Missouri State University

Department computer Science

Personal Study Assistant

Trey Alexander
Mackensie Bass
Ali Karimiafshar
Anh Minh Nhat Doan
Bryan Leach

Dr. Belkouche
YassineBelkhouche@missouristate.edu

April 27, 2022

Contents

1 Project description

1.1 Description

Google Chrome Extension to help students study by minimizing distractions from other websites. This is inspired by The Pomodoro Technique which recommendes studying for 25 minutes, then a 5 minute break, repeating that 3 times and then having a longer break. During study time, the user will only be allowed to access approved websites such as Google Classroom. This will be handled through a whitelist of approved websites. During the break time, the user is allowed to access any website they desire, but they are encouraged to practice healthy activites like stretching or drinking water. The user's keyboard and mouse inputs will be monitored to measure if they are actively working.

2 Requirements Specification

2.1 Functional Requirements

Req-01: On the first run, get information from the user.

- Description: The function makes the user provide their chosen username and a parent pin.
- Rationale: Student provides information to personalize the experience. Parent pin allows for access to control the whitelist.
- Input(s): Input information about the user and the chosen parent pin.
- Output(s): N/A.
- Dependency: N/A.

Req-02: Users can choose from a list of timer options when the extension is started.

- Description: The function lets users choose from a list of timer options. Example options would be A- 25 min study, 5-minute break, 3 times, 15 min long break or B- 45 minute study, 10 minute break, 3 times, 30 minute long break.
- Rationale: User might want to study for different amounts of time different days so this lets them choose.
- Input(s): Function receive option result from users.

- Output(s): Begin chosen session type.
- Dependency: N/A.

Req-03: There is a Timer to keep track of study and break time.

- Description: A timer is running to keep track of users' study time and break time. Timer can be paused, resumed, and reset by the user and system. Change color in study mode and break mode. Can be hidden or unhidden by the user.
- Rationale: It is easy to loose track of time or think that a lot more time has passed than really did while studying. A timer allows the user to know exactly how long they have been working, and how much longer they have.
- Input(s): Amount of time to start at, values for pause , resume, and hide the timer.
- Output(s): Send an alert when timer reaches 00:00.
- Dependency: Timer amount from Req-02.

Req-04: Keyboard and mouse interactions are monitored to detect if the user has stopped working.

- Description: User's keyboard and mouse inputs are monitored on an occurrence basis to ensure the user has not stopped working while not recording the contents of such inputs for privacy.
- Rationale: If a specified period of time elapses with no input from the user detected, it is likely the user has stopped working and measures must be take to help the user stay focused.
- Input(s): Mouse clicks and movements, as well as keyboard keystrokes.
- Output(s): Req-05 is called.
- Dependency: N/A.

Req-05: If no user inputs detected according to Req-04 after a predetermined amount of time, alert the user with a prompt to verify they are still working.

- Description: When the conditions of Req-04 are met, the user is prompted to verify they are still working. The study timer is paused until the user verification is complete.

- Rationale: Asking for user interaction to verify they are still working encourages more engagement with the material and ensures the user is present and attentive during the study times.
- Input(s): Req-04.
- Output(s): Prompt to user in the form of a pop up, or other contents.
- Dependency: Req-04.

Req-06: During study time, check if the Google Chrome browser is in focus.

- Description: The function checks if the browser window is in focus every few seconds. If it is found not to be, then the value is changed to false.
- Rationale: If the Google Chrome browser window is not in focus during study time, then the user is not studying so the extension needs to get them back on task.
- Input(s): N/A.
- Output(s): The in-focus value is changed to false, and Req-07 is called.
- Dependency: N/A.

Req-07: During study time, send an alarm and alert to the user if the Google Chrome browser is not in focus, and pause the timer.

