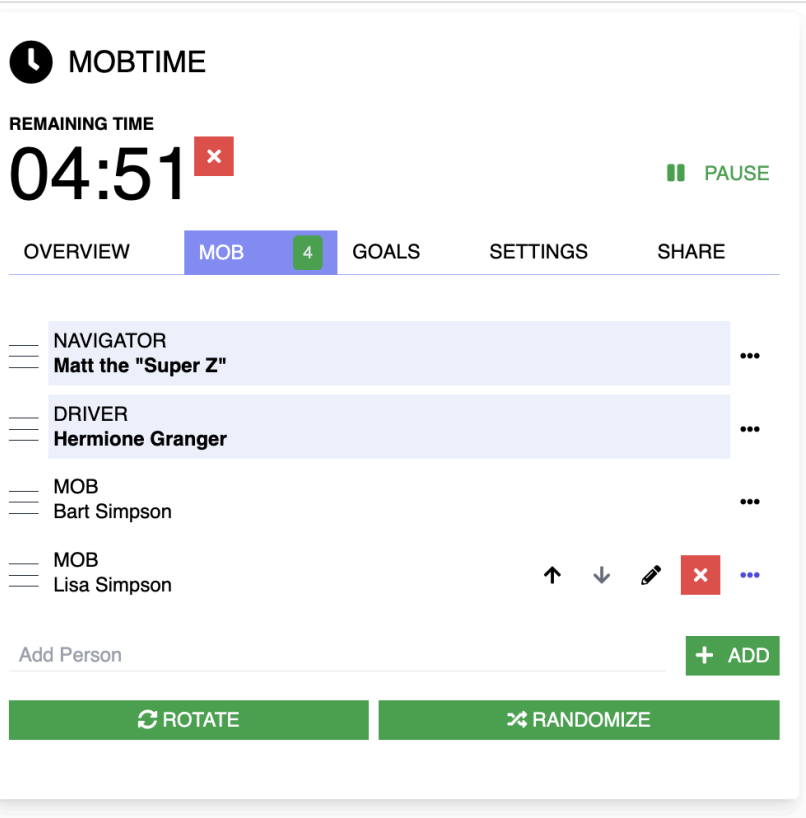
Basic Features

Goal: Create a mob timer for specifying participants and rotating roles for timed amounts. Inspired by https://mobti.me created by Github user mrozbarry.



Code Kata
Simplified Kata with Friendly Language
Mob Timer Kata
==========

Write a program to implement a mob programming timer.

The default duration for each turn is 5 minutes, and the default roles are Navigator and Driver.

The following is an example of a mob, with initial roles shown in parentheses:

Joe (Navigator)
Anita (Driver)
Carlos
Jennifer

After the timer has been started and the 5 minutes have elapsed, the mob order automatically rotates upward. For example, the above mob when rotated once looks like this:

Anita (Navigator)
Carlos (Driver)
Jennifer
Joe

When rotated again, it looks like this:

Carlos (Navigator)
Jennifer (Driver)
Joe
Anita

Pause / Restart

A user may pause and restart the timer as many times as desired until the timer has been running (not paused) for the full 5 minutes.

Cancel

The timer may be canceled, which resets the time back to 5 minutes but does not rotate the mob.

Manually Rotate

The mob may be rotated manually at any time. If the timer is running when this happens, then the time is reset back to 5 minutes so the next turn can have the full amount of time.

Modifying the Mob

The mob can be modified at any time, whether or not the timer is running. A person may be:
- Added - Note that when a new person is added to the mob, their default position is at the bottom of the list
- Removed
- Renamed
- Reordered (for example, moved from 3rd to 4th on the list of mobbers)

Randomize

The mob order can be randomized so that any person has an equal chance of being in any position on the list.

Modify Turn Length

Instead of each turn being fixed at 5 minutes, allow users to set a different amount of time.

Bonus Features
==========

Metrics

- Timer usage: Keep track of the total time in the meeting and how much time the timer was running vs. idle.
- Participation: Keep track of the number of turns each person got in the roles of Navigator and Driver and how much time they spent in each of these two roles.

