SP-17 — Red Chrome Plugin

CS 4850 - Section 03 – Fall 2024

Sharon Perry

Nov. 14, 2024

Ashley Nestor
Team Lead

Devin Beck
Lead Developer

Website:

https://sp17-redchromeplugin.github.io/SP17-CatTracker/index.html

Github:

https://github.com/SP17-RedChromePlugin/SP17-CatTracker/tree/main

Stats:

Lines of Code	No. Components	Total Man Hours
1081	23	146

Table of Contents

Introduction

Overview

In an era where everyone is online more often than ever, it can be incredibly difficult to manage the amount of time one spends online. Because of this, limiting time spent on social media platforms and other attention-hogging websites is vital to our health. Our solution: a cat companion that lives within your browser.

The CatTracker is a web solution that tracks the time its users spend online and provides a convenient reminder system to keep track of important events and activities. The CatTracker lives as a downloadable Chrome extension through the Chrome Web Store utilizing HTML, CSS, and JavaScript as a code base. It can be managed through the built-in settings menu along with Chrome's Extension Manager.

Primary Goals

The CatTracker provides a cute and lightweight Chrome Extension to track websites that the user visits and provides a simple reminder system. It provides insight into what the user spends their time on online as well as keeps track of top websites. The reminder system provides simple editable reminders that the displayed cat will inform the user of by meowing. The extension uses very little storage and processing power in order to be as lightweight and easy for anyone to run as possible. The cat itself will display simple and cute animations so it feels more life-like.

Project Discussion

The CatTracker started as an idea about a Chrome Extension interactive cat, somewhat similar to a virtual pet. The team concluded that another, more functional feature would be important for marketability. The feature decided on was a reminder system, much like a cat would never let its owner forget meal time, our virtual cat wouldn't let its owner forget important tasks.

From there the team started work on creating the cat and defining its major features: reminders, interaction, and the menu with plans to expand these features to: cat personality and connection with Google Calendar. It was at this point when the team met with our advisor who gave us an excellent idea: time tracking. Time tracking is a feature built into phones, but what about web browsers? This marked a huge shift in our focus and became the first feature other than the visuals that we implemented.

The cat was born as a single sitting cat png file that opened a text box when clicked. Then the team implemented a menu system where the open menu icon would show on hover. Clicking this button would open up the settings and calendar button, both of which would send a message to the console for verification. The next thing the team worked on was the menu functionality that was revisited several times for more menus and libraries to properly convey the information we wanted the user to have access to. This marked the completion of the time tracking feature.

The team then shifted our focus to the cat animation feature. After the challenges of animating the menu opening and the struggles of formatting the displays, we decided that our original plans of having complex cat interaction and behavior would be too difficult and take away from the time tracking and reminder features. Thus, we decided on a simple timeout to change the animations. The other cut we made to the animations was to limit it to three simple to implement behaviors: sitting, walking, and sleeping. These were relatively simple to implement, but added a whole new dimension and liveliness to our cat.

Finally, the team focused on one of our original features: the reminder system. We decided to keep it simple and allowed the user to enter a name for the alarm, the date of the alarm, and the time of the alarm. The alarm would then inform the user through a text bubble and play a meow. The alarms can be edited and deleted freely for ease of user access.

Throughout the development process the team worked on several smaller features including: extension scaling, happy cat on hover, and meowing. All of these features were not originally planned on in our requirements, but fit in with our other features well and were simple to implement. Features such as these add life to our cat and care to our project as a whole.

Some notable challenges faced included: how best to display data, communication between the Chrome API and our interface, and how best to edit reminders. The data display required new libraries to be added to best display all of the collected data. We also encountered some communication difficulties within our code with the Chrome API in order to update the reminders. The reminders had to take data from the display and send it back into the div where it would be sent to the library to store and then sent back to the display. These challenges slowed down development significantly, but overall the team is happy with what we accomplished in spite of these challenges.

Project Plan

Project Website

The CatTracker currently has a website that can be found at: https://sp17-redchromeplugin.github.io/SP17-CatTracker/index.html. The website gives a general overview of the CatTracker, as seen in the Introduction, as well as contains the major documentation and video overview of the CatTracker.

