) which seems like a poor decision as the whole purpose of a CSS file is that it can be cached utilized in multiple files to cut down on costs. Since it is just in one file it simply increases the overhead. This issue of a CSS file being used in only one file persists for all the CSS files.
- A good chunk of "!important"s thrown around, which entails there is a sizeable amount of overlap in the CSS which implies the styling is rather tangled.
- A majority of the styles involving measurements use pixels, which is pretty bad. This is because only absolute values are being used, meaning there is little leniency towards varying screen sizes. It also shows that only the generic screen size of a monitor was considered (1920x1080).
- There is no consideration for different screen sizes and media queries are never used.
- A majority of the CSS rules are catered towards a specific class or id, which means any overlap in classes would cause a conflict (possibly why the !importants were everywhere). It also means that there could be a large amount of redundancy in code as id styling could be overlapping and could be caught in the same tag, but such nuance is lost when setting rules mostly for ids.
- The styling for the timer page and the modals in the page are separated into different pages which is a bit confusing.

Chad’s Codebase (Javascript)

- window.onload is used instead of DOMContentLoaded which could cause bottlenecks in loading to echo through the javascript.
- For the eventCloseModal, there is a ton of redundancy in the switch case, as we are doing a ton of get elements that we don't need to, which may cause an unnecessary amount of lag.
- In the main js, the session date system seems brittle because the time, lastVisist, getting the session date are hingent on loads and unloads going through smoothly.
- Creating the progress bar is really brittle.
- In general, the task item is rather brittle because some attributes are check and maintained through the class value instead of some internal system while other attributes are maintained through their own attributes.
- In the stats section, the unload handler seems to set the stats list to the local storage, but this is redundant.
- Many values are hard-coded in, especially in the timer javascript. This means that any attempt at adding modularity will be harder to implement down the road if we plan to do that.
- Fair amount of magic numbers, at least in the timer js which could cause issues down the road. For example, the time of a break is 5 minutes, but the value which sets the time html, and the value which is given to starting the timer, can be made into two seperate values.
- Distractions are only logged in local storage if the session is completed.
- There are many instances of the same elements being grabbed constantly in functions even though they could be grabbed once at the start as global variables.

James’ Repo HTML/CSS

- HTML is div-soup, tons of in-line CSS despite linking two CSS sheets
- CSS files are very long and not well commented.
- Viewing in DevTools, half of the CSS is overwritten and redundant.
- UI-wise we’d likely start from scratch with the CSS with either project, but this repo will be more awkward to work with due to the HTML being div-soup

James’ Repo JS

- **Lack of comments**, JSDoc style comments are not thorough (no @params or returns, descriptions are too vague)
- Lots of the function/variable names are confusing (i.e.`*minuteChange*` changes how long the timer lasts and not the actual time during countdown, `*mixBut*` is the start/stop button, onBreak refers to if the timer is currently supposed to be paused). It’s about 50/50
- **Not well decomposed**. There is no main.js or .js that’s clearly the ‘runtime’ file- the running of the timer is based in the **settings.js**
- **Strange/unnecessary inheritances.**
    - e.g. Starting the timer is defined in settings.js, which relies on whether a task is selected, which is determined by a function defined in stats.js, which is based on whether the innerHTML matches exactly with the string “`Current Task: None`”.
    - Also, *timeAdvance* and many other functions are **exported despite not being called elsewhere**.
    - Also some **circular dependencies** which don’t seem to make sense (stats.js and settings.js inherit from each other.)
- Currently uses **setInterval** and calls a function that updates the countdown circle and time for main timer functionality. Not sure if this is the best, or how modular this would be.
- Lots of weird bugs.
    - For example per Lars, if you change the settings, the timer’s length changes in real time as you edit, but the changes will either be saved or not saved depending on how you close out the setting modal (clicking **save OR the X** resets the timer to the new time, clicking **off** the modal discards changes).

