Trailmix

Project Part 3: Design

Team 8

Sarah Cooper

Matthew Deagen

Leo Galang

Marc Ace Montesa

Instructors

Sergiu Dascalu, Devrin Lee

External Advisors

Jay Thom, Nathan Thom, Maxwell Thom

Department of Computer Science & Engineering

University of Nevada, Reno

November 17, 2020

Table of Contents

Abstract

As large tech companies like Google and Facebook continue to dominate the Internet through running targeted ads, privacy becomes a growing concern. To address this issue, Trailmix will be a browser extension that randomizes user interests to allow for complete anonymity and reduce advertisement tracking. Trailmix will be written in JavaScript React and will utilize the concept of a rotating library of user profiles and a whitelist of recommendations approved by the user. The advantage this software has over competitors is the utilization of a database system that will allow for users to use the complete functionality while still allowing the user to actively mitigate and control their preferences with forced recommendations.

Introduction

Data security remains an integral part of the ever growing age of technology. As digital privacy is not largely considered during the creation of applications, larger corporations are able to get away with tracking and logging user preferences and data; which may potentially be used in questionable ways. Commonly seen is third party advertisements and corporations selling user's data to other companies where they can then target those users with advertisements. With the increase of users connecting to the internet and signing up for applications such as Facebook, YouTube and other social media without being cautious of their digital footprint, the increased need for applications that can protect them against companies who may use their data.

Team 8's recommendation is to develop a data obfuscator application that will be implemented as a Google Chrome web extension, and rolled out onto the Google Chrome Web Store. As the team's development progresses, major changes were made to the data structure of the application. The first is that instead of utilizing a peer-to-peer network to connect users to other users' preferences, the team elected to use a centralized database to store and generate profiles for users to be assigned to. Here, user preferences are stored in one database containing their hashed username, password, and search terms. This allows users to review their preferences securely, and erase them accordingly. From here, user preferences are deallocated from the individual users and randomly assorted into a list. Finally, a second database may be used that will take this list of preferences, randomize them, and allocate them into randomly generated user profiles. Then a user starts up the application, they effectively attach themselves to one of these randomly generated profiles and browse the internet with this set of preferences.