Deliverables

Due Dates:

- SRS - 9/1
- SDD - 9/1
- Project Plan - 9/1
- Working Prototype - 10/29
- Final Report Draft - 11/14
- Final Report - 12/2

Deliverables:

- Team/Project Selection document (Individual Assignment)
- Weekly Activity Reports (WARs – Individual Assignment)
- Peer Reviews (Individual Assignment)
- Project Plan (Group Assignment)
- SRS, SDD, STP & Dev Doc (Group Assignment)
- Present Prototype for Peer Review (Group Assignment)
- Final Report Package (Group Assignment)
    - Final Report (Group Assignment)
    - Source Code (Group Assignment)
    - Website (Group Assignment)
    - Video Demo (Group Assignment)
    - Chrome Plugin Download on GitHub

Team Communication

Our team's communication with one another was done primarily through Discord. Each week on Mondays, the team meets for a minimum of 15 minutes to discuss project progress, upcoming deadlines, and plans for the next week. These meetings have been critical for assessing progress, keeping up with deadlines, and for the communication of the team as a whole.

Our team's communication with our advisor was done through Teams prior to important deadlines such as before the SRS, SDD, Project Plan, and Final Report Draft deadlines. These meetings last 15 minutes and serve as a way for our team as a whole to communicate directly with our project advisor about our current progress along with how close our team is to its target goals.

Project Schedule

As shown in Figure 1. our team has a projected total work hours of 146 over the course of the project's duration of 8/25 through 12/1. The majority of the man hours occurred from 9/26 through 11/3 when the major requirements were being implemented along with the major documentation updates happening for other related deliverables. Our team averaged 10 hours per week worked on the CatTracker with a minimum of 3 hours to a maximum of 16 hours.

					SRS + SDD + Project Plan Due		*Project Prototype Complete*						*Review Requirements and Documents*				C-Day & Final Report		
Phase	**Tasks**	**Complete%**	**Current Status M**	**Assigned To**	**8/25**	**9/1**	**9/8**	**9/15**	**9/22**	**9/29**	**10/6**	**10/13**	**10/20**	**10/27**	**11/3**	**11/10**	**11/17**	**11/24**	**12/1**
Requirements	SRS	100%		Ashley	1	3													
	SDD	100%		Ashley		2													
	Project Plan	100%		Ashley	2	2													
	Meet with Advisor	66%		Ashley, Devin		1		1									1		
	C-Day Application	N/A		Ashley, Devin															
Project design	Chrome Extension Development	100%		Ashley, Devin			5	4											
	Menu Design	85%		Devin				2	4	2									
	Cat Functionality	85%		Devin					2	4	2	2							
	Reminder Functionality	85%		Devin					2	2				2	4				
	Additional Functions	50%		Devin						2	4		2	2		2			
	Develop working prototype	70%		Devin					2	2	6	2							
	Test prototype	70%		Ashley, Devin							2	4	2		2	3	3		
Development	Review prototype design	60%		Ashley, Devin							2	2	5						
	Rework requirements	85%		Ashley								2	3	5					
	Document updated design	85%		Ashley										4	2				
	Test product	80%		Ashley, Devin										2	3	4			
Final report	Presentation preparation	70%		Ashley, Devin										4	5	4	4		
	Poster preparation	N/A		Ashley, Devin															
	Final report submission to D2L	25%		Ashley, Devin														5	3
			Total work hours	**149**	**3**	**8**	**5**	**7**	**10**	**12**	**16**	**12**	**12**	**19**	**16**	**13**	**8**	**5**	**3**

Figure 1. Team Red Chrome Plugin's currently estimated gantt chart of man hours.

Version Control

Version control was maintained through GitHub branches. These branches can be split into three categories: the main CatTracker code branch, individual team members' code branches, and the website branch.

The main CatTracker branch takes code in from the individual team members' branches and all code published is reviewed for its functionality and readability. Code pushed to main must not stray from the major design considerations: readability, data privacy, and being lightweight. Additionally, the code is reviewed for general readability as well as detailed comments. Our team's dedication to readability has led to an overall smoothness in communication and clarity in our code.

Each team member has their own individual code branches where their changes to the main code branch are published. These were used to save code and keep track of changes before committing new code to main. The main code branch would be updated after notice of changes needing to be pushed to main were brought up in our weekly team meetings.

The website branch was created solely in order to use GitHub's built in website hosting system, GitHub Pages. The branch serves as a base for the website to be built off of. Changes made are done by all members of the team directly in the branch. The same code principles still apply, but are less strictly monitored than the main branch.

Requirements

Design Constraints

Product Environment

The CatTracker is a Google Chrome extension built to operate within the Google Chrome browser. It can be accessed by any operating system that can run Google Chrome and has the capacity to download a ZIP file from GitHub. The CatTracker is designed to be lightweight and run using Chrome's pre-existing resources on most Chrome web pages.

User Characteristics

Users of the CatTracker are cat-lovers who want a lightweight solution for managing their time. They are, at minimum, tech savvy enough to know how to download the GitHub files and use Chrome Extension Manager to unpack and load the files as a Chrome Extension. Users are required to be able to read English as there are no current plans to localize the CatTracker. Some functionalities will be understandable without knowledge of English based on the simplicity of the menus and features.