Codebase Scrutiny Conclusions

Chad’s Repo

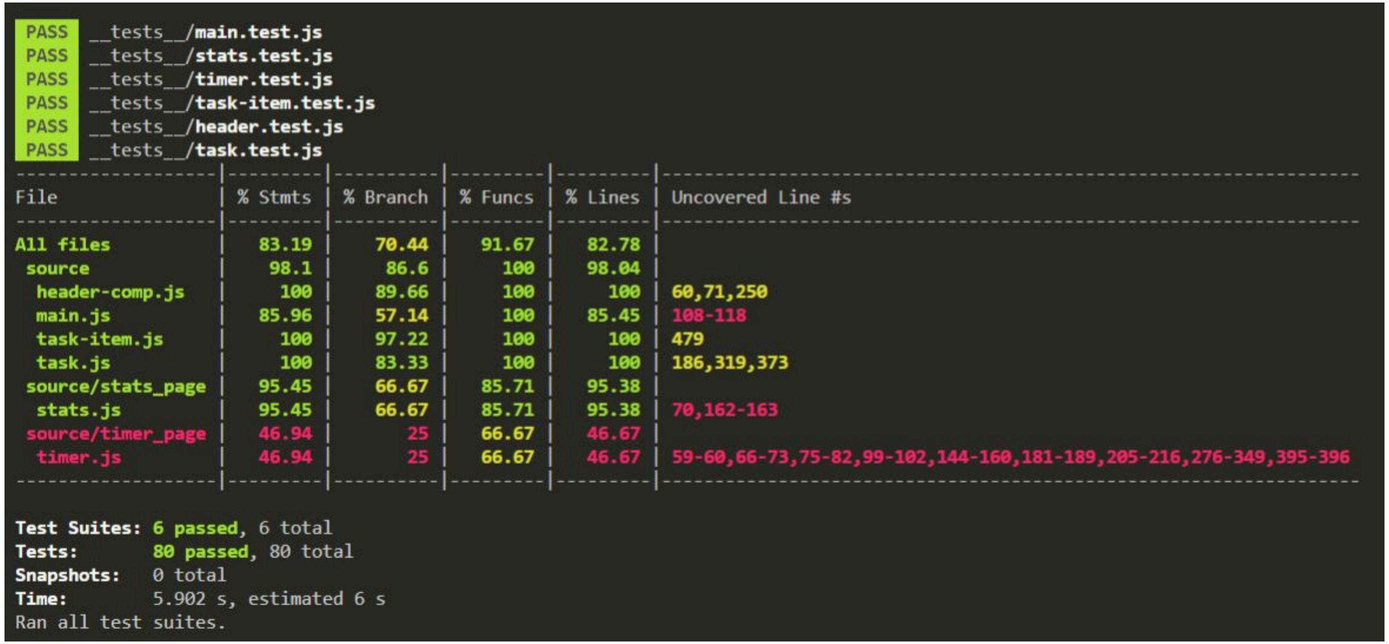
- The code base is rather dependent on Bootstrap which could be a concern for the future if we decide to opt-out of such a dependency.
- There is little to no semantic HTML used which means navigating the HTML will be rather difficult due to most of the elements being divs.
- The CSS is very messy and tangled as most rules are being enforced via class names and ids. Additionally, there are a ton of rule conflicts occuring so the CSS rules are tangling with one another.

James’ Repo

- Problems with HTML and CSS as well
- Strange inheritances, not well decomposed
- Lack of quality documentation or comments