- Description: During study time, an alarm will sound when the extension detects the Google Chrome browser is not the in-focus window. When the user comes back to the page, an alert will show explaining the alarm sounded because the browser window was out of focus.
- Rationale: The purpose of the alarm and alert is to try to prevent the user from using other applications on the computer when they should be studying. The extension cannot prevent other applications, but it can encourage the user to stay on task.
- Input(s): Function receives a value of false.
- Output(s): Sends out an alert and alarm to the user and pauses the timer until the alarm and alert is stopped.
- Dependency: Gets input from Req-06, access timer through Req-03.

Req-08: Implement a whitelist for only previously approved websites.

- Description: During study time, the user is only allowed to visit previously approved websites as specified in the whitelist.
- Rationale: If the user is allowed to access non-educational sites such as social media or games, then they will not be focused on studying and will be wasting time.
- Input(s): Can accept new additions to the whitelist from the user's parental figure with their pin.
- Output(s): N/A.
- Dependency: May receive input from Req-09 if a new site is added.

Req-09: During study time, access to a non-approved site will result in access to the site being blocked. Option to add the site to the whitelist is displayed with the alert.

- Description: Access to a site not on the whitelist will be blocked and a message will show explaining that it has been blocked. An option to add the site to the whitelist will also show but will require a parent pin to add it.
- Rationale: Sites not on the approved whitelist should be blocked when attempted to access because the user will not be studying if they are trying to access social media. But, if the site is a new educational resource, then the parent should be able to add it to the list of approved sites.
- Input(s): The whitelist so that the function can see if a site is not allowed.
- Output(s): Block the site, send an alert, add a new website to whitelist if they choose to do so.
- Dependency: Receives the whitelist form Req-08.

Req-10: When study time ends, display a message for the user to choose to begin or postpone the break.

- Description: When the study time ends, congratulate the user, and ask if they are ready for their break by displaying a "Start Break" button, and a "Postpone for x minutes" button with a drop-down menu selector for 1, 2, 3, 4, or 5 minutes more. Function will call the short or long break depending on the value of numOfShortBreaksLeft.

- Rationale: The user must take short breaks to avoid burn out which would result in less effective studying.
- Input(s): Called when the study timer ends.
- Output(s): Send the message to the user to being or postpone the break.
- Dependency: Receives the end timer message from Req-03.

Req-11: Short break: If a break is started, decrease the number of short breaks needed until a long break and begin break time.

- Description: During break time, the user is allowed to visit non-approved sites and can change the window. There will be a message shown when break time starts to encourage the user to participate in healthy activities such as drinking water, having a healthy snack, and doing some stretches. This function will also update numOfShortBreaksLeft that decides how many more short breaks there must be until a long break.
- Rationale: The Pomodoro Technique says to have short breaks for a few cycles, then a long break before starting again to help prevent burn out and to use time wisely. The users should be allowed to do whatever they would like during break time, but the pomodoro technique recommends healthy activities and not scrolling social media.
- Input(s): N/A.
- Output(s): Timer is changed to count down until the break is over.
- Dependency: Access the user's input from Req-10 and access the timer from Req-03.

Req-12: Long break: If it is time for a long break, begin the break.

- Description: This function begins the long break which happens once per session after a few cycles of study time and short breaks.
- Rationale: The Pomodoro Technique recommends having a long break after having a few short breaks.
- Input(s): N/A.
- Output(s): Break message is sent out like the message for short break from Req-11.
- Dependency: Is called from Req-10 and accesses the timer from Req-03.

Req-13: If the break is postponed, turn timer red and keep counting.

- Description: This function activates when the user does not take the prompted break. Turning the timer read and continuing to count.
- Rationale: Lets the user know that they need to take their break and that the timer is still counting.
- Input(s): User not having hit the break button.
- Output(s): Continues counting, but font is now red.
- Dependency: Input from Req-10 and timer from req-03.

Req-14: Do not allow for the study tabs to be opened during the break time

- Description: Does not allow the user to open the tabs they were using for studying during the break.
- Rationale: Forces the user to actually take their break.
- Input(s): Study timer and break timer.
- Output(s): Lock / unlock
- Dependency: Input from Req-10 and timer from Req-03.