Users who like cats will enjoy the different behaviors of the cat. Different cat behaviors occur randomly with hand-drawn animations and images displayed. The user can even mouse over the cat which updates the image of the cat to a happy cat. These users will not use as many of the time management features. A slight drawback for these users would be that the time management features would still run in the background, despite the user not interacting with them or the data gathered in the slightest.

Users who want to manage their time more effectively will enjoy the customizable reminders and the time tracking features that the CatTracker provides. The reminders are easy to see and edit which is highly desired by this type of user. Additionally, the website time tracking provides new insights that are not available by default on computers as well as an easy way to view this data.

Operating Environment

The CatTracker functions as a Google Chrome Extension accessible through download from our team's GitHub repository. The app can be accessed by any operating system that can install and run Google Chrome. The hardware requirements, as a result of this, match that of Chrome's system requirements. No other systems or programs other than Chrome are required to download and run the CatTracker.

Chrome's requirements are different per OS and the full list of requirements can be found here: https://support.google.com/chrome/a/answer/7100626?hl=en. The basic Windows requirements are: Windows 10 or later and an Intel 4 processor or better. The basic Mac requirement is macOS Catalina 10.15 or better. The basic Linux requirements are: 64-bit Ubuntu 18.04+, Debian 10+, openSUSE 15.5+, or Fedora Linux 39+ and an Intel 4 processor or better. The Android requirement is: Android 8.0 Oreo.

Product Scope

The CatTracker tracks the time its users spend on individual websites, allows the user to set, edit, and delete reminders, and has cat animations for different cat behaviors. It will be free to download for users who know how to download and run their own Chrome extensions. It will track the time users spend on all websites and keep this data for display purposes. This extension also displays a cat image that may be distracting and obscure some of the user's screen. This display is toggleable and the user can choose which menus are displayed at will.

This extension will not track user data beyond a week, it will not allow the user to control the displayed Cat Animations (beyond halting the current animation), nor be available on the Chrome Web Store. The data stored by the CatTracker is only stored for a week and has no option to store data longer. Websites, along with the time the user spends, are stored for that long in Chrome's internal storage. The user has no option to control the animations the cat displays other than stopping the currently playing animation on user click. Finally, the CatTracker is only available through our GitHub as a download and not through the Chrome Web Store.

System Features

Cat Interaction

Description

The Cat Interaction features include the on hover and on click effects. These allow the user to interact with the cat after the aforementioned action is taken by the user.

The on hover Cat Interaction activates only when the cat is performing the Sitting Cat Animation. When the user hovers over the cat and the cat is Sitting, the cat updates its image to the smiling cat image for the duration that the mouse is hovering over the cat. When any other animation: Sleeping or Walking, is occurring then only the secondary function, the open menus button will be displayed.

The on hover Cat Interaction has one other important function, displaying the button to open the Statistics Menu and Settings Menu. For the duration of the time the user has the mouse hovered over the cat, the open menus arrow shown in Figure 2. will be displayed.

Figure 2. The Open Menu button.

The on click Cat Interaction activates upon user click of the cat image. This interaction stops all current animations and opens up a text box. This provides the user agency over what the animations the cat is performing. The text box is merely a cute meow that informs the user that the cat is functional or other tidbits of information.

Response Sequences

When the user hovers over the cat the Open Menu button will be displayed. If the cat is currently displaying the Sitting Cat Animation, the on hover has an additional effect of changing the cat's image into the smiling cat image as shown in Figure 3.

Figure 3. The Cat Smiling image.

When the user clicks on the cat the cat will stop all running animations and particles. This will default the cat image back to the Sitting Cat animation and stop all movement. Additionally, the cat will display a text box informing the user of tidbits of information or a simple 'meow' from the cat. These tidbits of information include top visited websites for the day or other similar statistics.

Functional Requirements

This requirement requires the cat to have its own menu operations for the specified user actions. It requires control over the animation player and the images to update the specified UI elements to stop or open on command. The Open Menu button must be attached to the cat UI elements to move together and stop moving together so that consistency is kept.

Cat Animation

Description

The Cat Animation features include: cat sitting, sleeping, walking, and meowing animations. They play according to a timeout function running in the background when the CatTracker is live on a browser. The animations occur randomly without user input and control the UI cat element. The animations can be stopped by user action, see Cat Interaction on page 10 for more detail.

Response Sequences

The Cat Animations cannot be controlled by the user but they can be halted on user click. If the user clicks on the cat, all animation, movement, and particles cease.