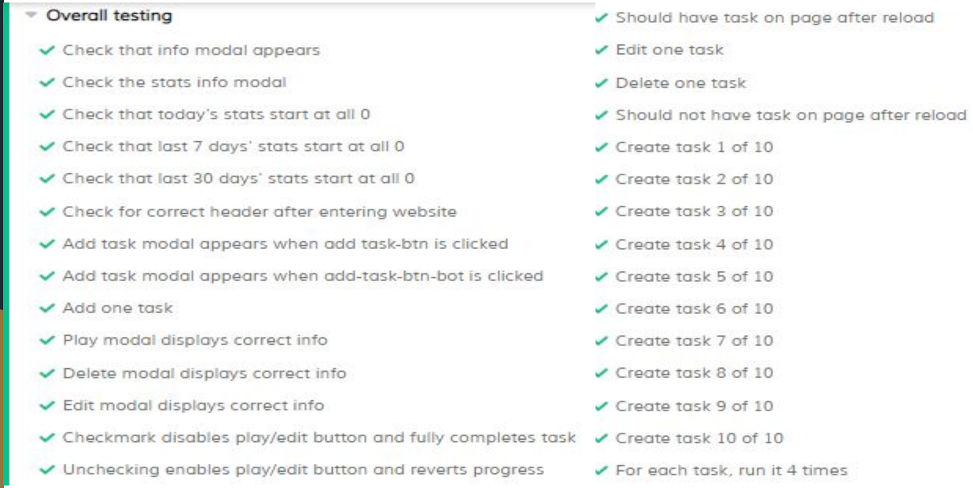
Testing

Chad’s Repo’s Jest Code Coverage



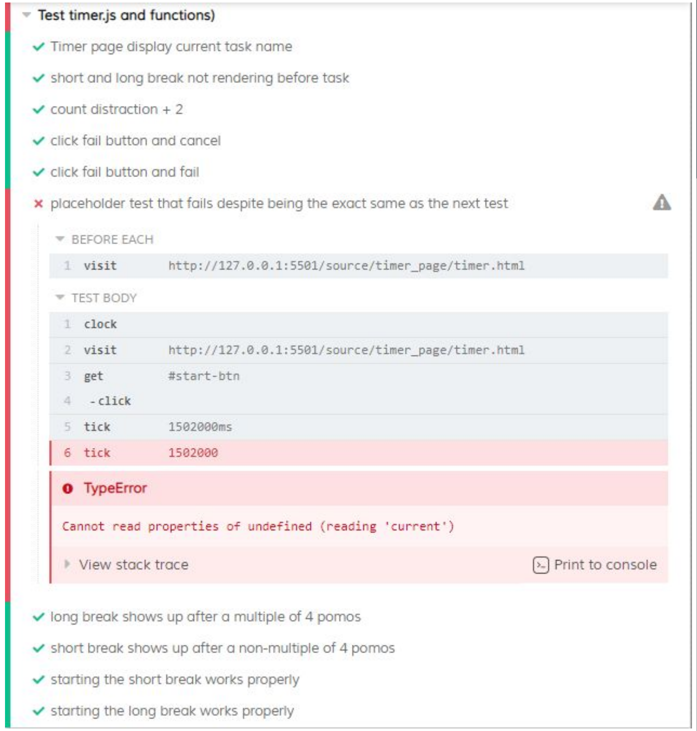
```
PASS  __tests__/main.test.js
PASS  __tests__/stats.test.js
PASS  __tests__/timer.test.js
PASS  __tests__/task-item.test.js
PASS  __tests__/header.test.js
PASS  __tests__/task.test.js
-------------------|---------|----------|---------|---------|-------------------------------------------------------------------
File               | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-------------------|---------|----------|---------|---------|-------------------------------------------------------------------
All files          |   83.19 |    70.44 |   91.67 |   82.78 |
 source            |    98.1 |     86.6 |     100 |   98.04 |
  header-comp.js   |     100 |    89.66 |     100 |     100 | 60,71,250
  main.js          |   85.96 |    57.14 |     100 |   85.45 | 108-118
  task-item.js     |     100 |    97.22 |     100 |     100 | 479
  task.js          |     100 |    83.33 |     100 |     100 | 186,319,373
 source/stats_page |   95.45 |    66.67 |   85.71 |   95.38 |
  stats.js         |   95.45 |    66.67 |   85.71 |   95.38 | 70,162-163
 source/timer_page |   46.94 |       25 |   66.67 |   46.67 |
  timer.js         |   46.94 |       25 |   66.67 |   46.67 | 59-60,66-73,75-82,99-102,144-160,181-189,205-216,276-349,395-396
-------------------|---------|----------|---------|---------|-------------------------------------------------------------------

Test Suites: 6 passed, 6 total
Tests:       80 passed, 80 total
Snapshots:   0 total
Time:        5.902 s, estimated 6 s
Ran all test suites.
```

Chad’s E2E Testing (Cypress)