High-level and Medium-level Design

System-level Diagram

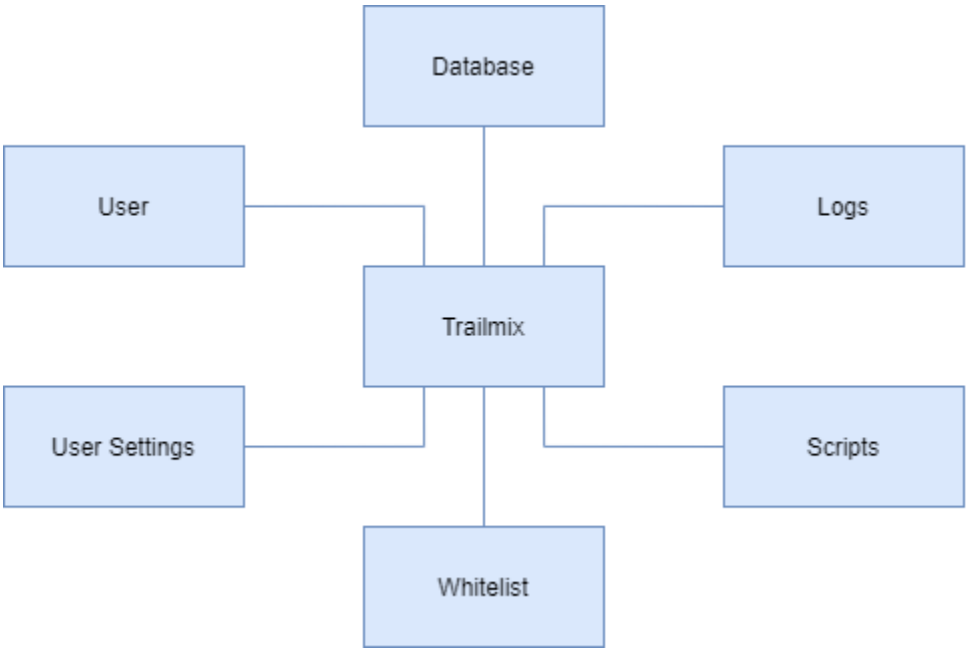

Figure 1: Trailmix Context Model

Program Units

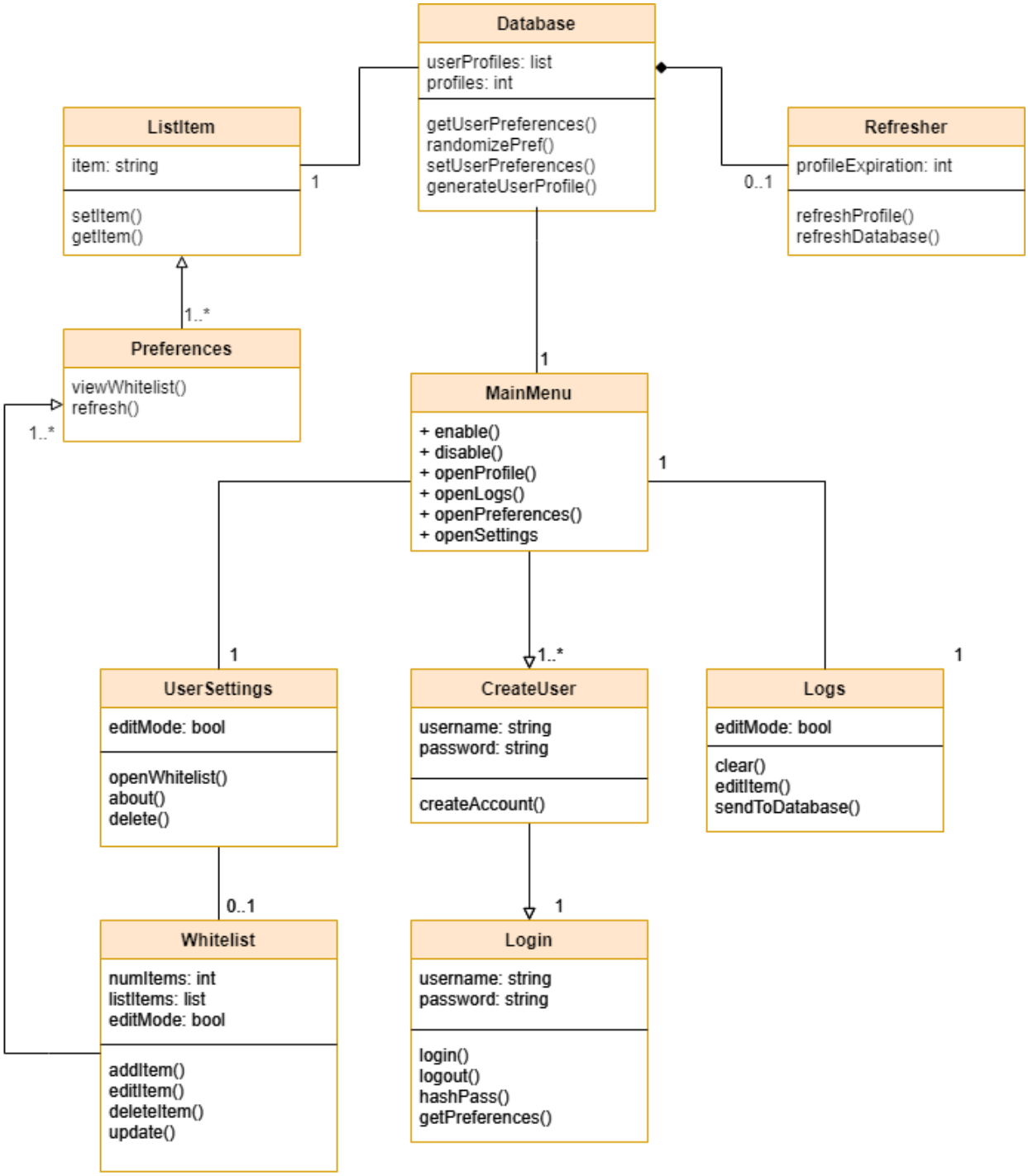

Figure 2: Class Diagram

Class: UserSettings	This class is designed to allow the user to make changes to the extension's settings.
editMode: bool	When this variable is true, the user will be able to edit information related to the settings.
openWhitelist()	This will display the whitelist page where the user can add,edit, or delete items.
about()	This will send the user to the about page, where they can read information about the extension.
delete()	This will allow the user to delete their account. This method will delete all of their account information from the server.

Class: Whitelist	The whitelist is a user-specified list of various topics that interests them. This helps determine what the user sees for their forced recommendations.
numItems: int	This is an integer that represents how many entries are in the whitelist.
listItems: list	This is a dynamic array of strings, where each string is a topic that the user is interested in.
editMode: bool	When this variable is true, the user will be able to edit information related to the settings.
addItem()	This method allows the user to add an entry to their whitelist. It also sets the editMode variable to true. numItems is incremented.
editItem()	This allows the user to change a specific entry in the list.
deleteItem()	This method allows a user to delete an entry to their whitelist. It also sets the editMode variable to true. numItems is decremented.
update()	This will verify the changes and set the editMode variable to false. It also sets the editMode variable to true.

Class: Preferences	This class is used to control the preferences of the user.
viewWhitelist()	This method sends the user to the whitelist page.
refresh()	When a change is made, a refresh button will appear. The button will call this method to reload the page.

Class: CreateUser	This class is what is used to hold the information of the account and allow for creating an account.
username: string	This is a string that holds the user's username.
password: string (digest)	This is a hash value that represents the password. The hash value is created by the plaintext password and a salt value.
createAccount()	This is a constructor that creates a new account for the user.

Class: Logs	This class keeps a record of the last ten search items done by the user. These entries can eventually be used to create profiles
editMode: bool	When this variable is true, the user will be able to edit information related to the logs.
clear()	This clears the entire log for the user.
editItem()	This edits a specific item in the log.
sendToDatabase()	This method will send all of the entries in the database to be added to profiles.

Class: MainMenu	This class describes the main user interface of the chrome extension. This is where the user can turn on the extension and change various settings.
enable()	This allows the user to enable the extension.
disable()	This allows the user to disable the extension.
openProfile()	This directs the user to their profile page.
openLogs()	This directs the user to their logs.

openPreferences()	This directs the user to their preferences.
openSettings()	This directs the user to the settings.

Class: Database	This class generates profiles for a user to attach to while browsing the web. This is where much of the main functionality of the program will take place, as the profile generation, assignment, and expiration would ideally happen here.
getUserPreferences()	Retrieves a list of the user's preferences to be randomized and assigned into a new profile, along with other users' preferences
randomizePref...()	Randomized the user's preferences with that of other users to be assigned into a new profile.
setUserPreferences()	Assigns the randomized preferences into tuples of the database that will be used for profiles.
generateUserProfile()	Allows a user to attach themselves to one of the randomly generated profiles above.
userProfiles : list	The list of all the profiles a user can attach themselves to.
profiles : int	Tracks the number of profiles on the database to keep track of storage usage.

Class: Refresher	This class is a companion class to the Database class, in which if the user needs to refresh their browsing profile, this would be done with the methods here.
refreshProfile()	Delinks the user from their previously used profile and attaches themselves to a new profile.
refreshDatabase()	Refreshes all the tuples of the database to generate an entirely new list of profiles with new sets of preferences.
profileExpiration : int	An expiration time for each profile, assigned at profile generation.

Class: ListItem	The role of this class is to be able to define and change each item in a list.
item: string	This is the variable that holds the list item.

setItem()	This method changes the item string to something else specified by the user.
getItem()	This method returns the item string.

Class: Login	The role of this class is to allow the user to log in and log out of their account.
username: string	This is a string that holds the entered username.
password: string (hashed)	This is a string that holds the entered password. This will be hashed with a salt value to authenticate the user.
login()	This method allows the user to log into their account.
logout()	This allows the user to log out of their account.
hashPass()	This method is responsible for hashing the password to make sure that it matches with the value in the database.
getPreferences()	This returns a list of preferences associated with the account.

Data Structures

Databases will be used to store all of the preferences across all users. A function within the database will randomly pick entries from these arrays and use them to create a complete profile.

Hashed Username	Hashed Password	User Preferences
ccdcd9302027fc2ddaef605f4cadd6e0	0e11fc23873b65e0b270ed27a0561771	#cars #iphone #kanye #ps5
bc926f7086deb75d0ec4977e6527d923	bddcf7982a606265f092237ace2e312a	#spiderman #comicon #nvidia #cats
e43fd347a77efd1e88	86bc0258a45e0e9770f1222eca5be7b7	#xboxseriesx

c19a0b716aec8d		#mallofamerica #singapore #logitech
89166afc310d10973468d5f38e979a3d	45c92e0b6b0aed0238890eb11ef2ac1a	#dog #fitness #webdesign #turbotax
f220fb401a5f737009ca8fe760de6e18	021954140486c113116f645093f8c1dc	#tesla #pikachu #tinder #boardgames

Table 1: Example of a user database

ID#	Preference #1	Preference #2	Preference #3	...	Preference N
001	#cars	#xboxseriesx	#tinder	...	*#preference*
002	#logitech	#fitness	#cats	...	*#preference*
003	#singapore	#tesla	#iphone	...	*#preference*
...	...	...	...	...	...

Table 2: Example of the database of randomly generated profiles

Lists will be used to gather user preferences, deallocate them from a specific user, then composed into a list with other users' preferences to later be allocated into the database for profile creation.