In the background code of the CatTracker, a timeout function controls the Cat Animations. The cat sitting animation is the base for all other animations. It is the default animation that allows for the most user control over the cat's animation. It is the animation that plays once the user clicks on the cat to end the current animation. The image itself can be seen in Figure 3.

Figure 3. The Cat Sitting animation image.

The cat sitting animation updates the cat's image to the cat sitting image and halts all movement. The sitting animation comes with a small chance for the cat to 'meow'. The meow animation opens a text box with 'meow' as the text and plays the cat meow animation.

The cat walking animation walks in a random direction (left or right) and flips the image and movement direction when the cat hits the edge of a screen. It utilizes a random number

generator to choose the direction of movement on startup. Additionally, the position of the cat needed to be kept after movement which required additional variables to be introduced to the cat. The cat automatically detects when it gets too close to a side of the screen to change directions which causes some issues on certain screens when resized. The actual animation can be seen in Figure 5.

Figure 5. The Cat Walking animation.

The cat sleeping animation halts all cat movement and displays the cat sleeping GIF as shown in Figure 6.

Figure 6. The Cat Sleeping animation.

The cat sleeping animation has an additional particle creation and deletion process. The image shown in Figure 7. is created for the duration of the Cat Sleeping animation. These particles move upwards while lowering in opacity until they reach the given threshold and are removed for storage purposes.

Figure 7. The sleeping ‘Z’ particle created in the Cat Sleeping animation.

Functional Requirements

Each cat animation had their own function along with an additional controller function that called the specific animation functions after the timeout occurred. Each animation required their own PNG file or GIF animation to be created. The sleeping animation in particular required a particle creator as well as remover. Each animation had to be cancellable in order to align with other requirements

Cat Menu

Description

The Cat Menu features include: access buttons, time tracking data, reminders, a UI scale slider, and the capacity to delete user data. These menus can be accessed through the cat’s connected Open Menu button. Each of the created buttons opens up its own respective menu. The specific functionalities of the time tracking and requirements will be expanded upon in the Time Tracking requirements section on page 16 while the Reminders Requirements on page 16.

Response Sequences

The Statistics Menu button opens the Statistics Menu when on click. The button itself appears once the Open Menus button has been pressed and the PNG image of the button can be seen in Figure 8. The resulting menu can be closed with another press of the Statistics Menu button. More information on the Statistics menu can be found in the Time Tracking requirements on page 17.

Figure 8. The Statistics Menu button.

The Settings and Reminders button opens the Settings and Reminders menu on click. If pressed a second time, the resulting menu is closed. The PNG image of the Settings and Reminders button can be seen in Figure 9. Further information about the Reminder functionality in the Settings and Reminders menu can be found in the Reminder Requirements on page 17.

Figure 9. The Settings and Reminders button.

Functional Requirements

The Statistics Menu allows the user to see stored data through the Time Tracking feature. It displays information like top visited websites, the most recent week's data, and other assorted graphs. The Settings and Reminders Menu allows the user to rescale the extension, delete all stored data, and all of the reminder features.

Time Tracking

Description

The Time Tracking features include data visibility and storage. It displays the most recent week of rolling data and the data for the present day. It has sections for the top visited websites, a weekly breakdown, and time spent by website charts. The data displays are created through chart.js while the rest of the information is through text.

Response Sequences

The Statistics Menu is clickable and draggable on a user's Chrome tab. It can be scrolled using the middle mouse scroll wheel or through the slider on the right of the menu. The cute cat menu display can be seen in Figure 10. The data shown on the charts can be clicked on to see different visions of the same data.

Figure 10. Sample data shown in the Statistics Menu.

Functional Requirements

The Statistics Menu has its own PNG image display that is the background for the informational text and graph displays. These elements must be linked in order to be draggable and resizable by the user. There are data storage and display intricacies that must be upheld in order to convey real-time data.

Reminder Feature

Description

The reminder functionality includes setting reminders, editing reminders, deleting reminders, and an audio-visual alert. The Settings and Reminders Menu allows the user to input a name, time, and date for the reminder. Once a reminder is added, it is listed at the top of the menu. The reminders created can then be edited or deleted through a button. The alert is handled by the cat display.

Response Sequences

Reminders can be set, edited, and deleted through the Settings and Reminders Menu. A view of this menu can be seen in Figure 11. Users are prompted with an input box for a reminder name, hour, minute, AM/PM, and calendar date for a reminder to be set. Reminders can be edited by entering in their name along with a new time and date for them to occur at. Reminders can be deleted through the specific delete reminders button shown along with the reminders display at the top of the menu.

This menu, similarly to the Statistics Menu, can be clicked and dragged around the Chrome browser at will. The menu can be closed with the same button that opened the menu.