Overall testing

- ✓ Check that info modal appears
- ✓ Check the stats info modal
- ✓ Check that today’s stats start at all 0
- ✓ Check that last 7 days’ stats start at all 0
- ✓ Check that last 30 days’ stats start at all 0
- ✓ Check for correct header after entering website
- ✓ Add task modal appears when add task-btn is clicked
- ✓ Add task modal appears when add-task-btn-bot is clicked
- ✓ Add one task
- ✓ Play modal displays correct info
- ✓ Delete modal displays correct info
- ✓ Edit modal displays correct info
- ✓ Checkmark disables play/edit button and fully completes task
- ✓ Unchecking enables play/edit button and reverts progress
- ✓ Should have task on page after reload
- ✓ Edit one task
- ✓ Delete one task
- ✓ Should not have task on page after reload
- ✓ Create task 1 of 10
- ✓ Create task 2 of 10
- ✓ Create task 3 of 10
- ✓ Create task 4 of 10
- ✓ Create task 5 of 10
- ✓ Create task 6 of 10
- ✓ Create task 7 of 10
- ✓ Create task 8 of 10
- ✓ Create task 9 of 10
- ✓ Create task 10 of 10
- ✓ For each task, run it 4 times



Test timer.js and functions)

- ✓ Timer page display current task name
- ✓ short and long break not rendering before task
- ✓ count distraction + 2
- ✓ click fail button and cancel
- ✓ click fail button and fail
- ✗ placeholder test that fails despite being the exact same as the next test

BEFORE EACH

```
1  visit      http://127.0.0.1:5501/source/timer_page/timer.html
```

TEST BODY

```
1  clock
2  visit      http://127.0.0.1:5501/source/timer_page/timer.html
3  get        #start-btn
4   -click
5  tick       1502000ms
6  tick       1502000
```

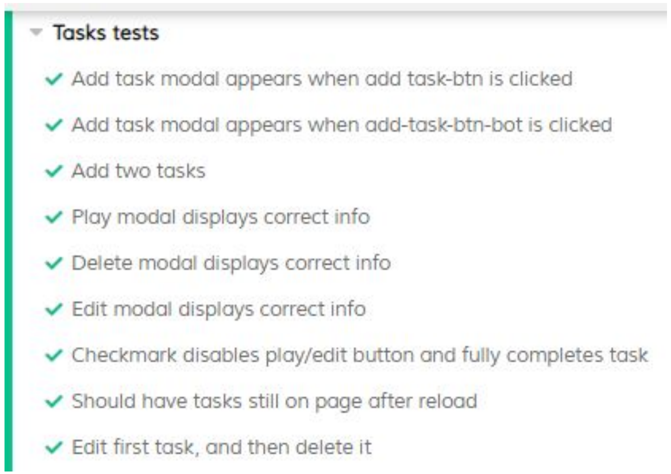
TypeError

```
Cannot read properties of undefined (reading 'current')
```

View stack trace   Print to console

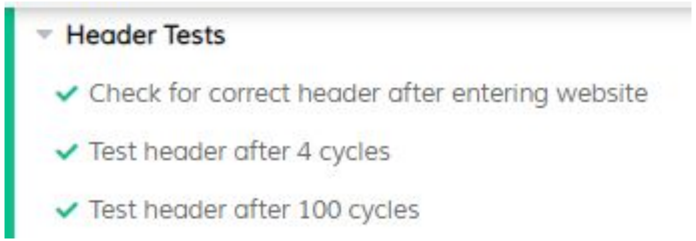
- ✓ long break shows up after a multiple of 4 pomos
- ✓ short break shows up after a non-multiple of 4 pomos
- ✓ starting the short break works properly
- ✓ starting the long break works properly

Chad’s E2E Testing (Cypress) (Cont.)



Tasks tests

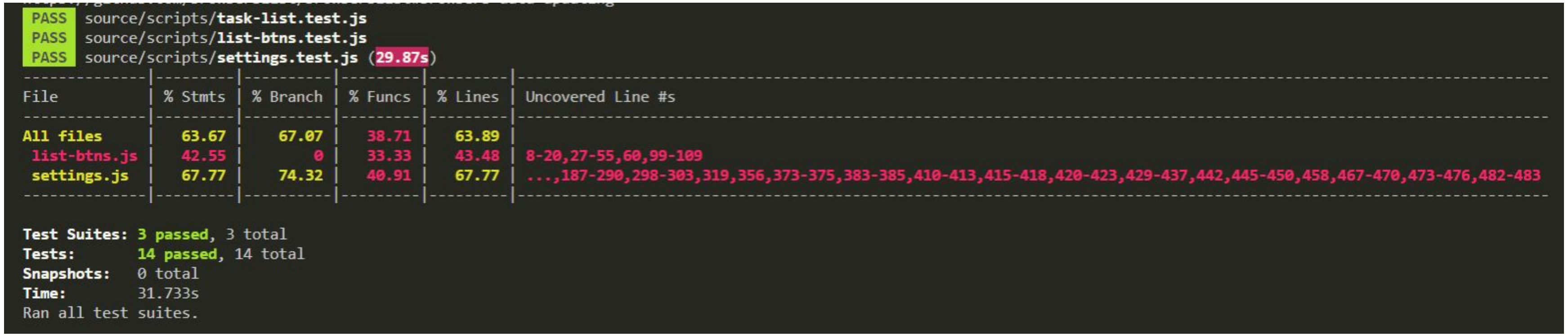
- ✓ Add task modal appears when add task-btn is clicked
- ✓ Add task modal appears when add-task-btn-bot is clicked
- ✓ Add two tasks
- ✓ Play modal displays correct info
- ✓ Delete modal displays correct info
- ✓ Edit modal displays correct info
- ✓ Checkmark disables play/edit button and fully completes task
- ✓ Should have tasks still on page after reload
- ✓ Edit first task, and then delete it