Detailed Design

State Diagram

The state diagram for Trailmix specify the different states that Trailmix can be in and the transitions to reach each state. The first state is the login state where the client will be prompted to either log in or make a new account. If they need to create a new account, they are automatically sent back to the login state. Once they are logged in, the next state would be the icon menu state. This state can lead to many different states such as enable, disable, settings, preferences, logs, and profiles. The enable state will set the enable icon to green and will resume the functionality of extension. Similarly the diable state sets the enable icon to red and will disable the functionality of the extension. While in the disabled state, the extension can also be in other states, such as logs. The next state that the extensions can be in is the log state. This state simply shows the current user's search history that is logged by the extension. This is to allow the user to know what the extension is taking from them. The next state is the profile state. This shows the full profile of the users including their username and the list of preferences that they have in their profile at the moment. It also allows the user to generate a new profile or list of preferences. The preference state shows the current list of preferences as well as a refresh icon, a delete preferences icon and an add preferences icon. The refresh state will allow for the user to generate a new set of preferences. Delete preference state will allow a user to choose (either type in or click) on the preference they do not want to see. The add preferences, is similar to delete preferences, where the user can add a preferences they do not mind seeing to their whitelist. The last icon that can be seen on the preferences state is the whitelist icon. This icon leads to the whitelist state where a user can see the whitelist they have created. Lastly, the settings state allow a user to change their password, logout or uninstall the extension. The state to change a password will take the old password and the new password to change or update the password. That state will then lead to the login state to gain access with their new password. The logout state will also send the user to the login state and they must log in again. In the login state, functionality is not working. Last users can uninstall the extensions which will end all processes of the extension.

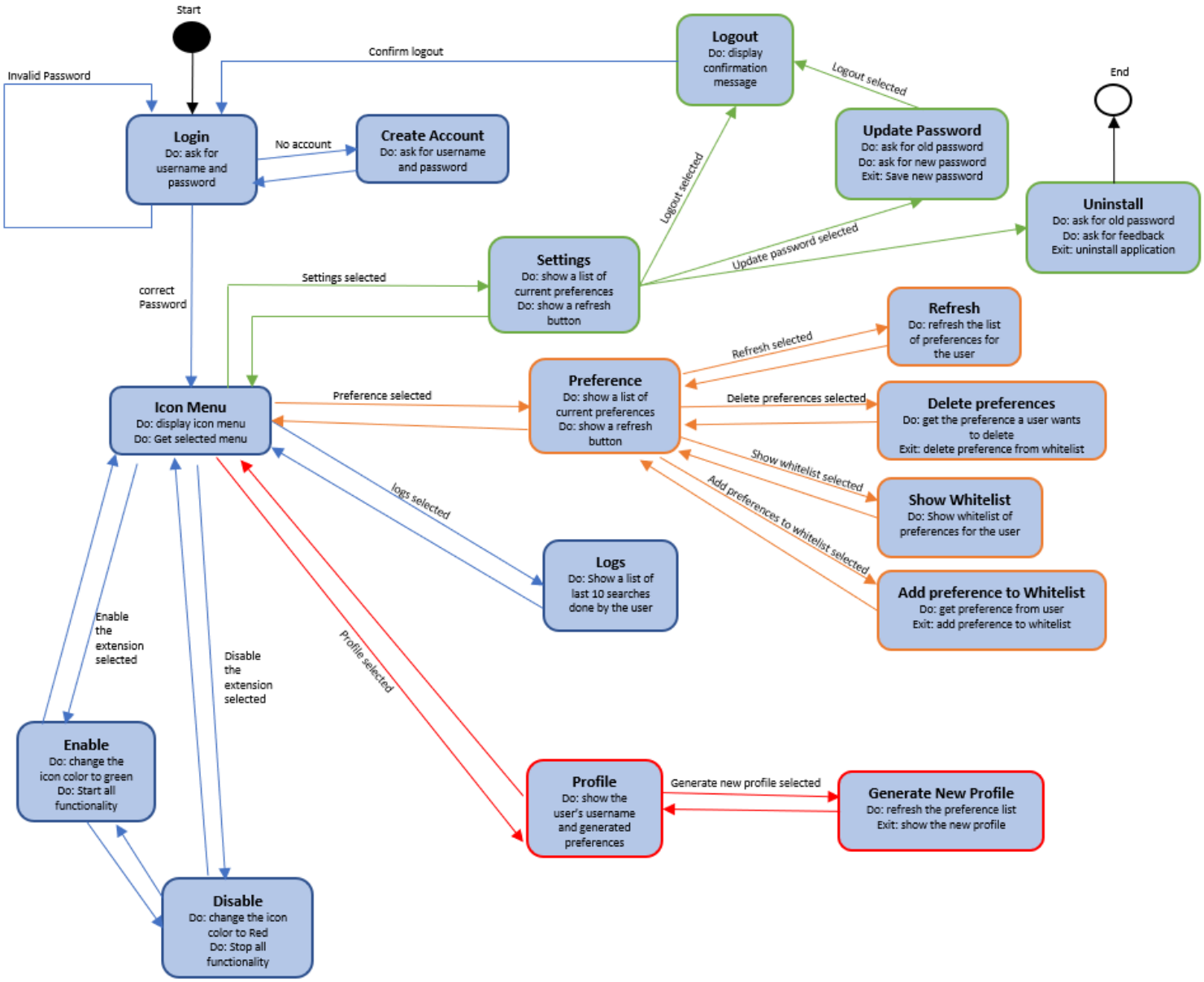

Figure 3: State Diagram for Trailmix.

Sequence Diagrams

The following sequence diagram shows the sequence or steps that will be taken when a client wants to generate a new profile with new preferences. First the user would login, assuming they have an account. After getting in, the user can choose to generate a new profile which will call the function GenerateProfile(UID). The server would then call the function UpdatePref(UID) where UID is the primary key and would allow the server to pull the correct tuple information from the database of users. While UpdatePref(UID) is running, another function is called PullPrefs() to pull preferences that are gathered together in the preferences database. The database returns a list of preferences that UpdatePref(UID) can use to update the user's preferences. Last the server will return to the user the new profile with the updated list of preferences.