Retrospective

The mob should be reminded to do a retrospective such that the reminder:
- Does not interrupt a turn
- Occurs early enough in the meeting that there will be time to complete the retrospective
- Preferably occurs after everyone has had one turn as both Navigator and Driver, as long as the above criteria are met first

Possible Tests

1. **Timer**:
    1.1. Functionality
        1.1.1. Enter Turn Duration in decimal minutes or minutes/seconds
        1.1.2. Display Remaining Time in minutes and seconds
        1.1.3. ▸ Start Turn
            1.1.3.1. Label => Pause
            1.1.3.2. Time Remaining after 2 seconds (string) => 4:58
        1.1.4. ⏸️Pause after 2 seconds
            1.1.4.1. Label => Resume
            1.1.4.2. Time Remaining after 2 seconds (string) => 4:58 (no change)
        1.1.5. ▸ Resume at 4:58
            1.1.5.1. Label => Pause
            1.1.5.2. Time Remaing after 2 seconds 4:56
        1.1.6. 🔔 Time expires
            1.1.6.1. Label => Start Turn
            1.1.6.2. Time Remaining after 2 seconds 0:00
            1.1.6.3. 🔔 Play sound if enabled - mocking?
            1.1.6.4. Don't play sound if disabled - mocking?
        1.1.7. ❌Cancel (stops timer and sets time remaining to 0)
            1.1.7.1. Label => Start Turn
            1.1.7.2. Time remaining: 0:00
    1.2. Tests (duplicate)
        1.2.1. Set/get Turn Duration in decimal minutes
        1.2.2. Status
            1.2.2.1. Get initial status (not running)
            1.2.2.2. Get status after ▸ Start Turn (running), ⏸️Pause (not running), ▸ Resume (Running)
            1.2.2.3. Get status after time expired (not running)
        1.2.3. Label
            1.2.3.1. Get initial label (Start Turn)
            1.2.3.2. Get label after ▸ Start Turn, ⏸️Pause, ▸ Resume
            1.2.3.3. Get label after time expired (Stat turn)

1.2.4. Time Remaining
1.2.4.1. Get initial remaining time as a string (0:00)
1.2.4.2. Get remaining time after ▸ Start Turn (5:00, 4:58), ⏸️Pause (4:58, 4:56), ▸ Resume (4:56, 4:54)
1.2.4.3. Get remaining time after time expires (0:00)
1.2.5. Time Paused
1.2.5.1. Get initial time in minutes and seconds (0:00)
1.2.5.2. Get remaining time after ▸ Start Turn, ⏸️Pause, ▸ Resume
1.2.5.3. Get remaining time after time expires (0:00)
1.2.6. ❌Cancel (stops timer and sets time remaining to 0 and sets to initial status)
1.2.7. Get Timer Mode
1.2.8. 🔔 When Time Expires: Play sound if enabled
1.2.9. ?? Get Turn Duration minutes
1.2.10. ?? Get Turn Duration seconds

2. **Mob (people)**:

2.1. Functionality
2.1.1. Get Number of People in Mob (Note: In Mobtime UI, this is shown at the top, e.g., 4 means there are four people currently in the mob)
2.1.2. ➕ Add Person
2.1.3. ⬆ / ⬇ Move Person Up or Down One
2.1.4. ✏ Edit Person (Rename)
2.1.5. ❌ Delete Person
2.1.6. Drag Person (Reorder)
2.1.7. Rotate Order
2.1.8. Randomize Order
2.2. Tests
2.2.1. ➕ Add Person
2.2.2. Get Number of People in Mob (Note: In Mobtime UI, this is shown at the top, e.g., 4 means there are four people currently in the mob)
2.2.3. ⬆ / ⬇ Move Person Up or Down One
2.2.4. ✏ Edit Person (Rename)
2.2.5. ❌ Delete Person
2.2.6. Drag Person (Reorder) - move person in between two people
2.2.7. Rotate Order
2.2.8. Randomize Order

3. **Mob Identifier**:

3.1. Functionality
3.1.1. Associate a mob with a unique string identifier
3.2. Tests
3.2.1. Create a mob named "elephant" and "tiger".
3.2.2. Set duration for the two different mobs and get the durations using the identifier

4. **Goals**:

4.1. Similar to editing Mob, except: no need for Rotate or Randomize, and:
4.2. For each goal - mark as done/not done (checkbox)
4.3. Get Number of Goals Completed out of Total Goals (shown at top, e.g., 3/5 means 3 of 5 goals completed)
4.4. Add Multiple Goals at a Time (when this option is enabled, you can add one goal per line)
4.5. 🗑️ Clear Completed Goals (deletes all completed goals but leaves those still not done)

5. **Mob Order**:

5.1. Functionality:

5.1.1. Mob Order (comma-separated list of positions/roles; default: Navigator, Driver)

5.2. Tests

5.2.1. Set/get roles using comma separated list.

5.2.2. Iterate through the list.

5.2.3. If number of people > # roles, fill in "Mob" for the roles when iterating

5.2.4. Specify default role, fill in default role for the roles

5.2.5. ?? Set/get roles using an array

6. **Client Settings**:

6.1. "You are not the timer owner" (?)

