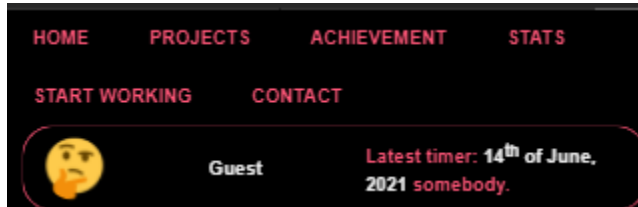


## The Header:



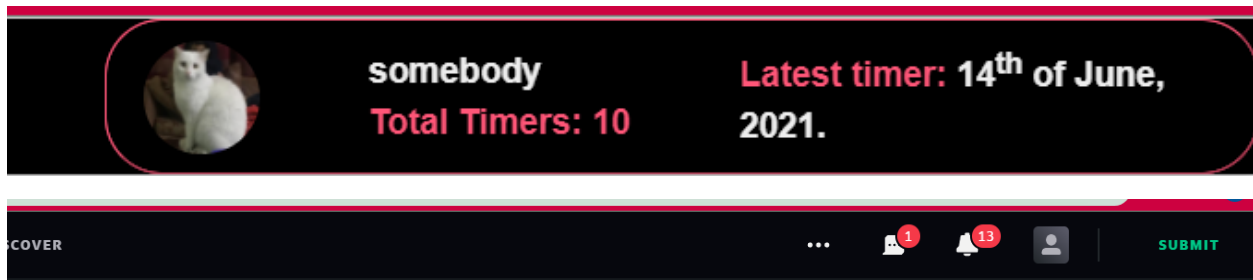
The header is the simplest and most ubiquitous concept. As such, the implementation matches its complexity.

The header uses the *grid* layout for its main layout. As such, the main header is responsive. This is the header in a narrow screen (400px).



The left side simply consists of links to other parts of the site. The other names are self-explanatory and *Start Working* leads to the timer page.

To its right we have the User information panel. The inspiration was from modern art websites such as DeviantArt or ArtStation which have the user information in a similar view:

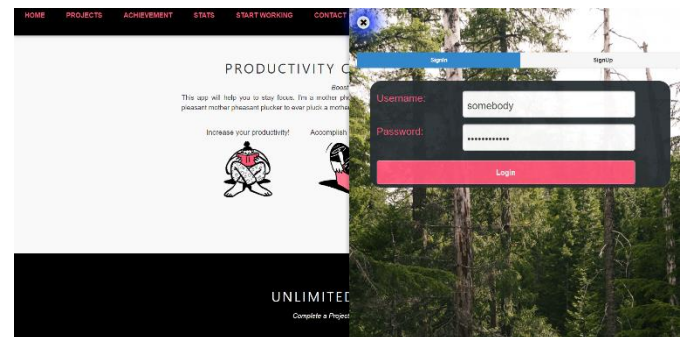
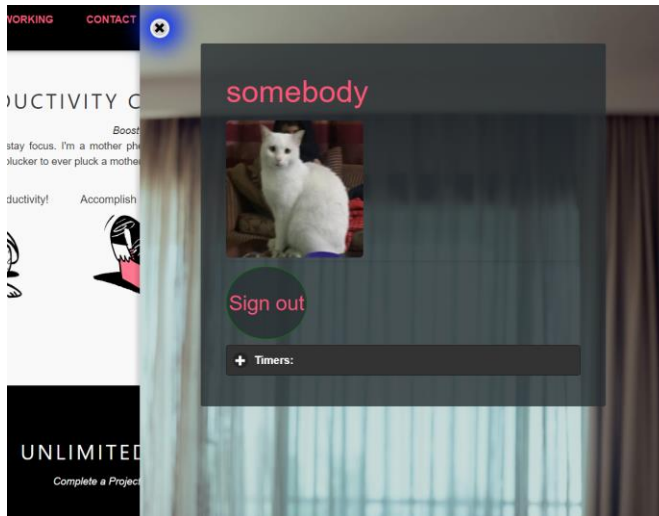


The user avatar is followed by the username and the latest timer they undertook. In case of a guest account (one that hasn't currently signed in), it displays the default avatar, along with the latest timer that was undertaken and the identity of that user.