Header Tests

- ✓ Check for correct header after entering website
- ✓ Test header after 4 cycles
- ✓ Test header after 100 cycles

James’ Repo’s Jest Code Coverage



```
 PASS  source/scripts/task-list.test.js
 PASS  source/scripts/list-btns.test.js
 PASS  source/scripts/settings.test.js (29.87s)
--------------|---------|----------|---------|---------|-------------------------------------------------------------------------------------------------------
File          | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
--------------|---------|----------|---------|---------|-------------------------------------------------------------------------------------------------------
All files     |   63.67 |    67.07 |   38.71 |   63.89 |
 list-btns.js |   42.55 |        0 |   33.33 |   43.48 | 8-20,27-55,60,99-109
 settings.js  |   67.77 |    74.32 |   40.91 |   67.77 | ...,187-290,298-303,319,356,373-375,383-385,410-413,415-418,420-423,429-437,442,445-450,458,467-470,473-476,482-483
--------------|---------|----------|---------|---------|-------------------------------------------------------------------------------------------------------

Test Suites: 3 passed, 3 total
Tests:       14 passed, 14 total
Snapshots:   0 total
Time:        31.733s
Ran all test suites.
```

- Note: only list-btns.js and settings.js were accounted for in the tests; there also exists task-list.js and stats.js

Testing Conclusions

Chad's Repo

- High Code Coverage for most javascript files when using Jest Unit Testing.
    - The only exception is the the timer javascript is not covered that well in the unit testing which is concerning since that is arguably the most complex system
- E2E Testing covers general use cases in all the systems
    - E2E cases don't cover edge cases and may not be as expansive as we would want it to be.

James' Repo

- Moderate code coverage in the Jest Unit tests, but there are many gaps in the code, especially with the settings.js file.
- The only form of testing is the Jest Unit tests, so we would have to implement E2E testing from the ground up with this project.

Code Modification Tests

Modifying Chad’s Repo

- Change the text
  - Easy process
- Change images
  - Harder than it looks for the task icons, as the icons of the task are apparently a font taken from google, which seems really odd.
  - Icons elsewhere are easy to change
- Change the colors
  - Works for some but not for others. Notably, it seems that bootstrap prevents some changes such as changing the background color.
- Change the default timer lengths
  - Not too hard, but changing the content of the html and changing the actual timer length were two separate processes.
- Add another start timer button
  - Only needed to copy the HTML of the button and then add a new event listener for the second button. Not too bad.
- Add another fail timer button
  - Same deal with the start timer
- Remove the timer visualization and see if the time still goes down
  - A ton of the JS is tangled with the fact that the timer element circle is there.
  - After ripping out JS involving the circle element from the JS, the timer seems to work fine.
- Remove Bootstrap.
  - Removing bootstrap from the index and stats menu, while hurting the style, does not remove any functionality. The hardest blow comes to the timer page, which not only looks ugly, but does not function.
  - Oddly enough, removing the Bootstrap JS seems to not break anything, it is specifically the bootstrap CSS which breaks things.
  - When removing the circle system from JS, removing the Bootstrap dependency seems to no longer break the system.