6.2. Enable timer sounds

6.3. Enable browser notifications

7. **Web app**

When person A changes mob settings on the web app, the web application for person B is automatically refreshed. Typical way to do this is with websocket. See https://ably.com/blog/websockets-vs-long-polling

For testing, you can mock sending and receiving. See https://www.npmjs.com/package/jest-websocket-mock. Testing it end to end is more complex (see here)

7.1. Get settings for a specific mob via URL

7.2. Application refresh for Duration

7.3. Application Refresh for Remaining Time after ▸ Start Turn

7.4. Application Refresh for Remaining Time after ⏸️Pause

7.5. Application Refresh for Remaining Time after ▸ Resume

7.6. Application Refresh for ❌Cancel (stops timer and sets time remaining to 0)

7.7. Application Refresh for Goals

7.8. Application Refresh for Mob Queue

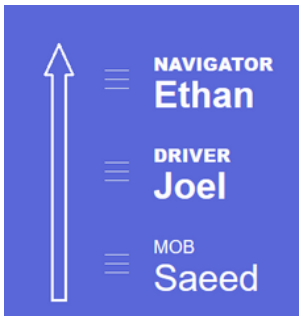
8. **Share**:

8.1. QR Code

Extended Features (Enhancements)

- **Improve what happens when rotate button is pressed in middle of turn**: When the rotate button is pressed, currently the time is moved to 0:00 (with no alarm sound). If this is done in the middle of a round, it can make things challenging. Possible remediations:
  - Change button from “Rotate” to “End Turn and Rotate”
  - When click “Rotate,” prompt to confirm rotate will end turn (or…)
  - Consider having some ability to restore the time to where it was before the rotate button was clicked, or otherwise solve the problem.
  - Consider having a way to add, subtract, or edit the time in the middle of a turn (if that doesn't cause misbehavior) - maybe requiring a majority vote (or at least one other vote...)
- **Prioritize features that would help a new mobbing group work well together ASAP, including: have visible default working agreements (TDD/Red-Green-Refactor and Driver/Navigator definitions, bias toward action, code by intent, etc.)**
- **GitPod times out after 30 min**

- **Add indicator arrow for order of mob rotation**



- **Refactor Product Name** - unhardcode strings ‘Mobtime Plus’ and make sure all instances of ‘Mobtime’ are renamed
- **Definition of Done** - Testing in all environments (web, mobile, VSCode desktop,... others?), Testing UI, etc…
- **Retrospective Timer**
- **Break Timers**
- **Stand Reminder**: Suggestion/reminder to stand (and stretch)
- **Sound Volume**: Add ability to adjust the volume of the alarm sound
- **Repeating Alarm** (more details here)
- **Animate Randomize**: Animate the UI when randomizing to make it more obvious that it's been clicked once and only once (i.e., appears unbiased / transparently fair)
- **Agenda**: More expanded agenda (in addition to mobbing, retrospective, and break timers, we might have agenda times for opening/chat, planning/closing, etc.)
- **TTS**:Text-to-speech option for announcing time for retrospective and/or maybe roles, etc. ("GPS for your meeting")
- **Persistent Client Preferences**: Persist client preferences/settings in browser local storage
- **Persistent Participants**: Persist participants automatically in cache, provide option to retrieve based on custom route ***[????]*** selected for mobtime
- **Integrated Videoconferencing**:
    - Add video+audio option (e.g., integrate with Jitsi)
    - Connect video names to mob participant names (so don’t need to enter twice) and maybe add extra highlighting around who is Navigator and Driver (e.g., thicker box, maybe larger video image)
    - Make how much time each person has spoken visible (and maybe make suggestions to amplify quieter voices) - overall, by role (driver, navigator, facilitator, etc.), for each round, trend, and cumulatively (e.g., if navigator isn’t getting to talk much, it may be a problem; maybe dynamically ask if want to add time for certain situations)
- **Add Development Environment (DE) integration**:
    - **Integrate** with cyber-dojo and/or vs code to allow users to have everything on one web page (development environment, timer, etc.)
    - Add feedback/support for **TDD**; e.g., show whether there was Red-Green-Refactor each round
- **Baton mode** - talking stick / taking turns
- **Simple breakouts**
- **Structured breakouts**, e.g.
    - 1-2-4-All
    - Other liberating structures (and other facilitation techniques)
- **Lean coffee**
- **Strong-Style Paring/Mobbing Tools** (e.g., tools to give more power/control to Navigator and/or allow others to more easily support the role)
- **Add Mob Programming RPG integration** (e.g., additional roles and checkboxes for behaviors that help the mob, similar to what is in Mobster app)
- **Voting on Next Step**: “What should the next step be: (a) More refactoring, (b) Add a similar test case to establish a pattern (before more refactoring), (c) A new test case”