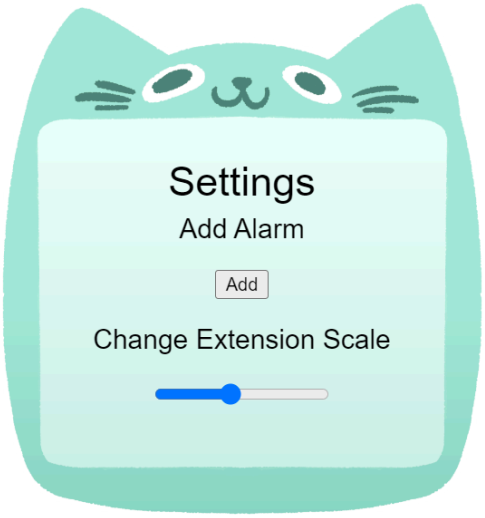

Figure 11. Example of the Settings and Reminders Menu.

Functional Requirements

The Settings and Reminders Menu stores its reminder data in Chrome's built-in local user data storage. It is required to communicate directly with Chrome's background service worker that handles the data storage.

Quality Attributes

Performance

The CatTracker is designed to be a lightweight Chrome Extension that utilizes Chrome's built-in data storage as a storage hub for user data. After some testing, it was shown to have little to no impact on CPU usage nor system memory. Testing was performed by checking the CPU usage and the memory usage without the CatTracker installed, then installed but not active, and finally with the extension active. After these tests, the CPU usage and memory usage fluctuated more with the machine itself rather than with the installation or enabling of the extension. Therefore, the CatTracker has minimal influence on system resources.

Data Storage

The CatTracker stores a rolling week of user data. It utilizes Chrome's local user data storage in order to do so. This keeps user data local to a Chrome account and uses Chrome's data security protocols. Data is only stored for a week at a time to minimize the storage required to run the CatTracker.

Security

Data security was one of the CatTracker team's major concerns. Our solution was by using Chrome's built-in local data storage. Chrome's data security was much more than the team could have created with double the total project time. It ended up causing some data communication difficulties between our data display library, chart.js, but the security tradeoff for the communication difficulties were worth the struggle. Data is local and not kept past a week of collection time which provides more recent data and more secure data.

Software Development

Design Considerations

Some general design considerations that the CatTracker team considered were: how to publish the application, how to keep user data secure, browser limitations, a limited time period, and no prior knowledge of Chrome extensions. These concerns came with their own set of issues and developmental struggles.

Some of the developmental struggles include: the limited time period and minimal prior Chrome extension developmental experience. The team took the first couple of working weeks in order to familiarize ourselves with HTML, CSS, and building Chrome extensions as a whole. These couple of weeks took development time, but were critical components in ensuring future extension building progress went smoothly and promoted good habits within the development team. The short time period, being a sixteen week span, drastically limited our scope for the project as a whole. Some requirements we initially wanted to implement had to be cut early on due to a lack of development time. It made the team focus on honing the critical components of the CatTracker and left little time for in-depth testing.

Some physical constraints included: how to publish the extension, how to keep user data secure, and certain browser limitations. The team initially planned to publish the extension on the Chrome Web Store, but were met with certain challenges. Registering as a developer capable of publishing an extension to the store came with monetary costs as well as new security concerns. Having the CatTracker publicly available meant that anyone could download and reupload a doppelganger of the CatTracker with malicious intent. As a result of these concerns, the team decided it would be simpler to leave the CatTracker as an application targeted towards those users tech savvy enough to download and run the extension from a GitHub.

Other physical constraints included keeping user data private. User data privacy was a major topic of concern for the team and was heavily researched before finding the current solution. The solution settled on was to use Chrome's built-in data storage and security. The time limitations for the project left the team with little choice to design a more secure solution. The only sacrifice would be Chrome's resources for the most data security we could offer to users.

System Architecture

The software components of the CatTracker program communicate with Chrome's API, a background service worker, and Chrome's local data storage through a front-end handler in order to provide the cute cat UI directly to the user's system. Chrome's API and the currently impacted page act as a baseline for our front-end handler to provide the UI.

As shown in Diagram 1., There is a triangular pattern of communication between Chrome's API, our front-end handler code, and Chrome's background service worker. This is as a result of the communication being blocked by Chrome between our library storage and display needing multiple communication and update channels.