Since the header is present on all pages, it is only feasible to include the related functionality with it.

As such, the header contains functionality for areas such as logging in/signing up or checking your statistics etc.

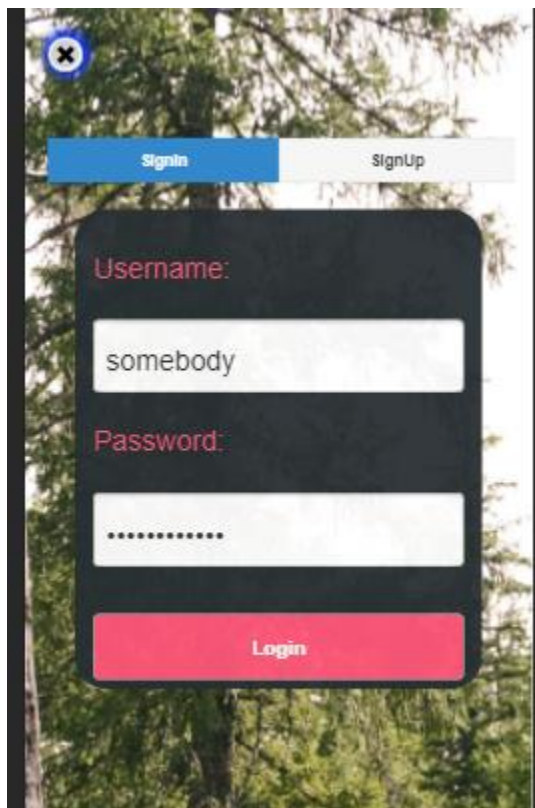
Clicking on this panel reveals a JQM slide-out panel, properly indented and sized for further uses.



Signed-in user (left) and guest user (right)

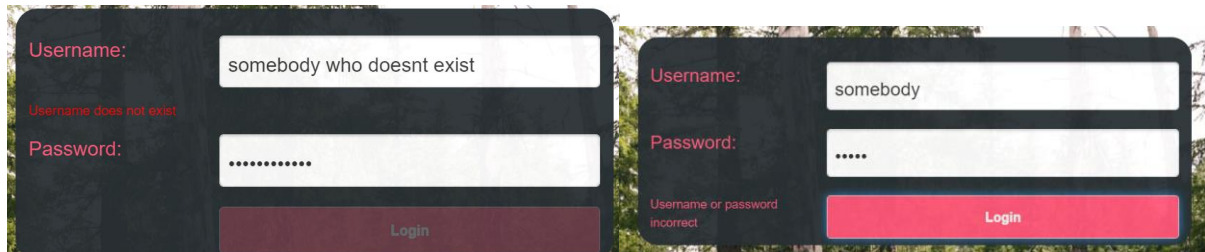
Further, the panel covers the whole screen under narrower screens and thus displays all its contents properly, as can be seen in the following screenshot, where the sign in panel squeezes to make room for the inputs and labels on small screens.

The forms inside are wrapped in jQuery mobile tabs, to make the interface cleaner.



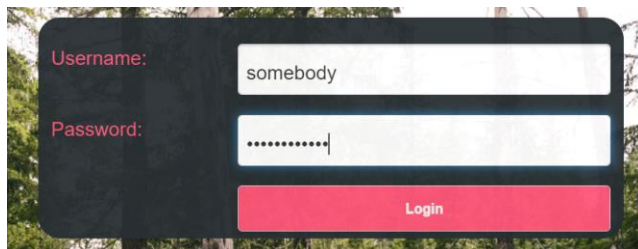
Sign in Procedure:

Once the user types into the Input labeled Username, the database is checked in AJAX behavior for the username, and the user is promptly notified it is not. A similar mechanism exists for password, however, the password is checked only when Login is pressed, to reduce the queries made for the password.



Two screenshots of a login form. The left screenshot shows an error message "Username does not exist" in red text above the Password field. The right screenshot shows an error message "Username or password incorrect" in red text below the Password field. Both forms have a "Login" button at the bottom.