- **Thumb voting**: e.g., up (yes) / sideways (willing to try, even if some reservations) / down (no) (possibly with configurable definitions/rules, e.g., pass if no thumbs down and if thumbs up are 50+%)
- **Voting types: both anonymous and non-anonymous**
- **Working Agreements** - editable, evolving, visible (draw extra attention to it at certain times, e.g., start of session and retro)
- **Signals/nudges**: rabbit hole / ELMO / parking lot, respect role of Navigator, honor agreements generally/specifically, custom/user defined,... (with "+1" capability for others to bolster the idea)
    - **Manual**
    - **Automated** (context sensitive and/or random reminders/suggestions/nudges), e.g.,
        - **Not rotating**: If writing code with timer off and not rotating frequently, give coaching message/nudge, e.g., “You’ve been coding for __ minutes with no rotation” and/or “coding for __ minutes without the timer running,” etc. (i.e., if cyber-dojo is integrated with timer, we'll know if code is being added and the timer isn't running; even without cyber-dojo integration, if we have jitsi integration, we’ll know if screen sharing hasn’t rotated)
        - **Multiple talker**s: When multiple people are talking while timer is running, put up reminder to make sure people are letting the Navigator make the final call if they want to. (Add this to the Mob time plus backlog)
- **Intro/tutorial for new mobbers (or for review/refresher)** - Design mob intro and/or pre-qualifying multiple choice questions that get new mobbers mobbing ASAP. Should be less than 5-10 min. but ideally much shorter/integrated with the experience. (This could be used for conference attendees and/or students, and could also help stratify the group if breaking into multiple mobs/breakouts to make sure some are experienced in each mob.)
- **Gitpod VS Code add-in**:
    - Write code to host timer in gitpod; not just in VS Code desktop (e.g., Mobti.me runs in VS Code desktop but not in gitpod
    - Include time not in VS Code towards time elapsed (fix bug)
    - **[???]** Get it to work in gitpod VS Code extension

