CA326

Well-being as a Measure of User-interface design

Technical Specification

Students:
Alif Hossain - 17314941 - alif.hossain5@mail.dcu.ie
Imrich Toth - 19307456 - imrich.toth2@mail.dcu.ie

Supervisor:
Hyowon Lee - hyowon.lee@dcu.ie

Date Completed: 04/03/2022

Table of Contents

Introduction

Overview

For our 3rd year project we chose the following topic “Well-being as a measure of user-interface design (Time Out)”. We chose this as we both believe we can provide a solution to users dealing with current or future health issues due to prolonged screen usage. This directly correlates with our topic in user interface design.

We managed to find a solution to this in the form of giving the users an option to limit the screen time usage of the browser they are using. This was achieved by creating an extension that is compatible with a variety of browsers. The extension allows the user to set a timer with the options of using hours, minutes and seconds. The set time can be used within a number of selected sessions by the user.

E.g. If a user wanted to split 3 hours over the day into 1 hour time slots, they would do so by setting 1 hour in the timer and No. of sessions as 3.

After the set timer has run out the users browser will be presented with a popup box stating how long they have spent on the screen and will be given the options to “Take a break” or “Later, busy at the moment”. If the user selects take a break then the program will pause the timer and not start the next session if there is a next session until the user decides to come back to the screen and is ready to continue into their next session. On the other hand if the user selects the “later” option, they are indicating that they are doing something important and do not wish to take a break yet, therefore the timer will run into the next session and will continue running until either the user pauses it or it comes to the next break. It is key to note that this extension works on **Google Chrome, Microsoft Edge, Brave Browser and Mozilla Firefox.**

Glossary

JavaScript: JavaScript is a programming language that is used to create interactive online applications. JavaScript may be used to power features such as interactive pictures, carousels, and forms. The language may be used in conjunction with back-end frameworks such as Node. js to power the mechanics of a web page, such as form processing and payment processing.

HTML: The coding used to organise a web page and its content is known as HTML (Hypertext Markup Language). For example, material might be organised in a series of paragraphs, a list of bulleted points, or by including graphics and data tables.

CSS: Cascading Style Sheets (CSS) is a style language used to specify the appearance of an HTML or XML document (including XML dialects such as SVG, MathML or XHTML). CSS specifies how items should be shown on screen, paper, in speech, or in other mediums.

Web Extension/Add-ons: A browser extension adds new features and functionalities to the browser. It's built using well-known web technologies including HTML, CSS, and JavaScript. It can use the same web APIs as JavaScript on a web page, but an extension can also use its own set of JavaScript APIs.

Frontend Coding: Front end development is a type of computer programming that focuses on the coding and building of website elements and functionality that the user will see. It is concerned with ensuring that the aesthetic parts of a website are functioning. Consider the front end to be the "client side" of an application.

Backend Coding: A website's backend is made up of servers, apps, and databases. Backend developers provide code that allows browsers to communicate with databases and store data into the database, read data from the database, update the data, and remove data or information from the database.

DOM: The Document Object Model (DOM) is a computer API that is used to create HTML and XML documents. It specifies the logical structure of documents as well as how they are accessed and changed.

Gitlab Repository: A repository is a location where you may save your code and make modifications to it. Version control keeps track of your modifications. Each project has its own repository.

System Architecture

For our web extension we have incorporated 3 different programming languages. JavaScript, HTML and CSS were used for both the frontend and the backend of the extension. Like any other extension we created a manifest.json file which provides vital information about your extension to Chrome, such as its name and the permissions it requires.