Req-15: At the end of the break, display an alert to begin the next session, and play an audio alert if no response for 1 minute. If this is the end of a long break, ask if they want to restart the same session, stop, or choose a different session

- Description: Asks user at the end of their break if they would like to begin their session again and ask if they want to resume, stop, or start a new session. Plays an audio alert after a minute if no response.
- Rationale: Allows user to decide their next steps after a break.
- Input(s): User says: resume, stop, start new.
- Output(s): Repeats user's input and reacts with the appropriate action. Resumes where user left off, stops the program, or starts a new session.
- Dependency: Input from Req-10 and timer from Req-03.

Req-16: Allow parents to approve new websites with a parent pin

- Description: Allows access to new websites with parent's approval via a chosen PIN.
- Rationale: User can only visit approved sites during study sessions.
- Input(s): Parent PIN, website URL.
- Output(s): User can now visit that website during study sessions.
- Dependency: Req-09 to bring up whitelist option.

Req-17: Website for user to view their history, progress data, statistics, and reports, as well as participate in gamification.

- Description: The user will have the option to log into a website (available on localhost for this project's scope; this website could be deployed on a specific domain) where useful data about their progress can be viewed. The user will be able to participate in gamification to encourage improvements over time.
- Rationale: A centralized location for the user to view all of their data can be very useful to spot trends, access reports, and improve study habits.
- Input(s): Username and password of the user.
- Output(s): Webpage containing user session data.
- Dependency: Python, Flask, and databases.

Req-18: User has the ability to stop the session at any time (emergency stop, may require parent pin)

- Description: User can stop their session with an "end button".
- Rationale: User needs to do something else or leave mid-session. Or they finish all of their work / studying.
- Input(s): User pressing the "end button". (Possibly a parent PIN input.)
- Output(s): Message that the extension is being closed. (Or maybe a parent PIN input box for approval.)
- Dependency: N/A.

2.2 Non-Functional requirements

Req-19: Moderate level of responsiveness

- Description: The extension's features should have a moderate level of responsiveness, at no less than 2 seconds for any action that does not depend on data transfer over The Internet.
- Rationale: This extension does not need to be especially responsive. It should be low enough that the accuracy of the timers is maintained.

Req-20: Very easy to use

- Description: The system should be very easy to use for users, with simple instructions and short descriptions. Only an elementary school reading level should be required. This requirement does not have to extend to settings menus or any other areas that require a parent PIN.
- Rationale: This system will likely be used by children.

Req-21: Self-contained failure

- Description: If the extension crashes, produces an error, or otherwise fails, it should not compromise the integrity of the entire browser or any open tabs. At worst, only the extension itself should cease functioning.
- Rationale: Users should not suddenly lose progress or data in their studies. Even though they will likely have to restart the browser for the extension to get working again, the user will have an opportunity to save progress in their studies so that they may resume where they left off.

Req-22: No storage of browsing history

- Description: Browsing history of any sort should not be saved or stored by this extension. If need be, only the history stored by the browser itself should be accessed, but it should never be copied anywhere permanent.
- Rationale: Browsing history is often very sensitive data, and this extension should not become an attack vector for that data to be compromised.

Req-23: No excessive memory usage

- Description: This extension should not add more than 100 MB of memory per tab.

- Rationale: Although Chrome and other browsers are notoriously memory-heavy, this extension is simple enough that it should not contribute much to that problem. Notable extra memory usage originating from this extension is likely due to a memory leak or other error.