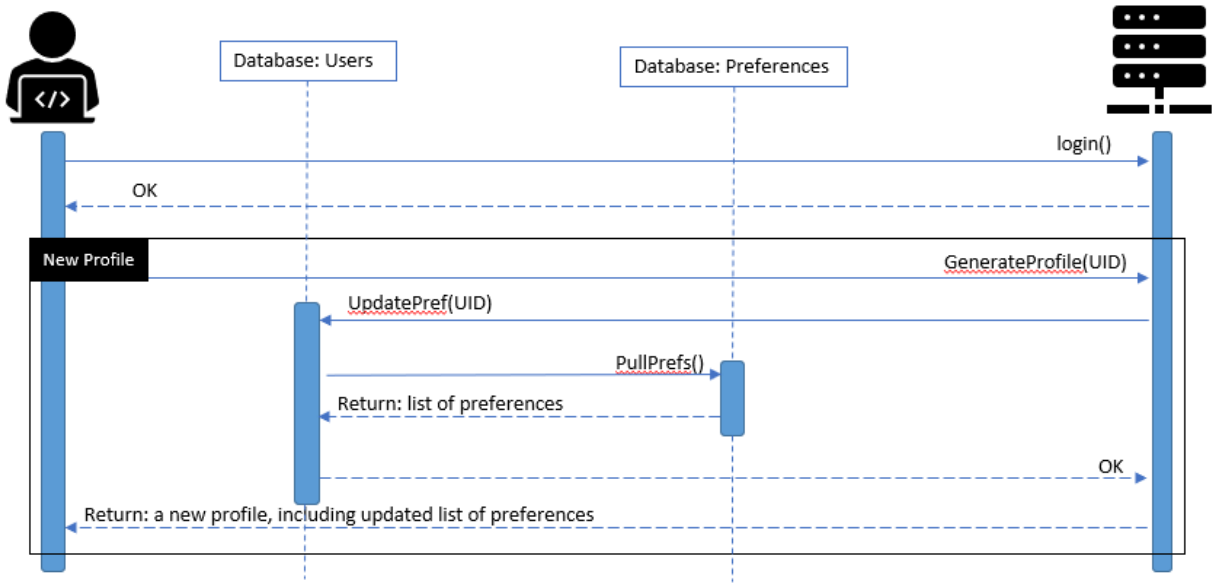

Figure 4: Status Diagram for Trailmix's GenerateProfile(UID) function.

Sequence Diagrams (cont.)

The following sequence diagram shows the sequence or steps that would be taken if the user wanted to update their whitelist of preferences with a new preference. Starting with the client logging in, and calling the server to AddWhitelist(UID, preference). The server then calls the database of users, using the primary key, UID, to call for the whitelist of that user. This is to check if the user already has the preference in their whitelist already. If the preference is not in the whitelist, then update the whitelist and return the message: “whitelist updated.” If the preference is already in the whitelist, do not update the whitelist and return the message: “preference is already in whitelist.”

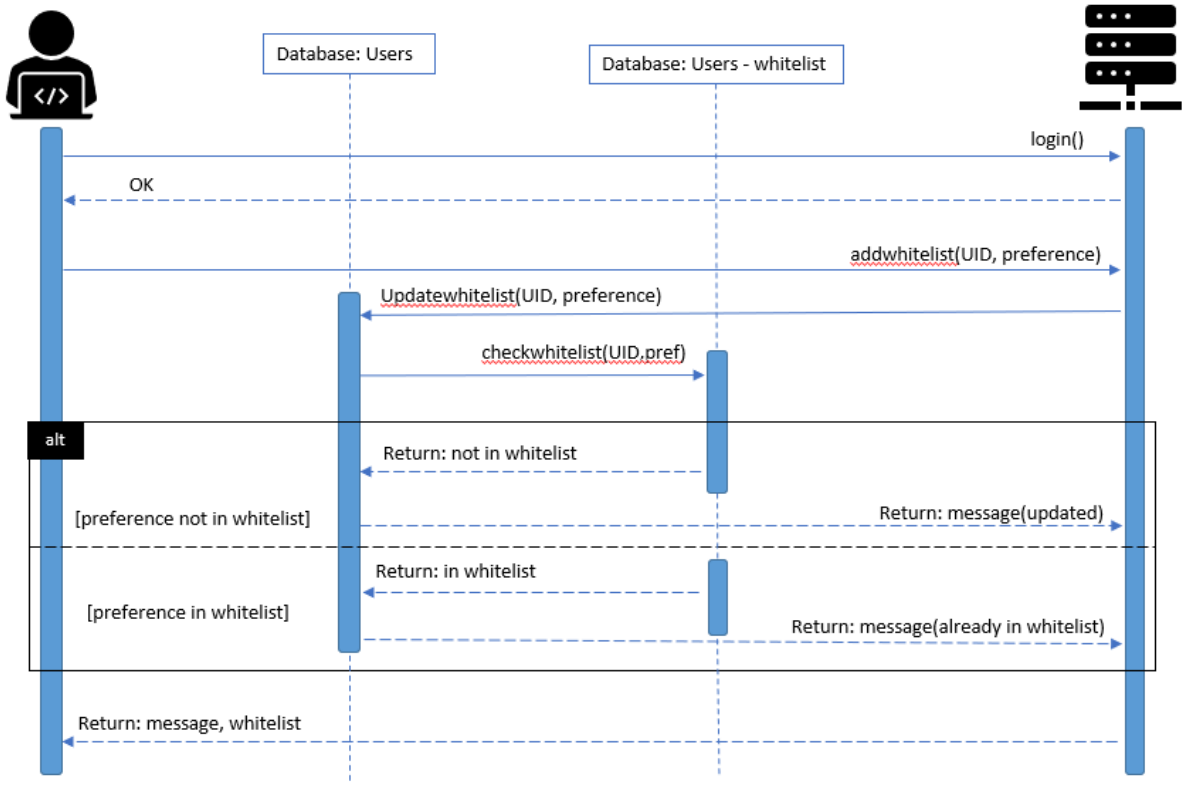

Figure 5: Status Diagram for Trailmix’s Addwhitelist(UID, preference) function.

Activity Diagram

The following activity diagram shows the activity that can be followed for the delete preference function. When a user calls the delete preference function, the system will send a confirmation. If the user responds no, then exit the function, nothing needs to be updated. If the user says yes, then the function will check if the preference was in the whitelist. If it was, then delete it and return the message: "preferences are updated." Otherwise, if the preferences is not in the whitelist, return the message: "preference was not in whitelist."