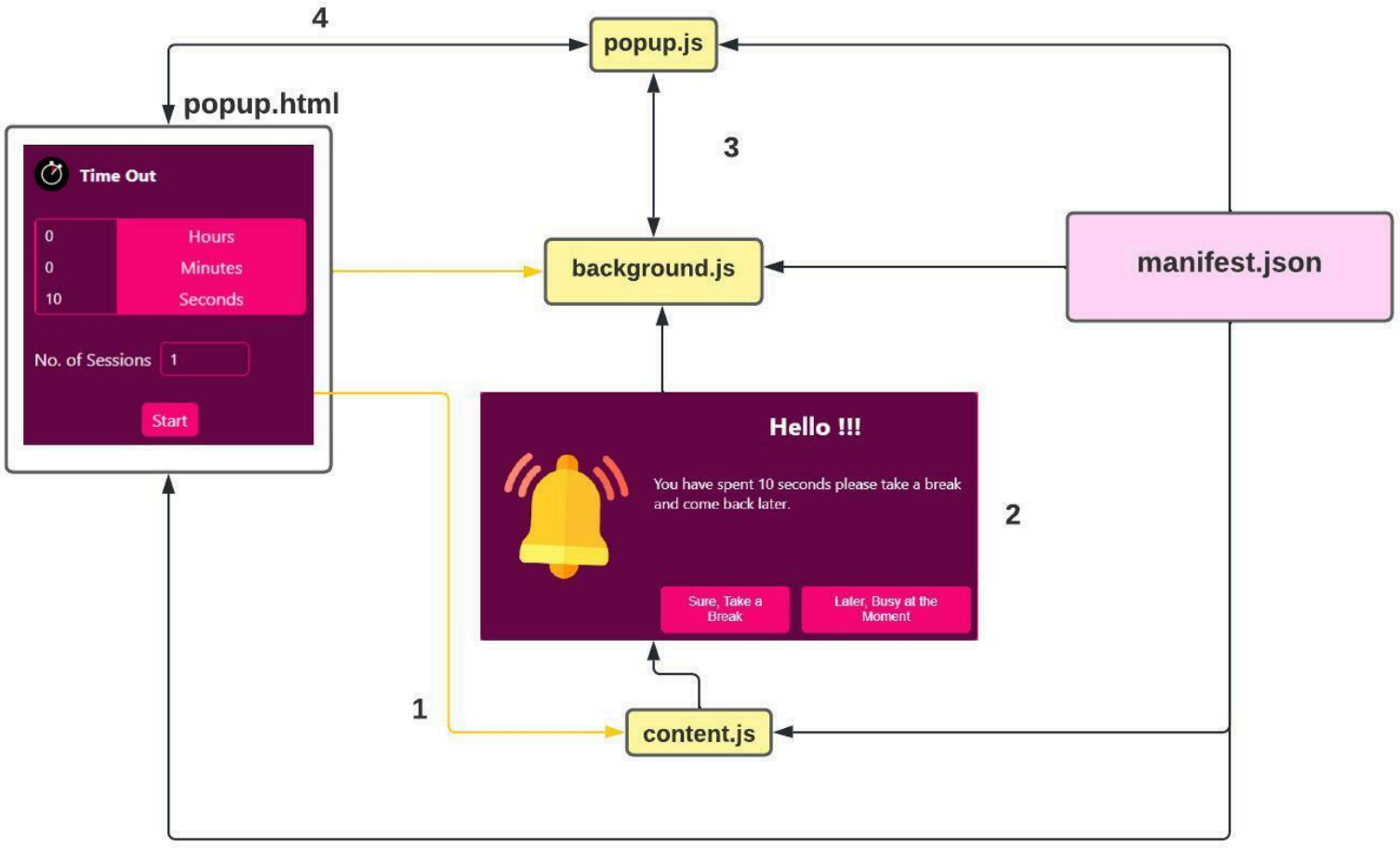
The popup refers to the primary page that visitors will view when they use your addon. It is made up of two files: Popup.html and Popup.js, which is written in JavaScript. Popup.html is the layout file for your extension's appearance. The markup of this file will change depending on what your extension does. Popup boxes are used in Javascript to show a message or notice to the user. In JavaScript, pop-up boxes are classified into three types: alert boxes, confirm boxes, and prompt boxes. When a warning message is required, the Alert Box is utilised.

The background script ('background. js') is a JavaScript script that runs when our extension is installed or when the user manually refreshes the extension. Because the background script does not have access to any of the URLs the user is viewing, it cannot change the DOM. Only a background script may interact with events and make use of the Chrome API.

Content scripts(content.js) are executable files that execute within the content of web pages. They can read the details of the web pages visited by the browser, make modifications to them, and communicate information to their parent extension by utilising the standard Document Object Model (DOM).