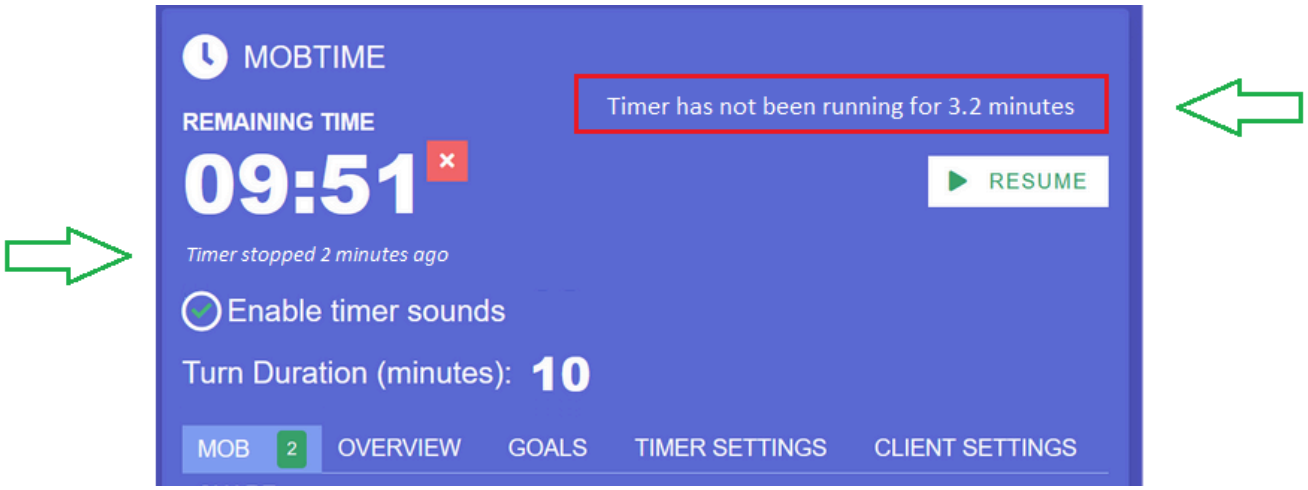
Technical Tasks

- Consider “Test Beaker”
- What about using VSCode Live? Why did we go away from that? Finding gitpod limited in terms of extensions and bracket colorizing
- Look at available JavaScript refactoring tool
- Learn about websockets
- Google "run vscode task from terminal"

Repeating Alarm

- **Proposal:**
    - Configure: Repeat alarm after x seconds (interval) and repeat the sound x times.
    - When time runs out:
        - Dialog on screen should give you the choice of
            - Starting your turn
            - Snoozing alarm for x minutes, or
            - Stopping alarm.
        - Dialog should show how many seconds
- **Principles (different people weight these differently)**:

        - Make hard to ignore.
        - Don't make so annoying people don't want to use it.
    - **Initial feature** (minimal):
        - Repeat 2x, 2 sec. apart (later maybe 4x, 10 sec. apart);
            - Number of repeats (i.e., repeat sound with no gap or 1 sec. gap)
            - Number of intervals (not needed if have snooze/stop buttons)
        - In UI, below "Enable timer sounds",
            - "Repeat sound"
                - Repeat until press stop or
                - Repeat _ times
        - later: configurable with checkbox to enable/disable, etc. **OR CONSIDER NEXT BULLET AS INITIAL FEATURE…**
    - **Additional/alternative features**:
        - **Timer Expired Additional Visual Cue**: Add an additional visual cue to make it more obvious when time has expired (this is especially important if sound and browser notifications are off).
        - **Browser Tab Text**: "Idle time: _._ min."
        - **Header** (Maybe make text red if more than a certain amount of time), options:
            - Idle Time: Timer has not been running for _._ minutes
            - Timer stopped _ minutes ago (but this



    - Maybe: **Flashing** browser tab similar to vclock:

    - Maybe: Repeating sound every __ seconds with dialog box (maybe for only a certain number of times)

Done

UI / Functional Changes

-

Technical Changes

-

State Management

In onblur

How to Deploy

The current process involves using gitpod and needs to be done any time you want to use the mob timer. Permanent deployment using github or netlify is in the works.

Note: If you don't use the mob timer for half an hour, the application times out and either the owner of the workspace needs to follow instructions for opening existing deployed workspace or you need to do a first time deployment.

First time deployment:

1. In your web browser, enter this URL: https://gitpod.io/#https://ethanstrominger/mobtime Gitpod will build you a workspace and open it with a auto-generated URL <something_random.io>, for example https://maroon_elephant_adfa334.io
2. A browser tab should open up with the mob timer using a URL which looks the same as the workspace URL except prefixed with 3000 (the port number). For example https://3000-maroon_elephant_adfa334.io The mob timer will prompt you for a mob timer id. Enter an id and press enter.
3. From a separate tab, enter https://gitpod.io/workspaces
4. Using the dropdown on the right, pin the workspace.
5. In chat, paste the URL from the mob timer window for other participants.

Opening existing deployed workspace:

1. Open the workspace either by directly entering the URL or select the workspace from https://gitpod.io/workspaces
2. A browser tab should open up with the mob timer using a URL which looks the same as the workspace URL except prefixed with 3000. For example https://3000-maroon_elephant_adfa334.io The mob timer will prompt you for a mob timer id. Enter an id and press enter.
3. In chat, paste the URL from the mob timer window for other participants.