For correct credentials:

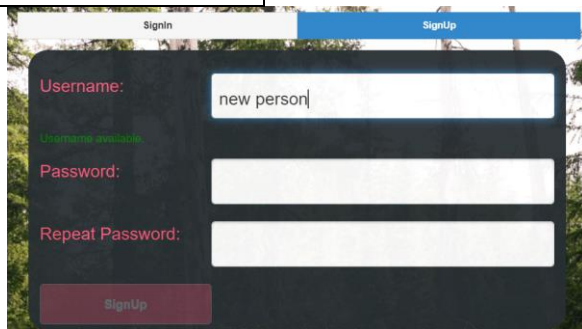


A screenshot of the login form with the "Login" button highlighted in red, indicating a successful login attempt.

Even if the user somehow manages to enter the wrong credentials through an attack, the page checks again when the actual sign in is performed.

Once the user is signed in, the page reloads and is ready to go.

#### Sign Up procedure:

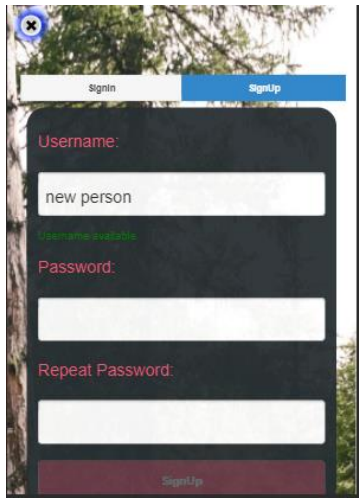


A screenshot of the Sign Up form. The "Username" field contains "new person". The "Password" and "Repeat Password" fields are empty. The "SignUp" button is disabled (greyed out). A green error message "Username available" is visible above the Password field.

The signup has similar mechanism to the signin. When something is entered in username however, it does the opposite, and checks if the username does **not** exist, and disables the Sign up button otherwise.

The same goes for *Password* and *Repeat Password* fields, that disable the Sign up button if they do not match.

As seen with Sign In, Sign Up is also responsive as illustrated by the following screenshot of it on a narrow screen.



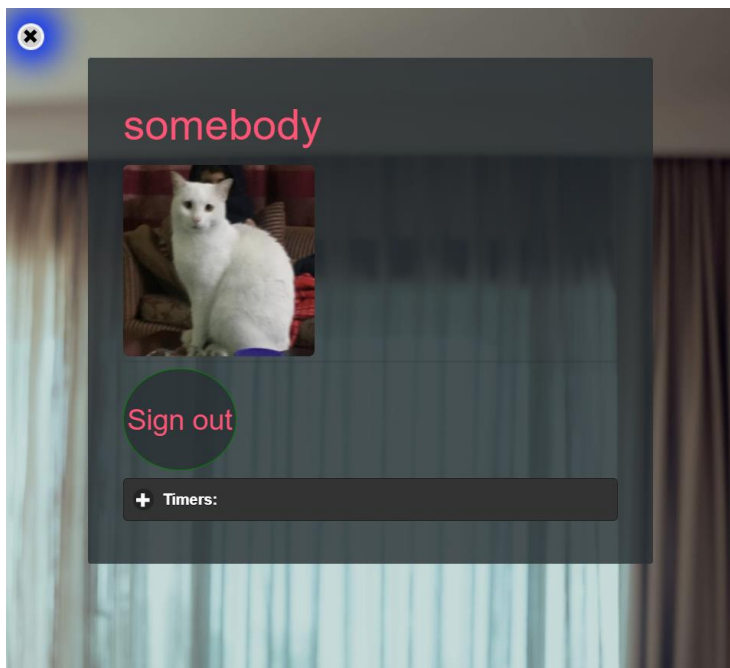
Upon creation of the new user, a default avatar is assigned to the user, and the necessary adjustments are made in the database. The user is ready to go ahead with the productivity:



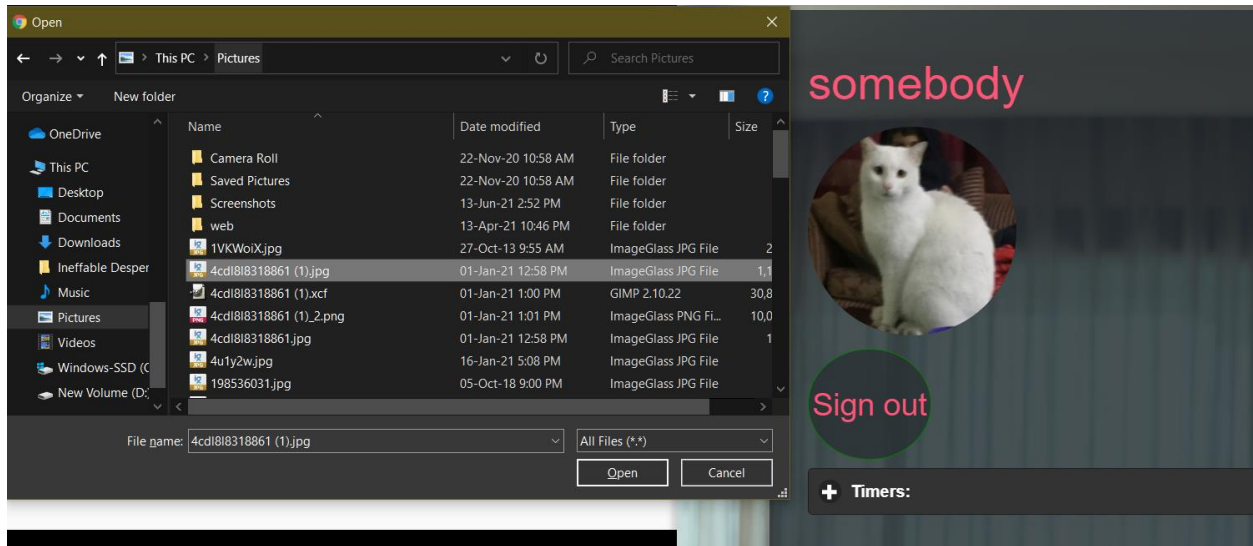
#### User Details:

If a user is signed in, the panel displays their avatar and their latest timers.

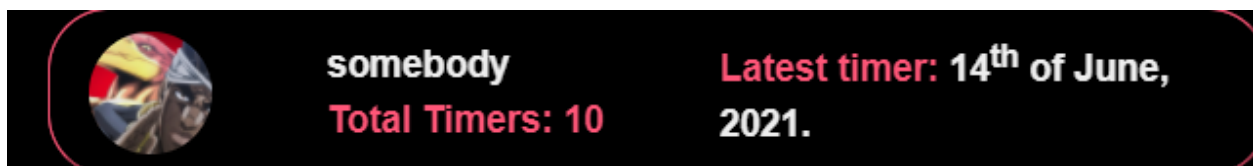
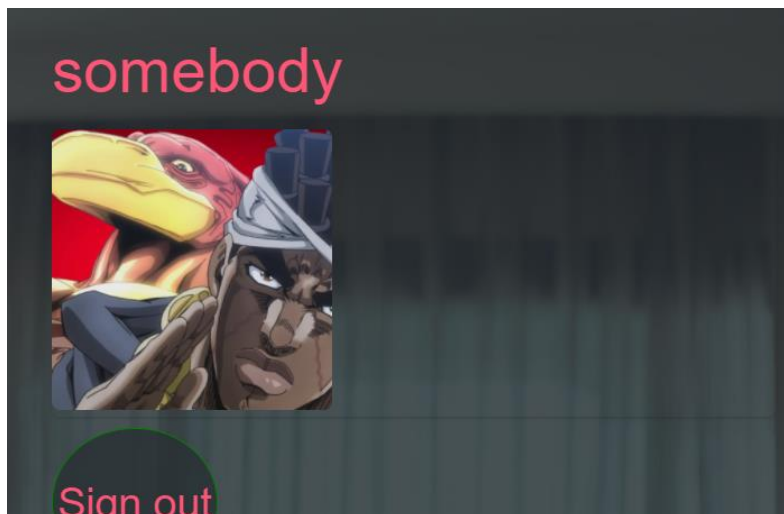
The full panel displays looks as follows:



The first is the username and a bigger avatar. Hovering over it rounds it, as if it was in the top panel again. Upon clicking on this avatar, a file picker opens:

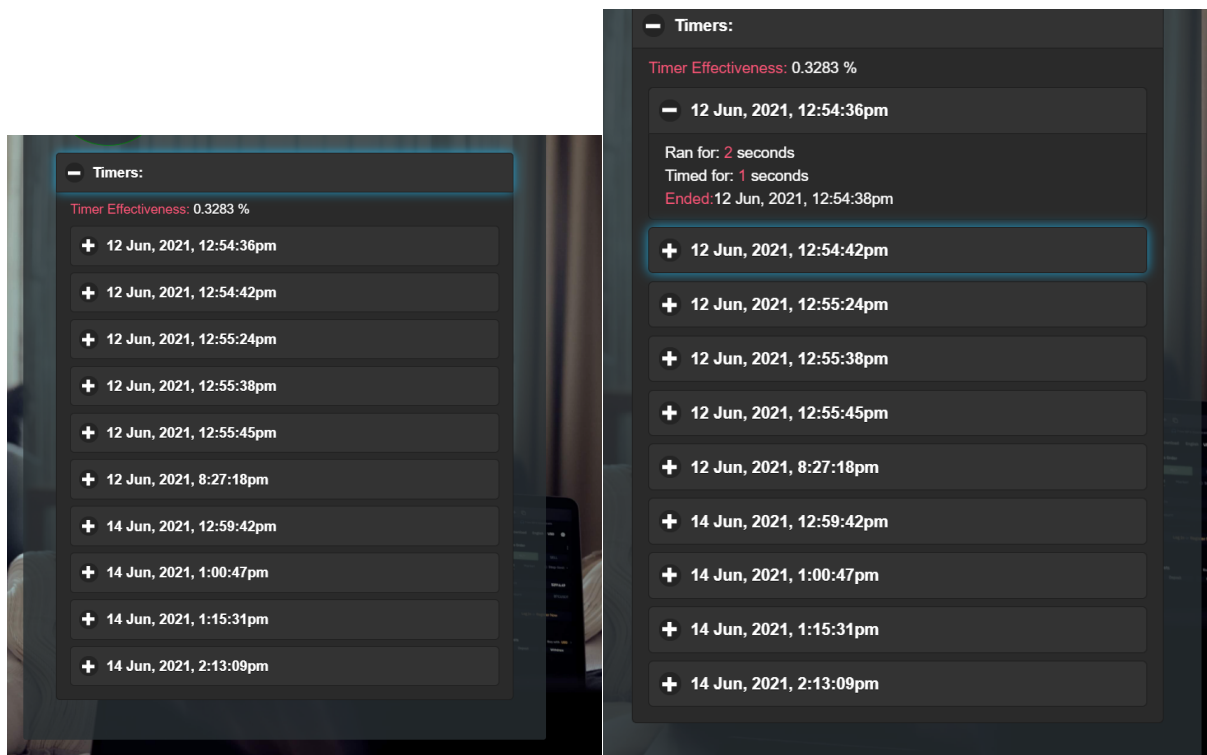


When a picture is selected, it uploads to the relevant folder, then replaces the old avatar (if exists). The page loads it into all the fields that use it in AJAX so there is no need to refresh:



Timers:

Just below the sign out button is the container for the timers. It is collapsible, so as to make space for further components.