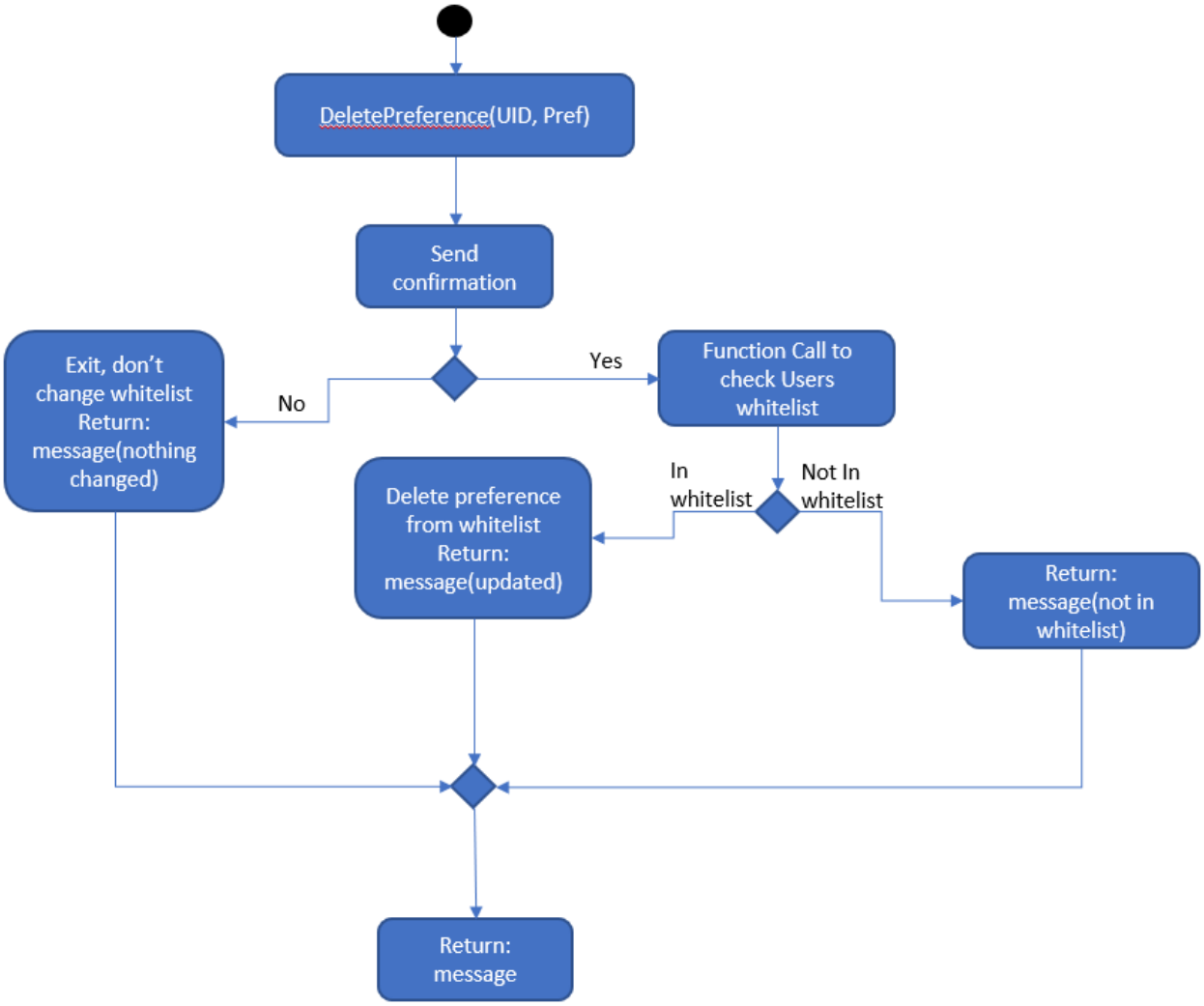

Figure 6: Activity Diagram for Trailmix's DeletePreference(UID, Pref) function.

User Interface Design

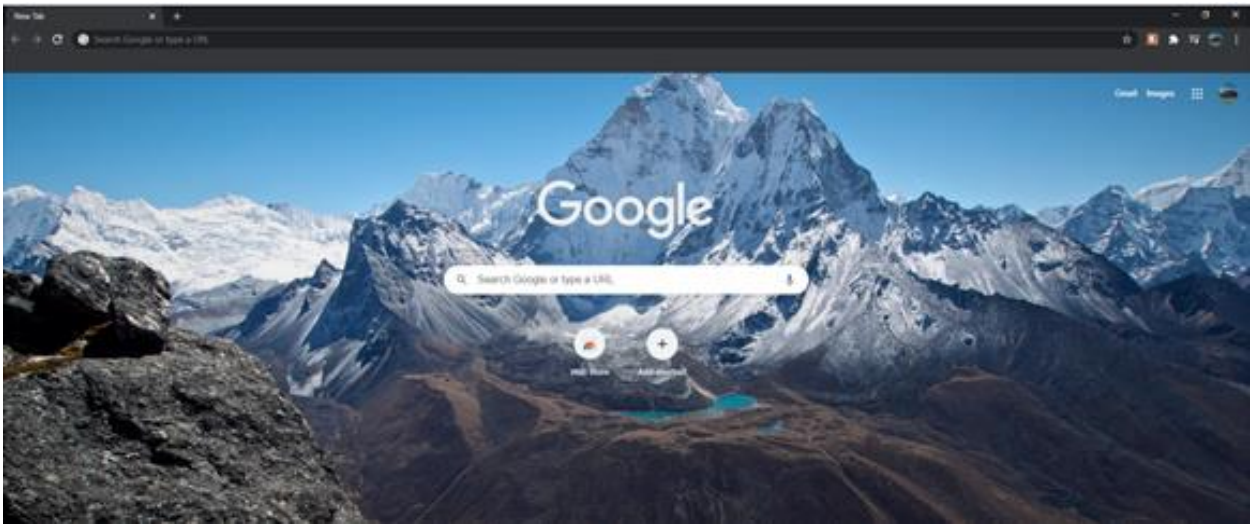

Figure 7: A snapshot of the Google Chrome browser with the Trailmix extension icon in the top right corner.