Overview of UI Changes/Notes

Header

- Mute/Unmute *(done)*
- Notification On/Off
- Set the duration *(done)*

Goals

- Existing (no changes)

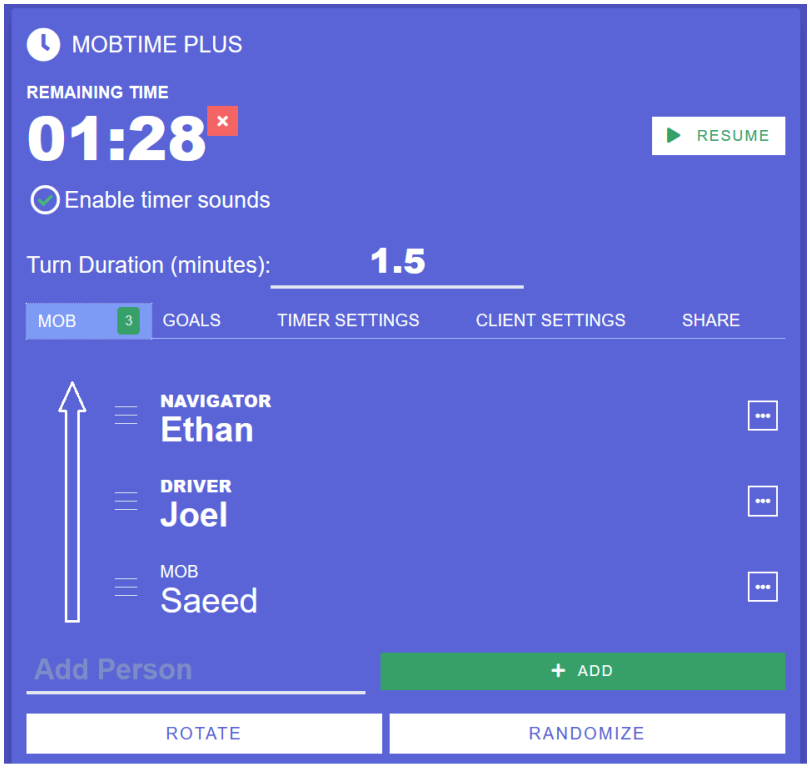
Roles

- Similar to Goals tab (commas for adding multiple roles)
- Clarify what order does

Mob
- Add Goals

General
- Improve where the controls are (not to the right, for example Edit)
- Other consolidation, names of tabs
- Smaller fonts

Current Vision (evolving incrementally)



Backlog

Code Kata / Original Mob Timer Features

Existing features of Mobti.me to be used in a code kata (business logic only; no UI so that it is TDD friendly):

9. **Timer**:
   - 9.1. Enter Turn Duration (enter time in whole minutes, e.g., 5 minutes) (maybe: change this to m:ss format instead - or decimal minutes, e.g., 2.5 minutes)
   - 9.2. Get Remaining Time: (in m:ss format)
   - 9.3. ▸ Start Turn / ⏸Pause / ▸ Resume
   - 9.4. ❌Cancel (stops timer and sets it to 0:00)
   - 9.5. 🔔 When Time Expires: Play sound if enabled

10. **Mob**:
    10.1. Get Number of People in Mob (Note: In Mobtime UI, this is shown at the top, e.g., 4 means there are four people currently in the mob)
    10.2. ➕ Add Person
    10.3. ⬆ / ⬇ Move Person Up or Down One
    10.4. ✏ Edit Person (Rename)
    10.5. ❌ Delete Person
    10.6. Drag Person (Reorder)
    10.7. Rotate Order
    10.8. Randomize Order
11. **Goals**:
    11.1. Similar to editing Mob, except: no need for Rotate or Randomize, and:
    11.2. For each goal - mark as done/not done (checkbox)
    11.3. Get Number of Goals Completed out of Total Goals (shown at top, e.g., 3/5 means 3 of 5 goals completed)
    11.4. Add Multiple Goals at a Time (when this option is enabled, you can add one goal per line)
    11.5. 🗑️ Clear Completed Goals (deletes all completed goals but leaves those still not done)
12. **Timer Settings**:
    12.1. Mob Order (comma-separated list of positions/roles; default: Navigator, Driver)
13. **Client Settings**:
    13.1. "You are not the timer owner" (?)
    13.2. Enable timer sounds
    13.3. Enable browser notifications
14. **URL**:
    14.1. Given a Mob Name, Create a URL (return https://mobti.me/ + <name entered, html-escaped>)
    14.2. Given a Mob Name, store and retrieve it
15. **Share**:
    15.1. QR Code

Deployment Plan

To Be Implemented

- Deploy using github or netlify (Ethan)
- Once it is stable, use a dedicated domain to signal the product is prime time

---

Retro

Possible Formats:

1. Format(s) we started with:
    a. **Feel, Like, Improve**
    b. **Feel, Like, Challenges, Improve**
2. Formats copied from https://miro.com/guides/retrospectives/ideas-games :
    a. **Mad, Sad, Glad**
    b. **4 L's:** Liked, Learned, Lacked, Longed for

c. **Starfish:** Keep doing, Do less of, Do more of, Stop doing, Start doing
d. **Sailboat:** What propelled us forward (like wind) , What held us back (like anchors)
e. **KALM:** Keep, Add, More, Less
f. **SWOT:** Strengths, Weaknesses, Opportunities, Threats
g. **Timeline Retrospective**
h. **Tell a Story** - silently write a story about what happened, incorporating “shaping” words, such as “mad, sad, glad” (or something else), plus actions
i. **Start, Stop, Continue**
j. **Scrum Values**: Openness, Courage, Focus, Respect, Commitment

4/7/2022:

Retro #2:

Ethan

- Feel: Eh
- Like: Progress
- Challenges
- Improve: Follow Navigator more; maybe make turns longer. Less debate on backlog tasks/tests think we’ll need ahead of time; instead: more “just start TDD and see what happens”

Joel

- Feel: Happy
- Like: Refactoring (code is much cleaner now with get/set, underscores for fields, etc.)
- Challenges
- Improve: Speak up if think Navigator isn’t being followed.

Retro #1:

Ethan

- Feel: Hungry
- Like: Rhythm
- Challenges
- Improve: Consider “Test Beaker” **(add to technical)**

Joel

- Feel: Good (rhythm)
- Like: Rhythm. Talked about disagreement/opinions on TDD.
- Challenges: Tests aren’t always reliable (i.e., maybe running before compile is done?). Inconvenient to see why test failed (Is there a way to make the terminal vertical?).
- Improve:

3/31/2022:

Ethan:

- Feel: Pleased / happy
- Like: Talked out different ideas.
- Challenges: Different ideas about what to do.
- Improve: Backlog could be neater, e.g., technical and feature backlog. Remember to give credit to co-author.

Joel:

- Feel: Happy
- Like: We tried something even though both didn't agree up front. Good to try 2 ways if need be. Bias toward action. Commit frequently.
- Challenges: Remembering how to compile and run tests - change directory, run only *.js tests (not ts or get errors)
- Improve: Spend time on Backlog refinement.

3/24/2022:

Ethan:

- Feel
- Like:
    - Working together (+1)
    - Working on the tutorials, understanding the basics; hello world (+1)
- Challenges
- Improve:
    - Speak up when want to **move on**. (And other person try to notice when other(s) want to move on.)
    - Speak up when think wording is **good enough**.

Joel:

- Feel
- Like:
    - Ethan speaking up
- Challenges
- Improve

3/10/2022:

Retro 1:

Joel:

- Feel: So so
- Like: Looked for all occurrences
- Challenges: Coding sooner
- Improve:
    - Don't jump into documenting
    - Add backlog item for refactoring to separate out code
    - Commit frequently on feature branch

Ethan:

- Feel: So so
- Like: Started to see how other code works
- Challenges: Coding sooner, not going down rabbit holes
- Improve:
    - Don't jump into documenting

    - Immediately after understand code, document it

3/3/2022:

Retro 2:

Ethan:

- Feel: Good
- Liked: Started a new task after break
- Improve: Keep using timer during longer discussions

Joel:

- Feel: Okay (planning, UI discussion not favorite)
- Liked: Doing two retros, much better rotating and keeping navigator roles, working on feature that might help us

Retro 1:

Joel:

- Feel: Okay (exploration not favorite)
- Liked: Time boxed, brought it to completion
- Improve:
    - Make Ethan's screen full while still seeing multiple browser tabs
    - Get language to describe physical VSCode

Ethan:

- Feel: Okay
- Liked: Figured out state and Actions.Init, move tasks to Done
- Improve:
    - Write up notes to remind myself of times I think could be improved
    - In retro, talker and scribe should be separate people

3/2/2022:

Ethan:

- Feel: Curious
- Liked:
    - Refactoring of tests
    - Digging into understanding the state variable.
- Improve:
    - **When in exploration mode**, still have one person be Navigator (captain) - and that person really needs to explain themselves and ask if the other person understands. Start with intent - explain why want to do something first (before asking Driver to do something). When mob timer expires, ask “are we being productive?” (avoid rabbit holes)
    - **DONE: Copy into this doc our working agreements (TDD, bias toward action, code by intent, etc.)**

Joel:

- Feel: Happy
- Liked:
    - Speaking up about recentering on our agreements re. timer and roles
    - Liked trying to understand the state variable and also was glad time box expired
    - Working on / paying attention to tests

2/24/2022:

Ethan:

- Feel: Excited
- Liked:
    - Got to coding quickly.
    - Got feature working.
    - Noting what we were going to do next (very specific).
- Improve:
    - Try Codespace

Joel:

- Feel: Great (got fractional minutes working!)
- Liked:
    - Rotating every 7 minutes.
    - When between-session branch wasn't working, we abandoned it quickly (for later between session work to rectify).
    - Refactoring - especially remaining js file and function to match setting name (setDuration instead of setLength).
    - Get visible functionality working every time when can; e.g., fractional minutes is working!!!
- Improve:
    - Deployment / refresh issue

2/17/2022:

Ethan:

- Feel: Happy
- Liked: Not too difficult to pick up again after a one-month hiatus. Learned more about the code for setDuration, and both of us had ideas that led to fruitful exploration (e.g., change to onblur, finding where duration was used, looking at what we did for sound, componentizing, etc.)
- Improve: *None at this time.*

Joel:

- Feel: Good, a little tired
- Liked: Multiple breaks since extra long session today. Good exploration/trying different angles/hypotheses. Did retro. Coded pretty quickly rather than too much time discussing (e.g., merging with Mrozbarry's branch) or doing environmental work (settings, repository stuff, etc.).
- Improve: *None at this time.*

12/16/2021:

Retro 2:

Ethan:

- Feel: Frustrated & hungry
- Liked: Getting to UI
- Improve: Moved everything out of wiki and into this Google document

Joel:

- Feel:
- Liked: Closed and reopened Firefox entirely to get Gitpod working again
- Improve:
    - Gitpod timeouts (slowed us down a lot). Tips:
        - Closed and reopened Firefox entirely to get Gitpod working again
        - Be mindful of 30 min. timeout: Refresh/wake it up before/after retros/breaks

Retro 1

Ethan:

- Feel: Hungry (need food)
- Liked: Simple to make code change. Having environment instructions on hand (made things faster, less struggle).
- Improve:
    - Add instructions for creating feature branches or checking out existing branches.

Joel:

- Feel: Good.
- Liked: Flow of looking at backlog and getting to work quickly. Easier to code now and make progress.
- Improve: Move retro to Google Docs (wiki doesn't autosave and has a small edit box - plus not WYSIWYG)

Retrospectives by Date:

12/14/2021:

Retro 1

Ethan:

- Observations: It took a while to remember what we did last time and get dev environment ready to use again.
- Feel: Engaged
- Liked: Notes from last time. Separate branch for broken stuff. Making progress. Learning. Sharing Gitpod workspace.
- Improve: Longer sessions and not as many (to improve flow/efficient start up)?

Joel:

- Feel: Engaged

- Liked: Same as Ethan's above, plus: Using different port so both of us could run the same Gitpod workspace. Keeping the old code until the new code works (always easy to get back to green - and can see something that works).
- Improve: Make note of what branch we're on at end of session.

12/9/2021:

Retro 2

Joel:

- Feel: Hopeful
- Liked: Componentizing and refactoring, excluded commit, change time to 10, put on separate list, good percent coding
- Improve: Nothing

Ethan

- Feel: Engaged
- Liked: Componentizing and refactoring, excluded commit, change time to 10, did not put off discussion
- Improve: Finalize some things about the process - squash or not, when to do branches - did it at same time, also discussed deleting branched

Retro 1

Ethan:

- Feel: Hyper (in a good way)
- Liked: Everything
- Improve: None yet

Joel:

- Feel: Good
- Liked: Using mobtimer and TTS agenda
- Improve: None yet

Template:

Retro _:

Ethan:

- Feel:
- Liked:
- Improve:

Joel:

- Feel:

- Liked:
- Improve:

Appendix

a. If creating a new Gitpod Workspace for **Deployment** (because old one timed out):
    - https://gitpod.io/#https://github.com/ethanstrominger/mobtime
    - **Pin it in** your list of workspaces (pinned workspaces are kept forever / not deleted).
    - update URLs in documents
b. If creating a new Gitpod Workspace for **Development** (because old one timed out):
    - https://gitpod.io/#https://github.com/ethanstrominger/mobtime
    - Wait for workspace to open
    - https://gitpod.io/workspaces
    - **from drop down, select Share**
    - update URL below