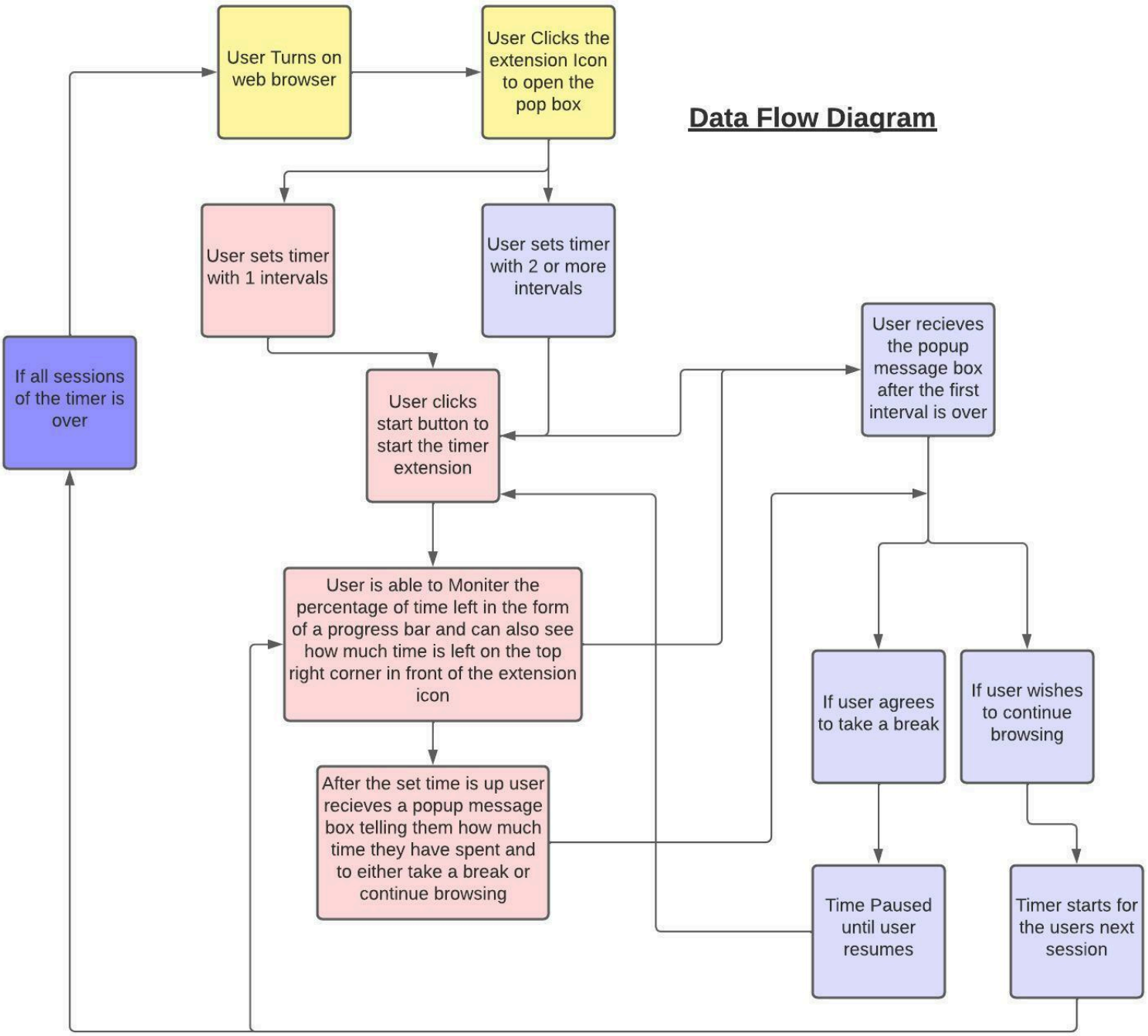
We also created the CSS for our model.css box which shows the message popup box with the animation framework. Alongside the model.css which laters calls on content.css, which contains the css for the timeout overlay and container. Lastly the popup box with the timer option is designed on the popup.css.