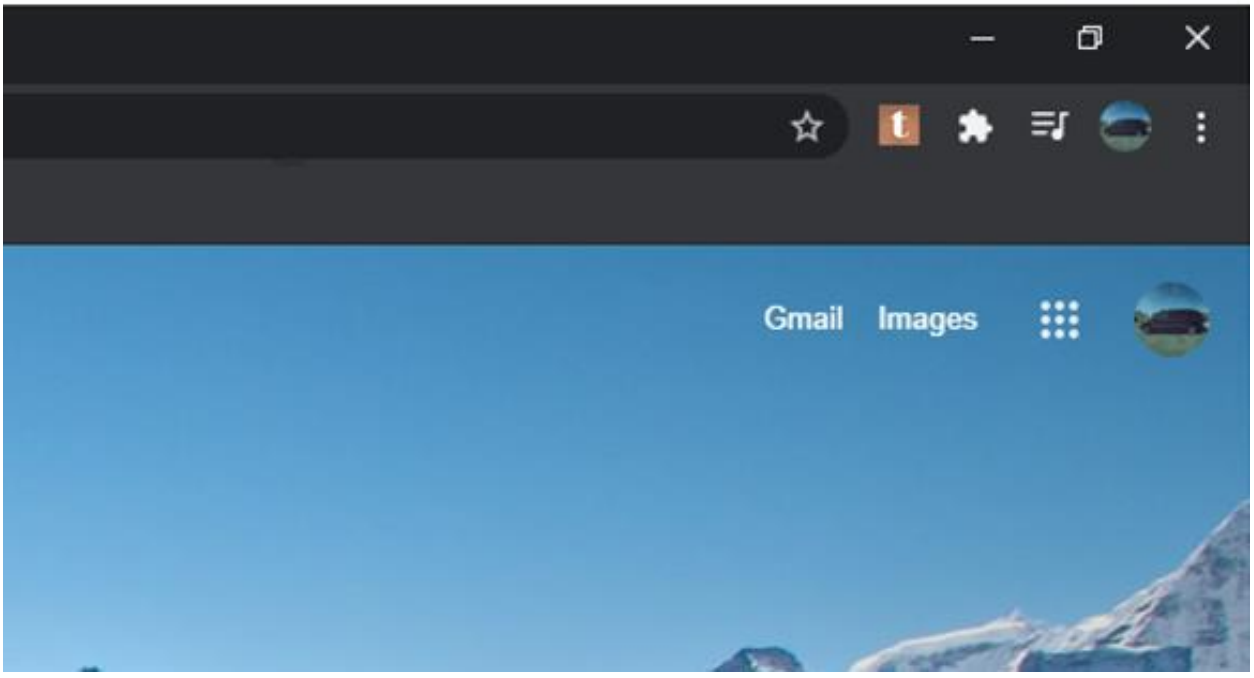

Figure 8: A closer look at the Trailmix extension icon.

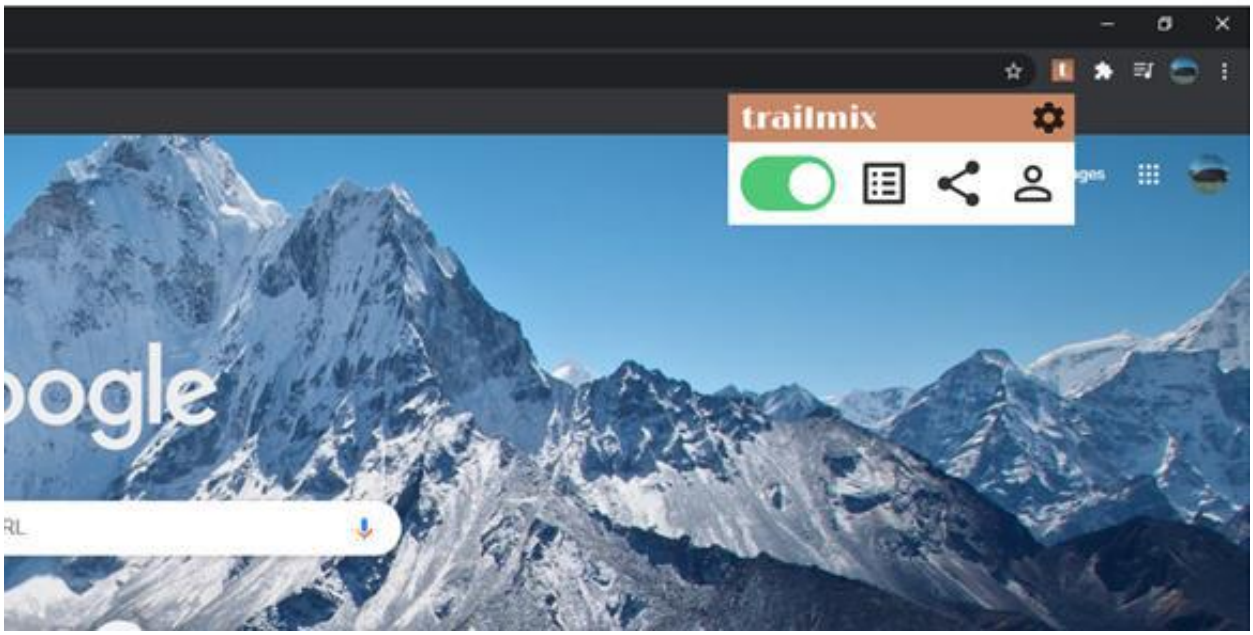

Figure 9: When the Trailmix icon is clicked, a drop down menu will appear. The settings can be accessed in the top right corner. Beneath the header, users can enable Trailmix as well as view log details, share Trailmix to different platforms, and see their profile.

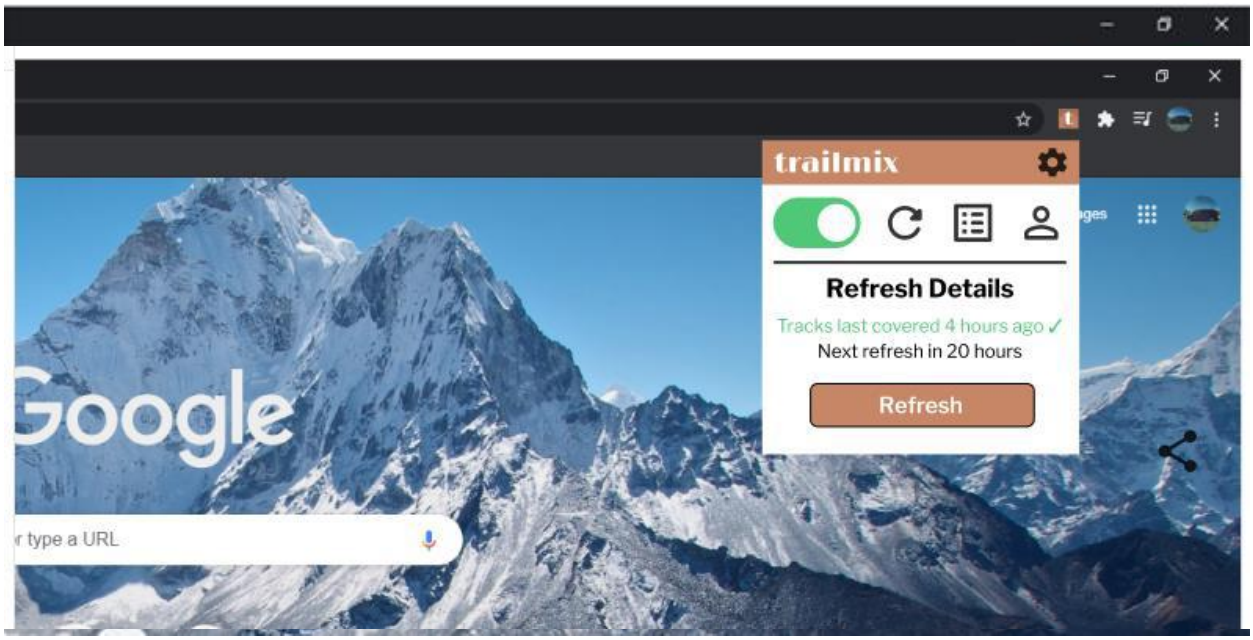

Figure 10: Users can disable Trailmix at any time.