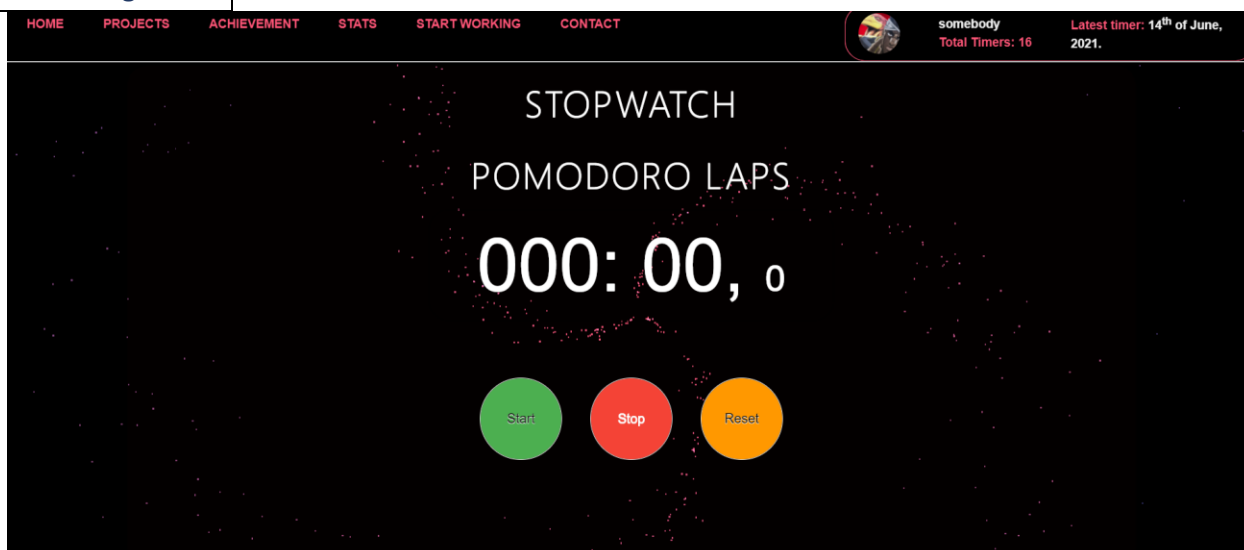
At the top is the Timer Effectiveness, which is an unrounded measure of how much time a user spends in timers, as compared to how long that timer actually runs, more on that later.

The timers are all in collapsible sets, and arranged in order of their starting time.

The whole container is wrapped in a JQuery Mobile theme, so the colors cascade down with proper color gaps.

## The timer:

The background:





This is what the user sees upon opening the page.

The background is much more complex than it first appears. It is, in fact, a **dimension** greater than a simple background.

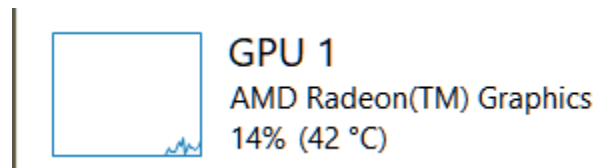
The background uses ThreeJS, a JavaScript library based on WebGL, that generates a **1000** particles in branches. The entirety of this simulation is customizable, from the number of branches to the amount of randomness and spin.

More branches	More Spin	More radius
		
Different outer color (violet trail)	Different inner color	More randomness and particles
		

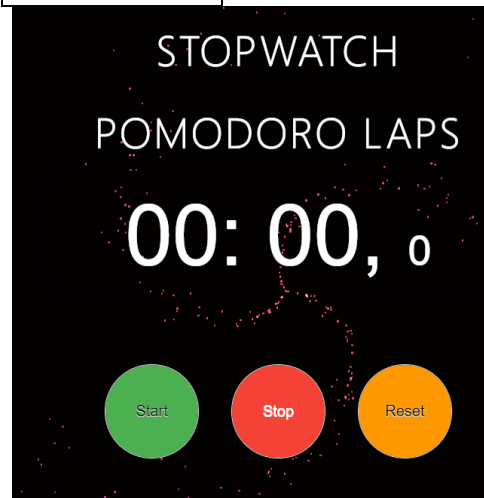
The simulation uses the most efficient methods of rendering, curated specifically for this render, with no shadows and minimal color grading, no physical calculations and only light arithmetic.

It has been tested on an AMD Radeon Vega integrated GPU, with the maximum limit of particles being 20000, with a refresh rate of 144Hz. The page was able to keep up even at the particle limit with no noticeable lag, keeping a frame time of 1ms. By current standards, the page should be able to run smoothly on a mobile phone up to 90Hz.

Further, the simulation is not tied to pixel ratio. Instead it is clamped to a pixel ratio of 2, which is why it should also be fine for devices with ultra-high pixel resolution such as high resolution mobile phones.