System Architecture For Time Out Extension

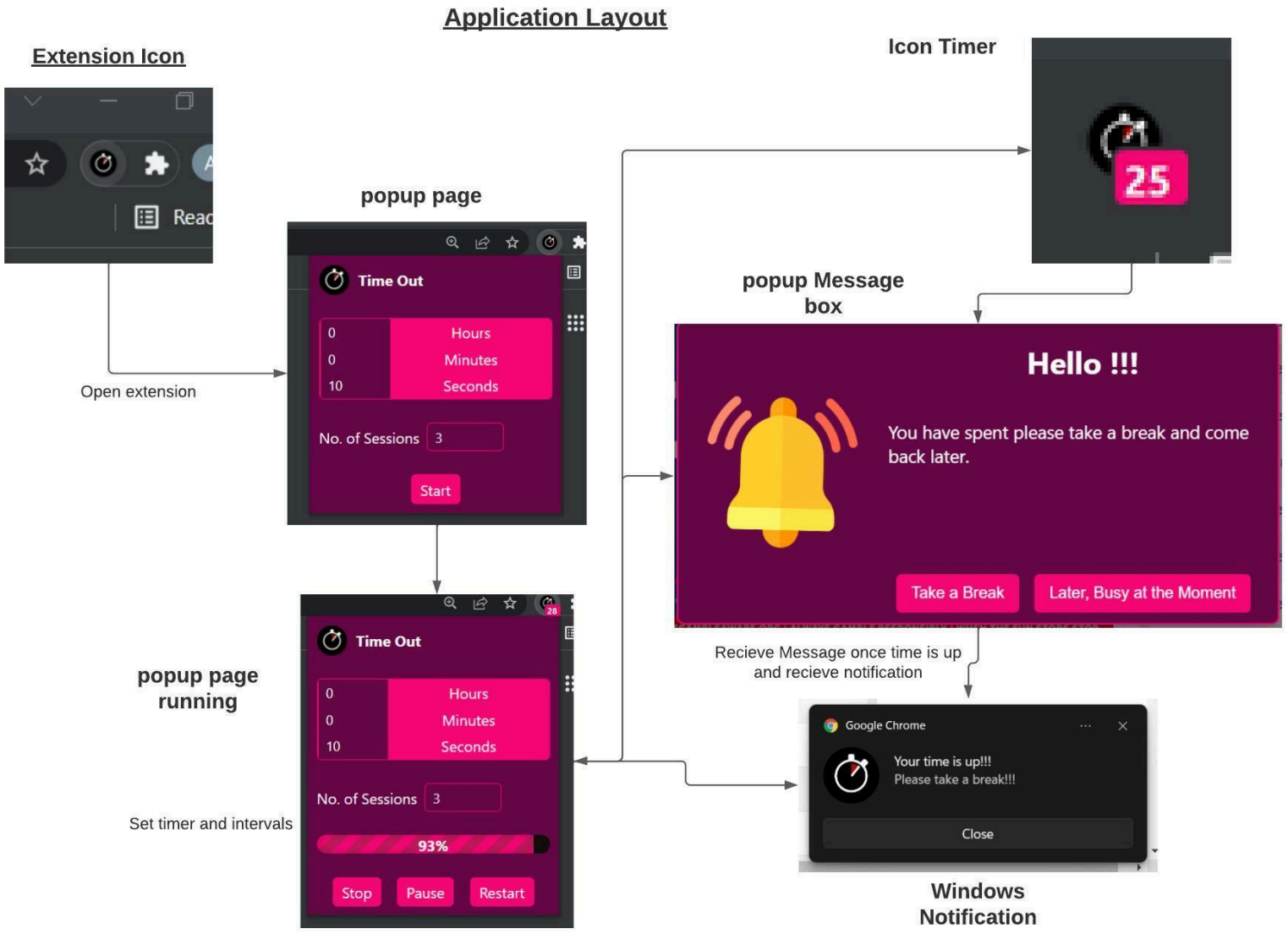


High Level Design

Data Flow Diagram

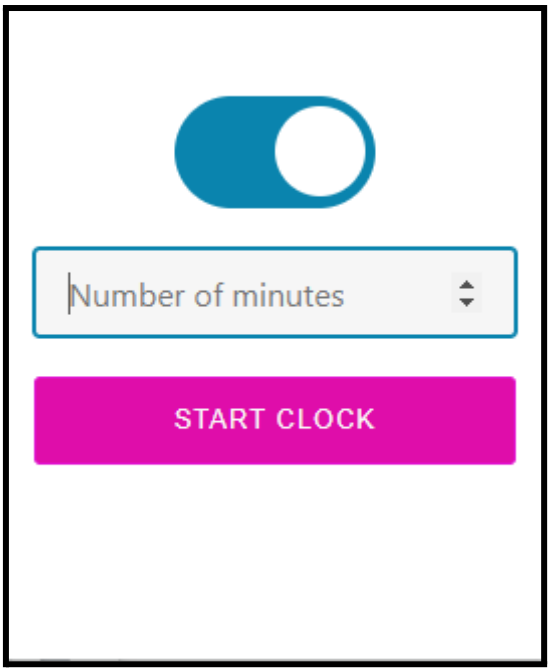


Application Layout



Design

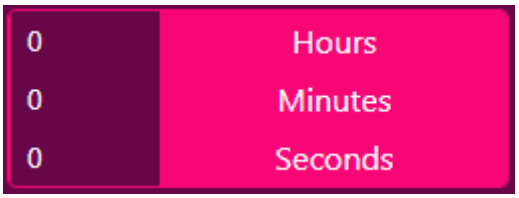
Initial Design:



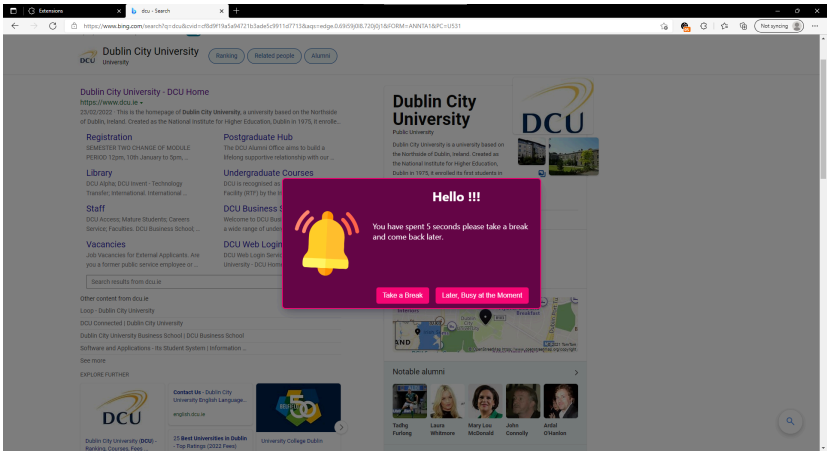
The initial design was not aesthetically pleasing with enough options for users, it was too cumbersome and lacked key information such as a visual representation of the timer counting down, a lack of satisfactory notification upon the timer finishing and no other option for time length other than minutes.

Final Design:

Our final design contained multiple major changes and improvements compared to the initial design. The first key feature we wanted to implement was the option for users to set timers for hours, minutes, and seconds, all at once or separately (user's choice).



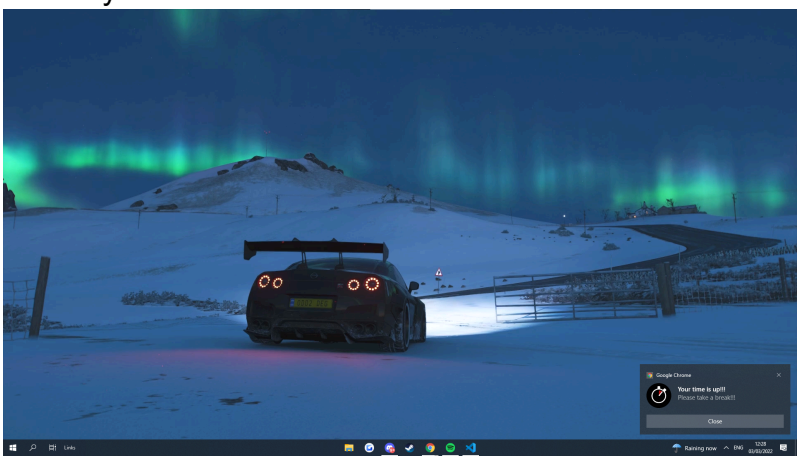
The second major change that was made consisted of adding an animation with buttons as options for users once the timer has finished counting down, the user then has the options to take a break from their screen or continue to work as they are doing something important, which results in the timer continuing into their next session.