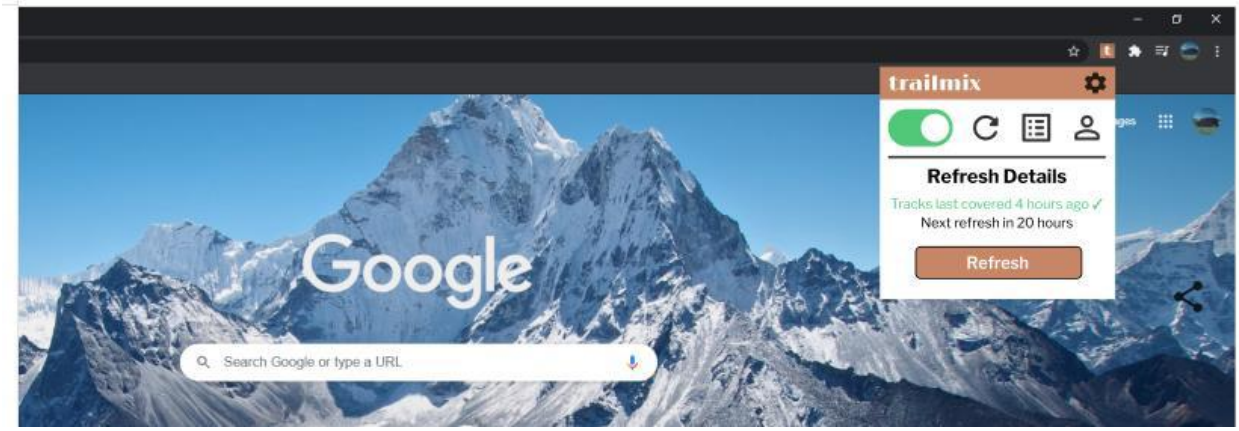

Figure 11: The refresh details are displayed when the icon next to the enable button is clicked. The information above tells us that Trailmix was used 4 hours ago and will refresh in 20 hours. Users can click the refresh button to randomize their preferences if they wish to do so.

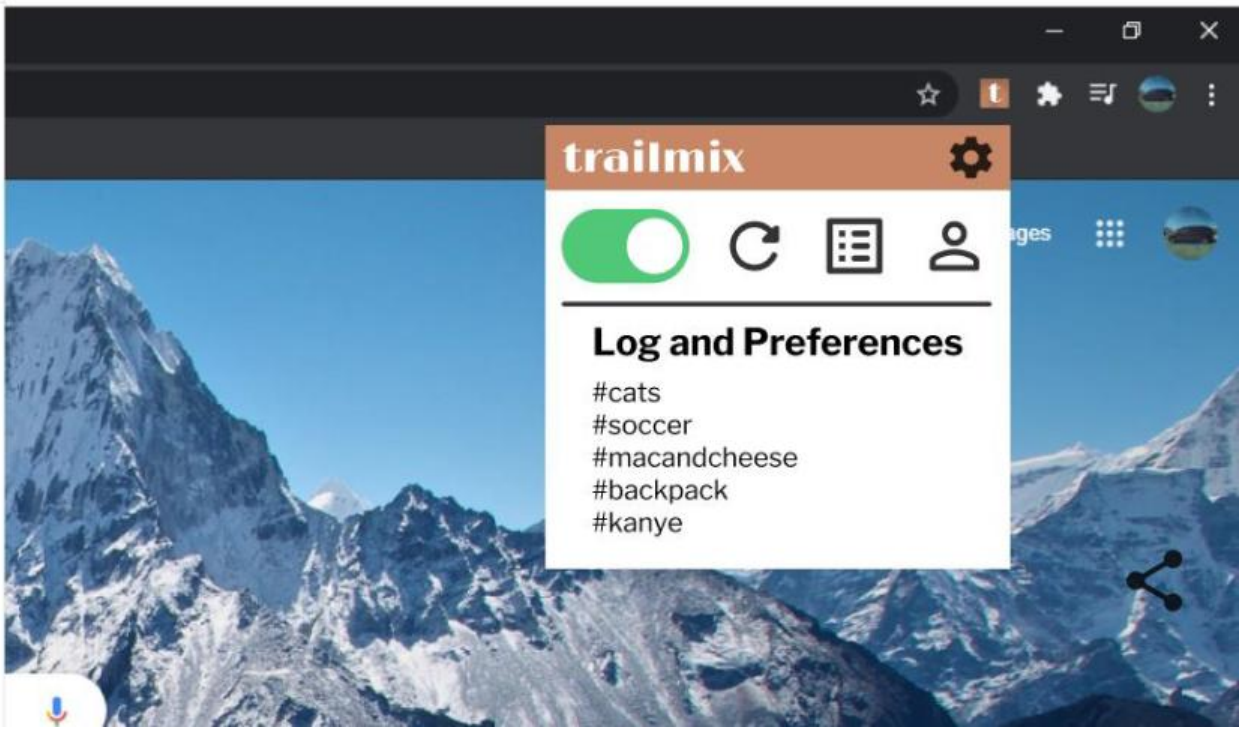

Figure 12: The log and preferences can be accessed when the list icon is clicked. The above screenshot shows the current preferences of the user.

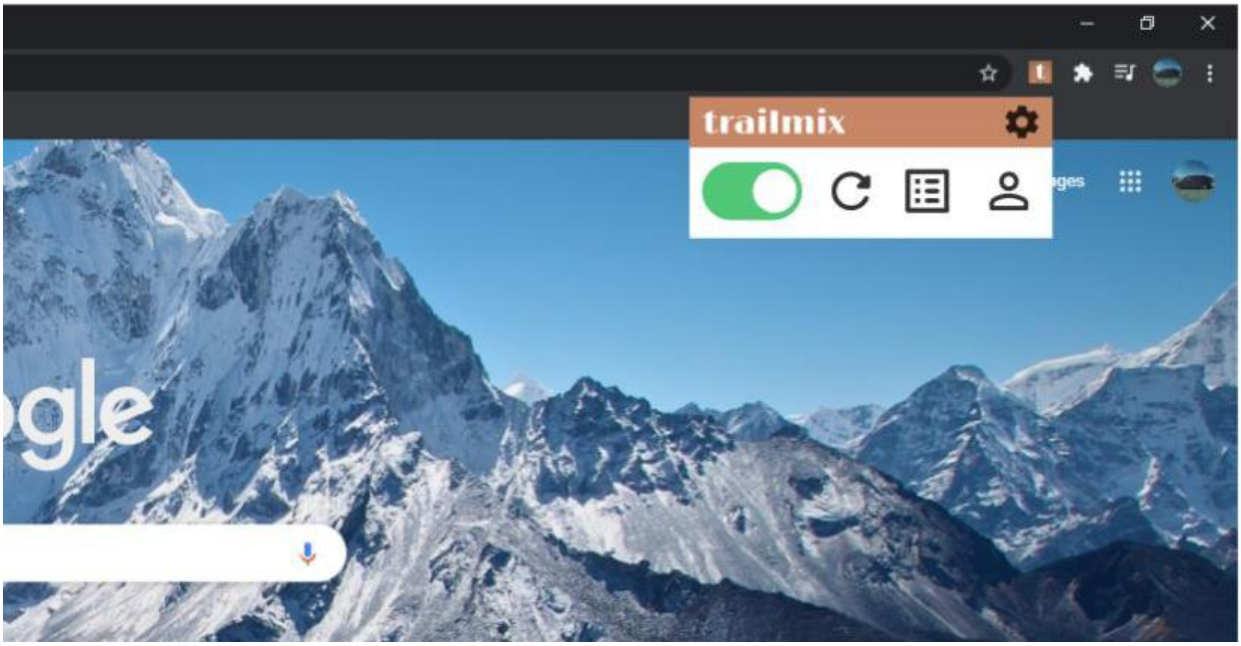

Figure 13: Next to the list button is the profile button. Here, users can access their account details. Team 8 is still working on button functionality and what they will display to the user.