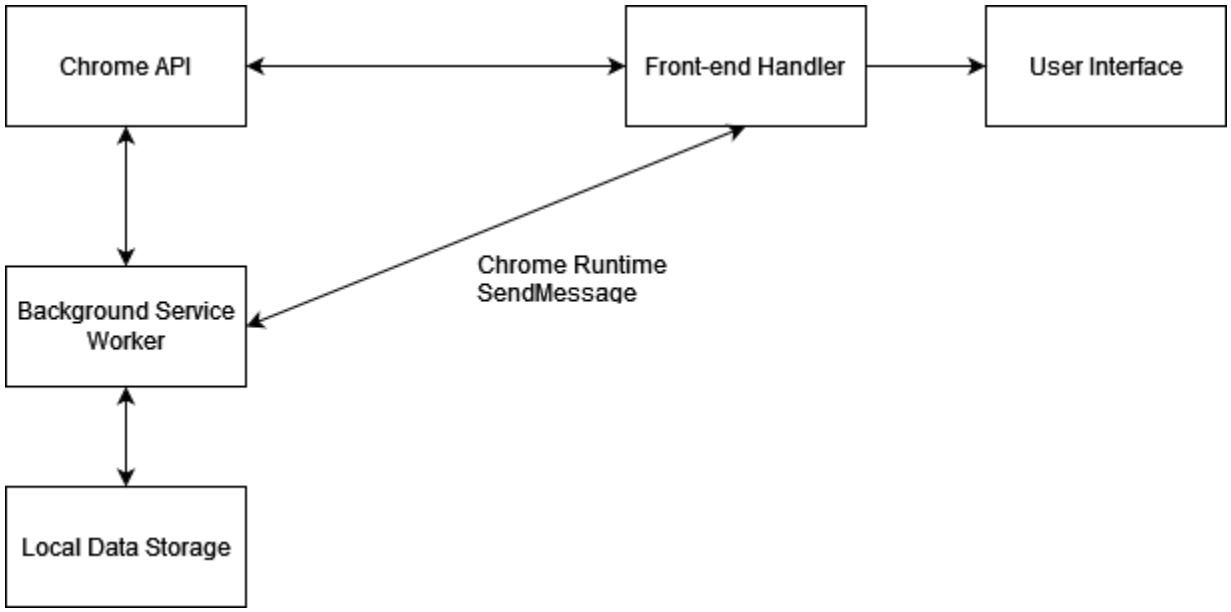

Diagram 1. System communication within the CatTracker Extension.

In Diagram 1. it can be seen how the two major sections of the CatTracker, Chrome's side and our team's code, coincide to provide functionality. Our team's code works as the front-end handler to provide the UI while Chrome's side provides the base API, the background service worker, and the local data storage. The communication between our front-end handler and Chrome's background service worker is performed through Chrome runtime's SendMessage. This feature allows our libraries to communicate with the stored data and sent back to our UI for display purposes. This communication is convoluted because our data is stored directly in Chrome for security purposes while our library that controls the displays cannot talk directly to Chrome's API.

The team needed to work around Chrome's API to use it for the background service work and data storage while being separate from Chrome's restrictions. We had to find a way to have our own styling without being impacted by the pages the CatTracker was running on. This was solved through a full screen div element layered on top of the website injected into the website as a shadowroot. The shadowroot in the front-end handler allowed us to prevent the problematic CSS styling overlaps.

Detailed System Design

The core system design elements include the CatTracker's many UIs, shadowroot, and graph display library. The CatTracker uses a full screen div/shadow box in order to keep its UI styled as specified in our team's code rather than taking the CSS styling of the Chrome tab that the CatTracker is running on. This div provides the environment for our cat display and contains the cat image and its' connected UI elements.

The UI system design elements all run through the aforementioned shadow box. A full list of our UI elements include: the cat itself, the cat's attached buttons, the settings menu, and the time tracking menu. The cat is the root for all of our UI elements and opens up the rest of the menus on user click. The initial click opens up the pathway to the time tracking menu and the settings menu buttons. Each of these buttons opens up their own UI respectively.

The time tracking menu is a UI that displays the user's top visited websites, tracks time spent online for the last rolling week, and displays the time spent on all individual websites. It uses the chart.js library to display the stored information. Chart.js provides all of the graph display functionality which enhances the effectiveness of the UI as a whole. The storage of this data and the communication between the chart.js library and our data in Chrome was a challenge due to the communication limitations. It takes more time than we would like for the extension to load, but the alternate solutions were much more laborious for a drastic decrease in data security.

The settings menu is a UI that allows the user to set, edit, and delete reminders; rescale the shadow box elements; and delete user stored data. The reminder functionalities did not take additional elements to work, and neither did the shadow box rescaling. These elements were relatively simple to implement for the critical functionality they provided. The stored user data required communication between the chart.js library display and Chrome's internal user data storage.