The third feature we decided would be a good idea was to add a small visual timer right on top of the extension icon in the top right corner of the browser. We decided this would be a good option as this would allow the user to easily track how long they have left by simply glancing at the top right corner instead of having to click on the extension and give their full attention to it. 

We also had to consider that users may not only use Chrome as their browser, therefore we made sure that this extension works for a number of different browsers. Our product works on not only Chrome but also Microsoft Edge, Mozilla Firefox and Brave Browser.

Finally we needed to take into account if a user was tabbed out of their browser and in another application or simply on their desktop, in this instance the timer keeps running as long as chrome is actually open and the user will receive a google chrome notification on top of the animation mentioned above, this notification will pop up for users while they are not in Chrome.



Problems and Resolution

Problem 1 - Understanding how web extensions operate.
The first problem we dealt with once we came up with the idea and started the project was to find out how a web extension/add-ons operate.

Solution
To solve this we each carried out research in order to find out more about the concepts and structures of an extension. Also discover information about the technical aspects of the extension development.

Problem 2 - Deciding the software.
After carrying out our research we needed to decide on the suitable programming software to use, in order to build the web extension.

Solution
After talking with my partner we both decided on using JavaScript alongside HTML and CSS for the front end. We chose this as it was the most convenient and efficient way to build a working web extension that will function on a variety of browsers.

Problem 3 - Animation after timer runs out
We needed to implement an animation that blocked the usage of the browser until the user selected an option.

Solution
We used webkit keyframes within the css to design the animations for this feature. Below is a short snippet of one webkit-keyframe that we used.

```
@-webkit-keyframes scale-in-center {
  0% {
    -webkit-transform: scale(0);
    transform: scale(0);
    opacity: 1;
  }

  100% {
    -webkit-transform: scale(1);
    transform: scale(1);
    opacity: 1;
  }
}
```

Problem 4 - Lack of timer options
We found that the user lacked the ability to select their time limit for the timer extension. We had to solve this in a way where the user had more options.

Solution
We achieved this by adding in input boxes which allowed the user to pick from hours, minutes and seconds for their timer. This allowed the user to have more say in the specific length of the timer.

```html
<div class="time-row">

    <div class="hour-input-group input-group">
        <input type="number" min="0" max="24" class="hours">
        <div>Hours</div>
    </div>

    <div class="minute-input-group input-group">
        <input type="number" min="0" max="59" class="minutes">
        <div>Minutes</div>
    </div>

    <div class="second-input-group input-group">
        <input type="number" min="0" max="59" class="seconds">
        <div>Seconds</div>
    </div>

</div>
```

Problem 5 - User needs timer information
From reviewing the extension application with our supervisor Hyowon Lee, we were informed that the user needed a way where they would be able to see the remaining time left on the timer without having to open the popup box.

Solution
We resolved this problem by putting a badge in front of the extension icon which allowed the user to see the remaining time left on the timer at a glance, without having to press on the extension icon or other buttons.


```javascript
function getTimeForBadge() {
    let h = Math.floor(secondsLeft / 3600)
    let m = Math.floor(secondsLeft % 3600 / 60)
    let s = Math.floor(secondsLeft % 3600 % 60)

    let hDisplay = h > 0 ? h.toString().padStart(2, '0') : ''
    let mDisplay = (h > 0 ? ':' : '') + (m > 0 ? m.toString().padStart(2, '0') : '')
    let sDisplay = (m > 0 ? ':' : '') + (s >= 0 ? s.toString().padStart(2, '0') : '')

    return hDisplay + mDisplay + sDisplay
}
```

Problem 6 - Chrome notifications
We needed to take into account that some users may be tabbed out of Google Chrome or even in another application while keeping Google running

Solution
We simply implemented some code within one of the javascript files to show a notification with a message once the timer has finished a session.

```javascript
function showNotification() {
    agent.notifications.create(null, {
        type: 'basic',
        iconUrl: 'icons/128.png',
        title: 'Your time is up!!!',
        message: 'Please take a break!!!',
        requireInteraction: true,
    })
}
```

Future Development

Upon the completion of our user testing we received some feedback on what improvements could be made to our extension.