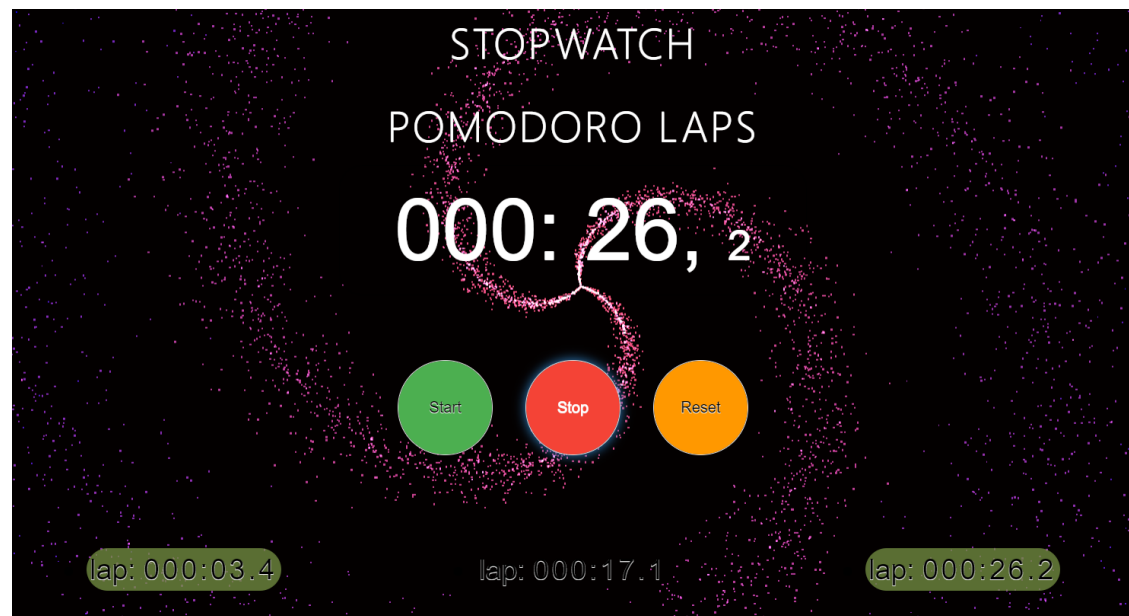


The pomodoro:



The user starts a timer by pressing Start. The page sends a request and the timer is recorded in the database to have been started

Then, every time the user presses Stop, a lap is recorded. This simulates the premise of a pomodoro, where one keeps track of how much time they work and rest.





The working time is described by the green laps, and the resting time has no color around it. The pomodoro session ends when:

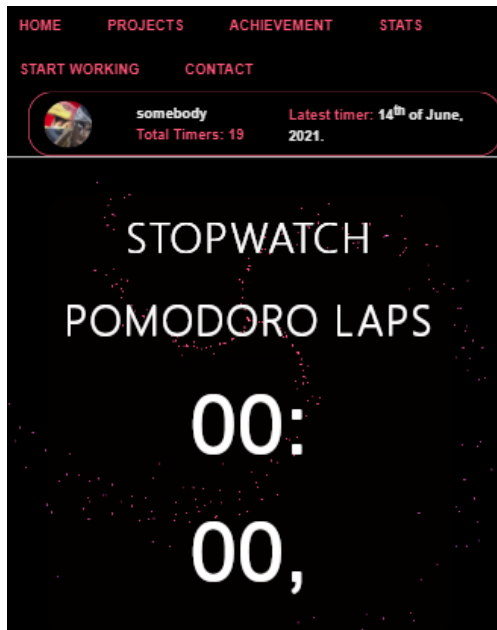
- The user presses Reset.
- The page is closed.
- Six pomodoro laps are reached (three repetitions are proven to be the maximum limit for productivity).

Pomodoro efficiency is defined as the time a person sticks to the pomodoro routine in the session.

Once reset, the started timer is updated in the database to have been closed.

Depending on whether or not a user is signed in, the timer is recorded either to the user's name, or to no user (which still contributes to total timer efficiency).

The whole timer is responsive to screen size:



## Stats:

---

The stats page simply describes various functional statistics of the website.



Ran for: 18 minutes 29 seconds

Timers for: 1 minutes 45 seconds

Average Efficiency: 45.61%

Active Users: 7

Total Projects: 4