Two major constraints included the difficulty in data communication and the extension's ability to work on all screen sizes. The data communication was difficult between the chart.js library and where the actual data was stored in Chrome. This is expanded upon in the System Architecture on pages 21 of this document. Being an extension built for any system capable of running Chrome meant that we had to consider different screen resolutions and the possibility of the user windowing the program. This was mitigated through using screen window sizings rather than pixel sizes as well as our slider to allow the user to manually change the UI sizes on their own device.

Development Process

Development Considerations and Progress

- Design Considerations
    - Security
        - Utilizing Chrome's built-in data storage for data privacy
        - Data stored locally
        - Not publishing to Chrome Web Store to avoid doppelgangers
    - Publishing
        - Published through GitHub as a ZIP file
        - To run:
            - Download ZIP file from GitHub
            - Locate Chrome's extension manager
            - Load unpacked ZIP file
            - Allow extension to view or edit data
            - Enable extension
- System architecture
    - Data Storage: built-in Chrome local data storage (as JSON)
    - Operating Environment: Chrome Web Browser
    - Minimum System Requirements: Same as Chrome's
- Cat injection
    - Full screen div is layered on enabled website, in front of content
    - Cat elements live inside of div
    - Div injected as shadowroot to avoid CSS styling overlaps
- Tools used
    - Github version control
        - Team members have individual branches for work
        - Website has own branch for styling and modularity
    - Visual Studio Code
        - HTML, CSS, JavaScript code editor
        - Used in conjunction with GitHub's desktop manager
    - Chrome Extension Manager
        - Uploading extension
        - How updates and testing were performed
    - Krita
        - Art program
        - Used to create all cat images and gifs
            - Art created for each requirement or feature added
    - Chart.js

        - Used for easy data visualization
        - Bar charts and pie graphs were utilized for displaying user data
- Implementation of major requirements
    - Cat behavior
        - Coded through a separate channel than from the time tracking
        - Uses a timeout interval to run animations
        - Allow for user click to stop all currently running movement, change image back to sitting image, and remove any particles
        - Sitting behavior:
            - Change image to sitting
        - Walking behavior:
            - Change image to walking gif
            - Randomly choose move direction
            - Move shadowroot, cat image, and everything else together
            - If cat hits the edge of the screen, flip image and inverse movement vector
        - Sleeping behavior:
            - Change image to sleeping gif
            - Create ‘Z’ particles
            - Move ‘Z’ particles upwards and have them slowly fade out
            - Remove ‘Z’ particles after a timeout to save resources
    - Website Tracking
        - Background service worker script
        - Data stored in dictionaries
            - tabDomains stores any active tab with a unique ID as the key and the name of the domain open as the value.
                - If a new domain is opened on the same tab, time spent on the previous domain is recorded and the value of the tab ID is updated for the new domain.
                - If the tab is closed, the time spent on the domain is recorded and the key is deleted.
            - tabStartTimes stores any active tab with a unique ID as the key and the date and time the domain was accessed as the value.
                - Works in conjunction with the previous dictionary to associate tab ID, domain, and time first accessed.
            - totalTime stores visited domain names as the key and the total time spent on the domain as the value.
            - totalTimeEachDay stores seven keys for each day of the week, and their associated totalTime dictionary as the value.
        - Weekly data storage as a rolling queue

            - Stores minutes spent online each day of the most recent week
    - Reminders
        - Background script stores reminders by name and date attribute
        - When date and time of the stored reminder has passed, a message is sent to the foreground script to display the reminder to the user.

Libraries Used

- Chart.js

Chrome Plugin Basics

- Built Using HTML, CSS, and JavaScript
- Basic Files:
    - Manifest.json file: contains extension's main configuration, name, description, and base actions
    - overlay.html file: holds the html code to inject into the website
    - addDiv.js file: script that runs in the foreground to inject overlay.html into the website
    - background.js file: tracks time spent on websites and reminders in the background, regardless of if the extension is active or not
- Extension Loading:
    - Go to the Chrome extensions manager
    - Enable developer mode
    - Click load unpacked and select the directory of the extension
    - Turn on the extension and allow it to view site data

Testing

After developing the CatTracker, the team reviewed our requirements and determined the most critical requirement categories. These categories included the: Chrome extension connection requirements, cat behavior requirements, cat menu requirements, time tracking requirements, and reminders requirements. Each category was expanded to list each of the functional requirements from the most critical to the most minute.

Each requirement in their category was tested as a pass/fail condition. If the requirement failed, then it was noted as such and assigned a severity. These severities were as follows, from most important to fix to least: critical, high, medium, low. Critical severity for a requirement would describe a requirement that was necessary for normal function. Low severity, on the other hand, would describe a requirement that most likely would not be completed before development time concluded.