- It was suggested more than once that users should be able to track their usage of the extension over certain time periods, such as seeing how much time they have spent in the last days/week/month or how many breaks they have had as well. A full breakdown of the user testing can be found in the user testing report.

- We intend to make this extension available for mobile devices in the form of an application, this is due to the fact that many users spend a significant amount of screen time on their phones.

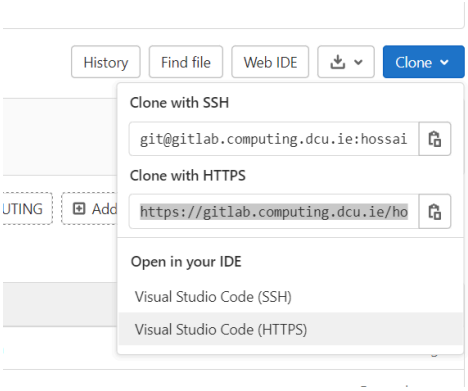
- We want to implement a tutorial upon the installation of the extension for users to get familiar with the usage of the application for newcomers.

- We would like to create the option for users to select whether or not the timer continues to run down while they have the selected browser minimised, as at the moment the timer runs down indefinitely unless it is paused or the browser is completely exited.

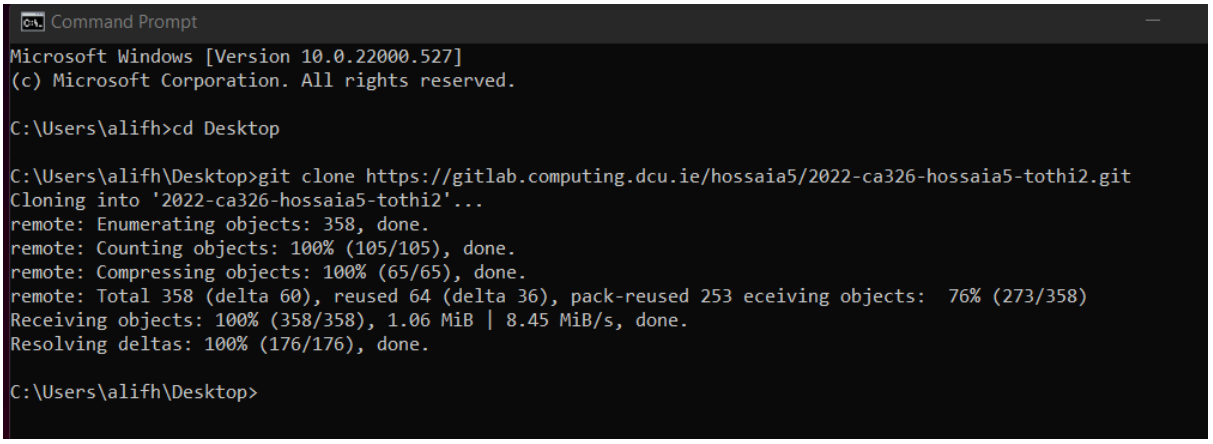
Installation Guide

Step 1: Go to the gitlab repository:
https://gitlab.computing.dcu.ie/hossaia5/2022-ca326-hossaia5-tothi2/-/tree/master/src

Step 2: Click the Clone button and use the clone with HTTPS.