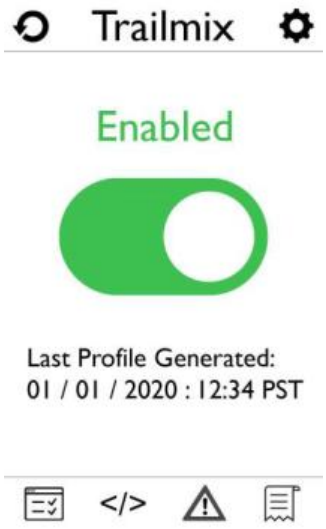

Figure 14

Figure 15

Figure 14 and 15: The initial screenshots of the Trailmix interface. There is a lot of overlap from the first design to the current design.

Glossary

A

Algorithm	A process or set of rules or instructions to be followed in either calculations or problem-solving operations in order to make a computer perform a specific task.
Anonymity	The quality or state of being unknown.

B

Blacklist	A list of objects or things that is regarded as untrustworthy.
Browser Extension	A module for an internet or web browser to add functions and features for day-to-day browsing.

C

Cookie	Saved information about a user, based on their search history
Computer Script	A list of commands that are executed by either a program or a scripting engine, mainly used to automate processes or generate some desired outcome.

D

Database	A set of data accessible by the client or server.
Digital Footprint	Trail of data created while using the Internet.
Domain	A domain name is an identification that defines the administrative authority or control within the internet. (e.g., .com, .net, .gov, etc.)

E

Encryption	Process of encoding information.The process of encoding information into secret code to hide the true meaning of the information.

F

Framework	A platform that provides a generic functionality for which the user may stage

	applications, that provides a library, compiler, and other developer tools to streamline the development process. In simpler terms, it is a general foundation that a program can be built on top of. An example of this would be React.

I

Incognito	Incognito or private browsing is a feature of web browser's that allows the user to search without saving search history and cookies.

L

Logger	A piece of software used to track a systematic recording of events, observations, and measurements.

O

Obfuscation	To make something obscure, unintelligible, or otherwise unclear to where it cannot provide any relevant information.
Open Source	Denoting software for which the original source code is made freely available and may be redistributed and modified

U

User Interface	The means by which the user and a computer system interact, in particular the use of input devices and software.

V

VPN	Virtual private network, or VPN, extends private networks over public networks to enable users to send and receive data across public networks as if they were using a private network

W

Web Browser	A way to retrieve information from the internet. Also called an internet browser.
Whitelist	A list of objects or things that is regarded as trustworthy.

List of References

Carbo, J. "Don't Just Rely on Data Privacy Laws to Protect Information," Security Magazine, Journal, accessed on November 3rd, 2020 : https://www.securitymagazine.com/articles/91775-dont-just-rely-on-data-privacy-laws-to-protect-information
This journal emphasizes the importance of data security, stating that while there are various laws that help preserve user privacy, many of these can be interpreted in ways that still forgoes user confidentiality. The journal also describes the best practices in preserving data privacy while browsing through the web.

Goldfarb A., Tucker C. "Privacy Regulation and Online Advertising," MIT Open Access Articles, Journal, accessed on November 3rd, 2020 :
https://dspace.mit.edu/bitstream/handle/1721.1/64920/Tucker_Privacy%20Regulation.pdf%3Bjsessionid%3D25925F96FD45465F5B3993D583629D45?sequence%3D1
This journal describes how advertising agencies use consumer data to target users in various advertising campaigns. This also writes of the European Union's response to these agencies, creating various laws with the intent to secure user privacy; however, this also demonstrates the loss of effectiveness against these targeted campaigns based on the generalized content and presence of some of these companies.

Google Chrome, "Getting started Tutorial," website, accessed on September 27th, 2020: https://developer.chrome.com/extensions/getstarted
This website is very valuable to Team 8's ability to develop a Google chrome browser extension. This just gives the basic idea of how to build an extension and how to upload it to the Google Chrome Web Store where all the browser extensions are.

React, "Tutorial: Intro to React," website, accessed on October 10th, 2020: https://reactjs.org/tutorial/tutorial.html
This website will allow Team 8 to use JavaScript's React to create Trailmix. Without using react, the whole extension would be done in HTML, CSS and JavaScript, however with the help of React, team 8 can reduce the amount of HTML and CSS that needs to be written and thus make the code simpler and easier to read and understand. Less code also means less overhead time which is important for a security application.

Tucker, C. "Social Networks, Personalized Advertising and Privacy Controls," MIT Open Access Articles, Journal, accessed on November 4th, 2020 :

https://dspace.mit.edu/bitstream/handle/1721.1/99170/Tucker_Social%20networks.pdf?sequence=1&isAllowed=y
This report reflects on how advertising data is shaped based on the user's browsing habits and preferences through the use of a randomized field experiment. The journal describes the effectiveness of the advertisements based on how much information the users provide to the website, and shows how proportionally likely it is for a user to explore an advertisement based on their browsing habits.

Xu L., Jiang C., Wang J., Yuan J., Ren Y. "Information Security in Big Data: Privacy and Data Mining," IEEE Access, Journal :
https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6919256
This journal elaborates on how privacy-preserving data mining (PPDM) can help resolve some of the privacy issues related to data mining. The journal also describes the four types of users involved in data mining applications and the various methods each user can employ to protect his or her sensitive information.

Contributions of Team Members

Team 8 worked together on the cover page, table of contents, abstract, introduction, glossary and references pages.

Sarah Cooper worked on the detailed design. She spent about 5 hours total.

Matthew Deagen worked with Marc on the high-level and medium-level design. He spent around 4 and a half hours total.

Leo Galang worked on the UI screenshots. He spent about 5 hours total.

Marc Ace Montesa worked with Matthew on the high-level and medium-level design, including the system-level diagram, part of the program units, and the data structures. He spent around 4 and a half hours total.