Chrome Extension Connection Requirements

Requirement	Pass	Fail	Severity
Chrome Extension can be uploaded to the extension manager	x		
Extension runs	x		
Custom Icon	x		
Extension named	x		
Asks for permission to track User data	x		
Connects with the User’s Google Calendar		x	low

Cat Behavior Requirements

Requirement	Pass	Fail	Severity
Clicking on Cat stops all animations	x		
Cat changes animations based on a timeout	x		
Cat walks	x		
Cat sleeps	x		
Cat sleep animation creates ‘Z’ particles	x		
Cat sleep animation removes ‘Z’ particles	x		
Cat sits	x		
On hover, Cat displays happy image	x		
Cat meows randomly when sitting	x		
Cat can be pet		x	medium
Cat can be fed		x	low
Cat can be played with		x	low
Cat personality		x	low

Cat Menu Requirements

Requirement	Pass	Fail	Severity
Cat can be clicked on to access menus	x		
Time Tracking icon	x		
Specific reminders icon		x	low
Settings icon	x		
Extension scale can be changed	x		
Extension data can be deleted	x		

Time Tracking Requirements

Requirement	Pass	Fail	Severity
Displays most visited websites	x		
Rolling time for past week is tracked	x		
Rolling time for past week is displayed in a graph	x		
Rolling time for past week is displayed in a pie chart	x		
Allows for users to customize graphs displayed		x	medium

Reminders Requirements

Requirement	Pass	Fail	Severity
Reminders can be set	x		
Reminders can be deleted	x		
Reminders alert the user at the specified time	x		
Reminders can be edited	x		

Summary

To summarize, the CatTracker is a project created for the CS 4850 course with the intention of adding a cute virtual companion to one's browser that would help them track their website history in easy-to-understand visuals and set reminders. Though the initial scope of the project was smaller, it grew as time went on. What started as a static image became a cat companion with multiple states to cycle through and multiple screens to showcase web browsing data and alarm systems, including settings that help the user make use of the extension more easily.

The project was designed around two main systems that would control the logic of the extension: the front-end handler and the background service worker. The front-end handler would be in charge of all visual elements and cat states, updating the html and layering it on top of any given page. The background service worker would track website usage from the user and handle data saving and alarms, regardless of if the extension was currently active or not. These two components spoke to each other through Chrome's messaging API and worked to allow the extension to be what it is.

In the end, our team was able to complete the vast majority of the initial goals we set out for our project as well as adding extra features that flesh out the extension as a whole. We stayed relatively consistent with the amount of time we poured into development and documentation, splitting the work evenly within. When time permitted, we were able to create smaller details like allowing the cat to meow and resizability to improve the user experience of the extension. The features that we were not able to get done in time were mainly around being able to interact with the cat, which we decided was ultimately secondary to time tracking, data management, and alarm features. The extension is currently in a functional state with no known fatal errors, with completed features working on any Chrome browser.

Appendix

- **UI:** User Interface, the visual link between the user and the logic of the program.
- **API:** Application Programming Interface, a set of rules that allows programs to communicate with each other.
- **Chrome Extension:** A program that runs in conjunction with the Chrome browser and created by third party users of the browser.
- **Background Service Worker:** A javascript file that Chrome will run in the background of the browser, regardless of activation.
- **Manifest:** A JSON file required by Chrome that lists all permissions and files included within a Chrome Extension.
- **Div:** HTML divider element that encompasses a specified section of the screen for convenient portioning.
- **Chrome Runtime:** A subset of the Chrome extension API that allows for portions of the extension to communicate with Chrome and each other.
- **Shadowroot:** A container used in web development to isolate portions of code from the rest of the page.

Works Cited

- Google. “Chrome Browser System Requirements - Google Chrome Enterprise Help.” *Google.com*, 2012, support.google.com/chrome/a/answer/7100626?hl=en.
- “Extensions / Get Started.” *Chrome for Developers*, Google, developer.chrome.com/docs/extensions/get-started.
- chart.js. “Chart.js | Open Source HTML5 Charts for Your Website.” *Chartjs.org*, 2019, www.chartjs.org/.
- “API Reference.” *Chrome for Developers*, Google, developer.chrome.com/docs/extensions/reference/api.
- “Software Design Document (SDD).” *Www.nuclino.com*, www.nuclino.com/articles/software-design-document.
- “Tips for Optimizing Chrome Extensions.” *Google.com*, Google Groups, 2024, groups.google.com/a/chromium.org/g/chromium-extensions/c/jNNksFBpa3A?pli=1. Accessed 14 Nov. 2024.