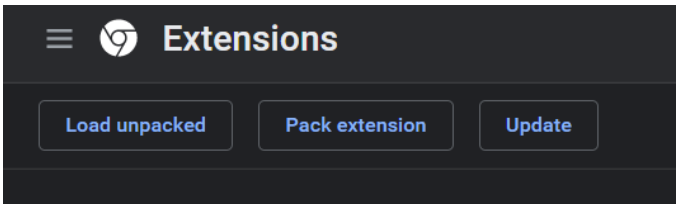


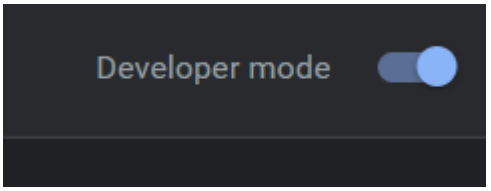
Step 3: Use git clone to download the web app by using command “git clone https://gitlab.computing.dcu.ie/hossaia5/2022-ca326-hossaia5-tothi2.git”
in your command prompt on windows or terminal on any unix based machine.



Step 4: Go to your preferred web browser(I'll use Google Chrome for this example). Click the settings button and use the extension button on the left.

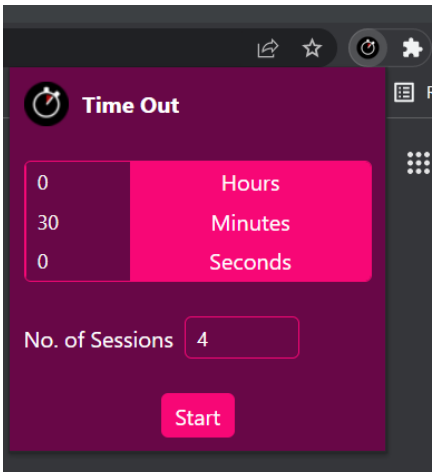
Step 5: Turn on developer mode and use load unpacked to select the "src" folder containing all the files for the extension. This should be the same folder you cloned from gitlab.





Step 6: Select the folder and then pin the extension icon. Then get rid of the settings page and continue browsing as normal.

Step 7: Lastly click on the extension icon and run the extension to your desired time and intervals.



Testing

Compatibility testing:
After creating a web extension that ran on Google Chrome we realised that not all users may necessarily use Chrome or may prefer another web browser. This caused us to make alterations to the code in order for it to run on multiple web browsers. The ones we have run it on and tested so far are Google Chrome, Mozilla Firefox, Brave and Microsoft Edge. For each of the browsers we had to make tweaks and changes in order for it to work effectively.

Integration Testing:
We also conducted a lot of integration testing as we had to build code that will run on different web browsers. Code for Google Chrome may not run on Mozilla Firefox and vice versa. And the same can be said for Brave and Microsoft Edge.

Also Integration testing comes into play when improvements or changes are made to the web extension and we have to manipulate the application for it to work effectively.

User Testing:
User testing was the most important form of testing we did for our project, as we made a working web extension we really needed other users to use the extension. We provided the user with the necessary guide and instructions needed to install and run the application on their own laptops/pc's. We also allowed them to use the extension for **over 24 hours** as it helped the user form a better opinion about the application. After the testing the user was asked to fill in a consent form and provide their feedback through a survey. We made sure to include key relevant questions that would help us gain a better understanding of what the users wanted from our product. Making sure to ask about the ease of use and new features the user would like for us to include or add. We provided option boxes for the user to check and space for feedback.

After collecting the feedback we learned a lot about how the users felt about the product and gained knowledge of areas where improvements can be made.

References:

Chrome Developers, 28th February 2014:
https://developer.chrome.com/docs/extensions/mv3/getstarted/

John Sonmez, 8th April 2015, Sitepoint:
https://www.sitepoint.com/create-chrome-extension-10-minutes-flat/

Alexander Zatkov, 19th April 2018, Sessionstack:
https://blog.sessionstack.com/how-javascript-works-under-the-hood-of-css-and-js-animations-how-to-optimize-their-performance-db0e79586216

Dvid Silva, 22nd October 2016, Hackernoon:
https://hackernoon.com/creating-popup-chrome-extensions-that-interact-with-the-dom-27b943978daf

Shahed Nasser, 24th April 2021, Personal blog:
https://blog.shahednasser.com/how-to-send-notifications-in-chrome-extensions/