Req-24: Adherence to browser and university policies

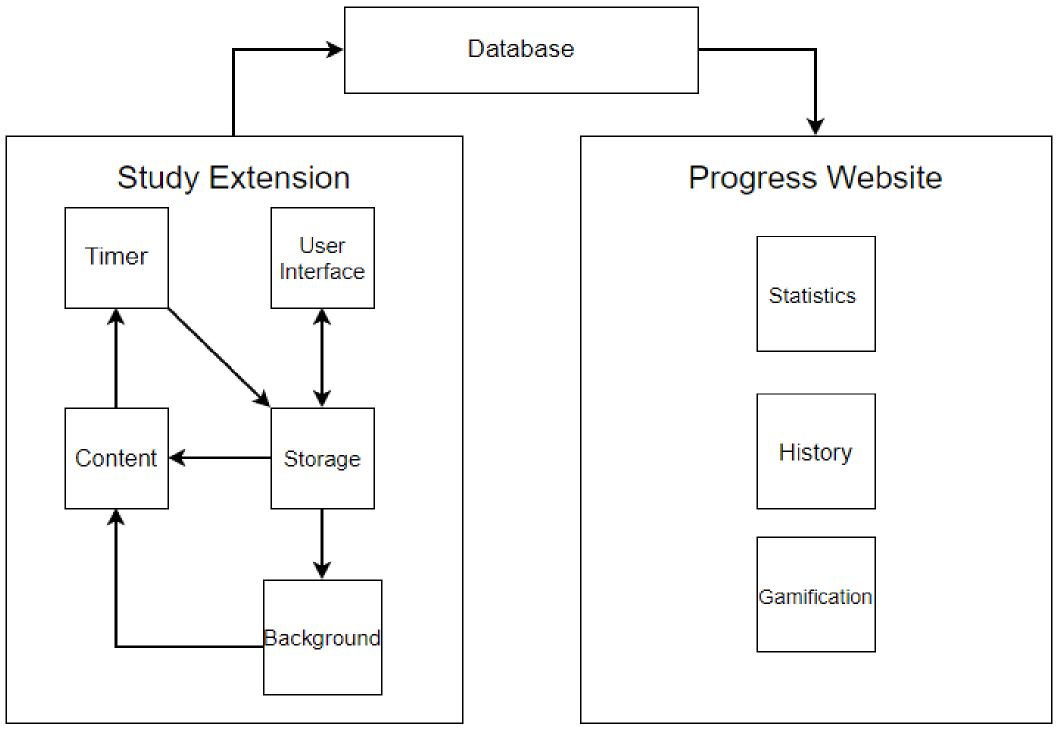
- Description: The extension should adhere to all policies that are required of a browser extension for Google Chrome, that was created by students of Missouri State University. This is subject to change if the extension is ported to any other browsers.
- Rationale: All browser extensions have policies they must adhere to in order to be listed on their respective app stores.

Req-25: Web-based features will update within a day

- Description: When any changes are submitted after a session, those updates should be visible to the user within 24 hours.
- Rationale: Users will need to see their overall progress in their studies within a reasonable amount of time. Since this will be dependent on a web server, some time should be allowed for data transfer and potential outages.

3 Design

3.1 Design Diagram



3.2 Module Descriptions

Study Extension: A google chrome extension to help users focus on studying and remember to take breaks. Users have the option to log in which gives access to the website. The extension stores data about the study sessions in the database.

User Interface: The popup menu where users control the session, make choices about the timer, provide information, and can access the Progress Website.

Storage: Storage provided by the chrome.storage API that is shared among the scripts in the extension.

Background: Monitors the content pages to implement the whitelist and closes tabs that violate it. Continuously checks for mouse or keyboard inputs for statistics.

Content: Makes changes to the current tab's content such as running the Timer.

Timer: Displays a timer on the screen and counts down to the provided time. Contained and executed in the Content Script. Timer accesses storage when finished.

Database: Database that stores information about the study sessions such as total time studied and keyboard and mouse input frequencies. Verifies user login in the extension or the website before allowing access to the website information.

Progress Website: A locally hosted website which allows users to view information and progress on their study sessions.

Statistics: Displays information such as total time spent studying and input frequency. History: Displays information on previous sessions such as date, time began and ended, and total time studied for the session.

Gamification: Provides incentives for the users to study more through a game-like system where the total amount studied is translated to points and then to levels and titles.

4 Implementation and Testing

4.1 Development environment and tools

Overview
The Google Chrome extension was developed in Visual Studio Code with JavaScript, HTML, and CSS and was tested using the Developer Mode in the Extensions page of the Google Chrome browser.