Modifying James’ Repo

- Let’s say I want to be able to pause, or change the effect of the start button (i.e. turn start into start/pause, as it currently just starts the timer then gets disabled). We could:
  - Use clearInterval and do a toggle when we click on the start/pause button
  - Create more eventListeners and add a toggle at timeAdvance
- I wouldn’t even try to modify the current CSS, I would suggest we redo that part from scratch due to all the entanglements. This should probably be one person’s entire job (i.e. one person does nothing but the CSS, maybe a little HTML this whole quarter).

- Hover mechanics are done by swapping out image
- For some reason, unable to change background color on local. Able to change colors of other things.
- Changing the default timer length is difficult because the default of 25 seems to be hard coded in a number of places. Figured it out eventually and it didn't seem to break anything.
- New start button doesn't do anything, doesn't break anything
- New Reset button doesn't do anything, doesn't break anything
- Removing timer visualization broke ability to add tasks/run timer

Code Modification Conclusions

Chad's Repo

- Changes to the repo were sometimes easy to implement and other times difficult to commit to
- Text changes, adding buttons, and the timer length changes were relatively easy to change
- Some changes like the timer wheel and the background color were based on Bootstrap, which meant changing them meant Bootstrap had to be messed with.
- Changing icons ranged from easy (stat icons) to near impossible without significant changes (task icons).

James' Repo

- Not a lot of code and no dependencies to learn (++simplicity)
- Simple layout would probably be easy to improve + customize (++simplicity)
- The code seems a bit messy. We would need to clean things up. Some weird inheritances and dependencies
- A fair number of bugs to fix

Overall Views and Conclusions

Overall Pros and Cons

Chad's Repo

- Pros
  - Test coverage for unit testing is generally high, with the only major exception being the timer javascript.
  - The documentation of the code base is generally fleshed out and accurate.
  - Onboarding documentation is generally thorough, at least for the system diagram, the pipeline, and testing. The only poorly documented section is documenting how a developer sets up the repo locally.
- Cons
  - The actual system is overly-complicated, rigid, and the dependency on Bootstrap is possibly a hindrance. HTML holds no semantic value and most info is pushed in the id and class attributes. The CSS trips over itself as most rules are based on the class and id. The javascript is hard to follow and is 'spaghetti'-esqe.
  - The timer section of the project is not fully explored. Tests hardly cover it and documentation implies that the timer was a trouble to get started to begin with, so working on it now might prove to be difficult to get into in terms of improvement.

James' Repo

- Pros
  - The structure is a lot more simple so adding things to it would possibly be more simple.
  - Less pages to worry about, only a single page to focus on.
  - No bootstrap dependencies needed to get started.
- Cons
  - Test coverage is rather low and there only seems to be unit testing, no form of E2E testing
  - Not much onboarding documentation. There are mentions of what exists in the project in terms of tools used, and even why these tools were used, but there is not much documentation on how to set them up.
  - Not as much documentation on code base, at least in regards to diagrams.
  - The structure of the front end is really messy, especially in the HTML, where everything is divs.
  - More bugs in the program, though they are shallow and shouldn't pose as a major problem.

Which Repo to Choose and why?

- Weighting Criteria:
  - Documentation Quality
  - Dependencies
  - Complexity (number of pages, ‘user overhead’/hardcoded steps/design for use)
  - Code quality/Structure
    - Brittleness/Hardcoding/Can we easily add features, a backend?
  - Test coverage (did not have time to look deeply at test quality)

Weighing the Two Repos

Criteria	Chad’s Repo	James’ Repo
Documentation Quality	More docs, has C4 diagrams, onboarding	Less documentation, does not have many diagrams, little onboarding
Dependencies	Uses Bootstrap	No dependencies
Complexity	Multi page application	One page, less code
Code Quality	Better code quality	Worse code quality
Test Coverage	Overall 83.19%	Overall 67.07% (Two js files are not accounted for)

Preliminary Decision

As of 3PM, 4/21/2022, we are moving forward with **Chad’s Repo**. Based on our initial criteria, Chad’s Repo is weighed more favorably than James’ Repo.

Notes:

- About half the team was missing from this meeting due to scheduling conflicts, so this is only a preliminary conclusion.
- We don’t have full agreement or consensus on the philosophy of the product (should it be incredibly simple, low overhead, or should it be more complicated?). Once we hash that out again, our decision may change.
- Criteria may change as a result of the next few meetings.