Manifest JSON
The first step in developing the extension was to create a JSON file titled ”manifest” which defines different aspects of the extension such as the name, description, version, scripts for the popup menu, scripts for the background service worker, content scripts, and various

permissions. The permissions requested were storage, tabs, activeTab, and scripting.

Website

Our website contains 3 pages, a Home page, a Stats page, and a Login page. The Home page consists of a welcome box for the dynamic user and instructions on the site. It also contains the user's dynamic name, current level and rank, next level and rank as well as rewards (if the level is a multiple of 5), and a progress wheel / percentage to show their current level.

The Stats page works by displaying the variables we use for our experience system in boxes. These boxes are split for what variables they represent: the mouse clicks, key presses, time spent studying, and days studied. Time spent studying is based on the amount of time the timer runs, and days spent studying is a simple counter that checks the date. Mouse clicks and key presses are collected while the extension is running.

The Login page works as a typical login page where it checks the database for a matching username and password and logs the user in, displaying their personalized stats and levels for gamification.

All of the variables listed above are tied to the user in the database, and are displayed based off of the profile that is signed in.

Server

The server was developed in the Visual Studio Code IDE with python 3.10 using The Flask and Flask-Sessions modules. The server is composed of an app.py file, which serves the website on the localhost. If desired, this can be later configured to deploy the server on a purchased domain, however, this step was omitted since there are no budgets allocated to this group project by MSU or the CS department. FLask sessions are used to facilitate a user's login actions. The server currently has support for various endpoints, such as "/", "/login", "/signup", "/logout", etc.
The server employs a database to store user's username and hashed password. Currently, for demonstration purposes only, these passwords can be temporarily acquired by the admin login credentials. The python class db_tools facilitates executing common SQL queries, such as creating a new users table, adding users to the database, retrieving passwords for comparison, etc.

A variety of tests were performed to ensure the system behaves as expected, such as:

- User visiting the "/" endpoint without signing in.
  - The user is rerouted to the login page.
- User attempting to sign in without having signed up before.
  - The user is redirected to the signup page.
- User attempting to sign up with existing username.
  - The user is redirected to the login page.
- User attempting to sign in with an incorrect password.
  - The user is redirected to the login page and advised the password is incorrect.
- The user attempts to manually access the "/logout" endpoint.
  - If the user is logged in, the session name is removed and user is logged out. Regardless, the user is redirected to the login page.

Popup Menu

The popup script contains the main control menu for the extension. This script runs each time the popup page is opened and first checks shared storage to see what state the popup menu should be in. The possible states are the main page, which displays when there is no study session started, a study state which displays when the user is in study mode, an intermission state for between study and break, and break state for during short breaks, and a long break state for during long breaks. The short and long break states are functionally the same, they just have a different title. Once the script determines which state it is supposed to be in, it shows or hides the appropriate elements so that it looks the same as the last time the user looked at it, unless the state changed when the timer ended.

Along side checking and updating the state, the script also pulls the relevant timer variables from storage such as the number of times studied this session, number of short and long breaks taken, number of cycles left, and the chosen number of minutes for studying, each break, and the total number of cycles. All of this information is used to display some progress information at the top of the page while in a session. The information is written as a few fractions such as the number of breaks taken out of how many more there are left.

When the popup is in the main page state, it also refills the choices and inputs that the user made in the timer choices menu so they do not have to reenter the information. Additionally, the customer timer inputs are saved between sessions to allow for easily repeated custom sessions.

When the user accesses and edits the text box that contains the list of domains for the whitelist, the function uses a regular expression to check if it is in the valid from of a domain such as www.google.com or en.Wikipedia.org. This check will not allow strings of just letters to get through which would mess up the whitelist. If the domain is valid, it is kept and stored. If not, it is removed and a message is shown beneath the text box listing the entries that were removed.

When the begin timer button is clicked, it checks that a radio button has been selected and sends an alert telling the user to make a selection before trying to begin. If an option is selected, it checks to see if it was a premade timer option or the custom. If the custom timer was chosen, the script also checks if all of the inputs are filled in before allowing the session to begin, and sends an alert if any are empty.

Once the choices are deemed valid, the session begins by saving all of the timer choice information to shared storage. Then the state of the page is set to study, the whitelist blocking is enabled, and the start timer function is called.

The function to start the timer begins by asking the user if they are ready to reload the current tab to allow the timer to show up. If they are not ready, then the state stays on the main page until they come back to press begin, and choose to allow the page to reload. This is to allow the user to make any necessary changes to the current page such as saving their work. Once the user is ready, the page is transitioned to the study state by hiding the timer options and end button, and showing the end button and pause button. Then it calls a function to get the end time for the timer and finally reloads the current page.

The calculation function gets the current time, gets the end time by adding the current time and the number of minutes to study multiplied by 60000 since the getTime function returns milliseconds. Then turns the time into a string of the form HH:MM:SS and saves it to shared storage.

Content Script

When the current tab is reloaded, the content script is run on the current tab's page. The content script first creates a div HTML element, and adds styling to prepare for the timer to be shown on screen. There is a message listener that responds to a message sent from the popup script when the user presses the pause button. When a pause message is received, the content script flips the paused value on or off, and saves the time the timer was paused or unpaused. When the content script is run without a message being sent, it first checks if the timer is supposed to be paused. If paused, it does nothing. Once it is unpaused, then the script accesses shared storage to get the end time for the timer and parses out the hour, minutes, and seconds and then stores it in a date object.

The script then enters a set interval function which runs every second. The logic of this function was found in an article and its reuse is described in the later section "Reused components". This function first checks if there is any value stored for the pause start time. If there is, then it subtracts the start pause from the end pause time to get the amount of real time that passed while paused, and adds that to the timer end time. The start and end pause times are reset to null. For each second that this function is called, it gets the current time and subtracts it from the end time to find how much is left. Then gets the hours, minutes, and seconds values of the difference and displays the string to the screen in the div created earlier. It then checks if the difference is less than or equal to 0, which means the timer is done.

If the timer is done, it checks what state the popup is in, increments the count for that corresponding state, stores the count, and changes the state to intermission. For example, if the popup was in the study state, then the number of study times completed is incremented. The timer display is then set to an empty string and the whitelist blocking is turned off.

Background Script

The whitelist blocking is handled in the background script, also called the service worker. The logic of this portion came from an article, and the extent of reuse is described in the "Reused components" section of this report. When the extension is ran, the background first checks if there is an existing whitelist in storage. If not, then it creates an empty array as a placeholder. It also checks if there is a value called "enabled" of type boolean and creates and sets it to false if there is not one.

Also in the background script is an event listener for when any tab in the browser window requests for a page to load. This is referred to as the tab being updated. When the script receives the notice, it grabs the tab id and information about the change. These parameters allow the function to grab the URL from the new tab and test it. First, it tests if it exists, or if it starts with "http". If either of these things are false, then the function returns and does not block it. This is so that pages that begin with "chrome" are allowed through and never blocked because they could be important settings pages. If it passes the first test, it moves on to the second test where the script looks through the list of allowed domains and looks for the domain of the current URL. If the domain is found, then the tab is removed, and the URL is saved to shared memory and a flag that says a page was recently blocked is set. This is so that when the user goes back into the popup page, the script checks for these values and asks the use if they want to add that domain to the whitelist. If they choose yes and a valid pin is provided, then the domain is added and will not be blocked the next time they ask to open it.

4.2 Testing scenarios and results

5 References

[1] Bucka, P. (30 July 2020) *Learn the most useful Chrome APIs by creating Block Site Chrome extension.* DEV. https://dev.to/penge/learn-the-most-useful-chrome-apis-by-creating-block-site-chrome-extension-2de8

[2] W3Schools. (n.d.)*How TO - JavaScript Countdown Timer.* https://www.w3schools.com/howto/howto$_js_countdown.asp$