CISC 322

Assignment 3: Report

Google Chrome Feature: Face First

Friday, November 30, 2018

Group: Bits...Please!

Emma Ritcey *15er21@queensu.ca*

Kate MacDonald *14km90@queensu.ca*

Brent Lommen *brent.lommen@queensu.ca*

Bronwyn Gemmill *14bvg1@queensu.ca*

Chantal Montgomery *15clm1@queensu.ca*

Samantha Katz *12sk93@queensu.ca*

Abstract

Our team developed a new feature for Google Chrome called Face First. This feature allows password and user profile protection using facial recognition for user validation. We began our design process by establishing two approaches to implement our feature. We performed a SAAM analysis on both approaches which identified the Stakeholders and the advantages and disadvantages of each approach. Our final approach impacted the UI, Browser, and Memory subsystems in our conceptual architecture. Our team investigated both the high level and low level systems the feature would impact in the Chrome system. We determined the non-functional requirements that our feature would satisfy while also identifying the potential risks to the system as a result of our new feature. We identified various test cases to insure our implementation is sound within the Chrome system.

Table of Contents

Introduction to Face First

Have you ever tried logging into a website but you couldn't remember your password? If you answered yes, you're not alone! In 2017 it was discovered that the average email address in the U.S. is connected to 130 accounts with 25% forgetting a password a day. With each website having different specifications and requirements regarding character count, numbers, capitalization, etc it can be difficult for users to remember their password for each account. Although Google Chrome currently has a password manager that stores passwords, our team wanted to create a more secure and personalized solution for user information. This feature is called Face First!

When opening a Chrome browser all of the user's information is unlocked. If another person were to open this browser they could see the user's browser history, bookmarks, and have access to their passwords. Chrome currently asks the user if they want to store their password so it can be automatically re-entered when logging into that account. However, when that password is automatically re-entered Chrome does not verify that they are unlocking this private information for the correct user. Face First would add an extra level of security and a more private browsing experience.

Face First will allow Chrome users to unlock all of their personal information with their face. By utilizing the webcam built into the current device, this feature will perform facial analysis to authenticate the user. Once the user is identified, all of their personal settings and information will be visible. Personal settings include, browser history, passwords, and bookmarks. If the user was rejected, the Chrome browser would still be functional but no personal information would be accessible.

Our team previously analyzed Google Chrome to propose a conceptual and concrete architecture. Using the information we discovered while deriving those architectures, we investigate how our feature could be implemented into these pre-existing systems. We also utilize the Software Architecture Analysis Method (SAAM) to determine the most effective way to implement our feature. This analysis will allow us to access how our implementation will affect quality attributes such as performance and modifiability with respect to stakeholders. Our derivation process and logic with regards to effectively and efficiently implementing Face First will be further discussed in the report.

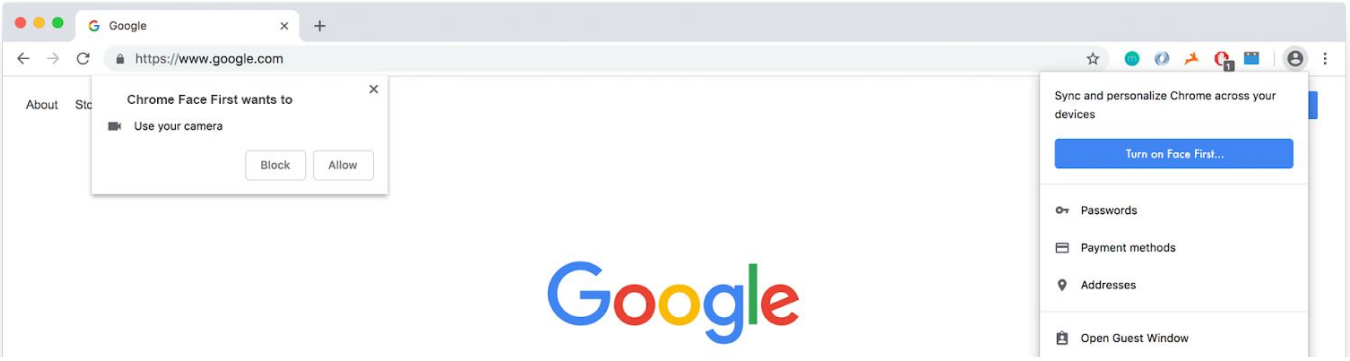

Figure 1. View when Chrome is opened.

Current State

After the Face First feature was proposed, it needed to be integrated into the current conceptual architecture. It was determined that the subsystems affected would be UI, Memory, and Browser. After discussion and performing the SAAM analysis (below) the proposed subsystems within our subsystems were developed and implemented into our conceptual architecture. In Figure 2 below, the current state of the conceptual architecture with the Face First features can be seen.

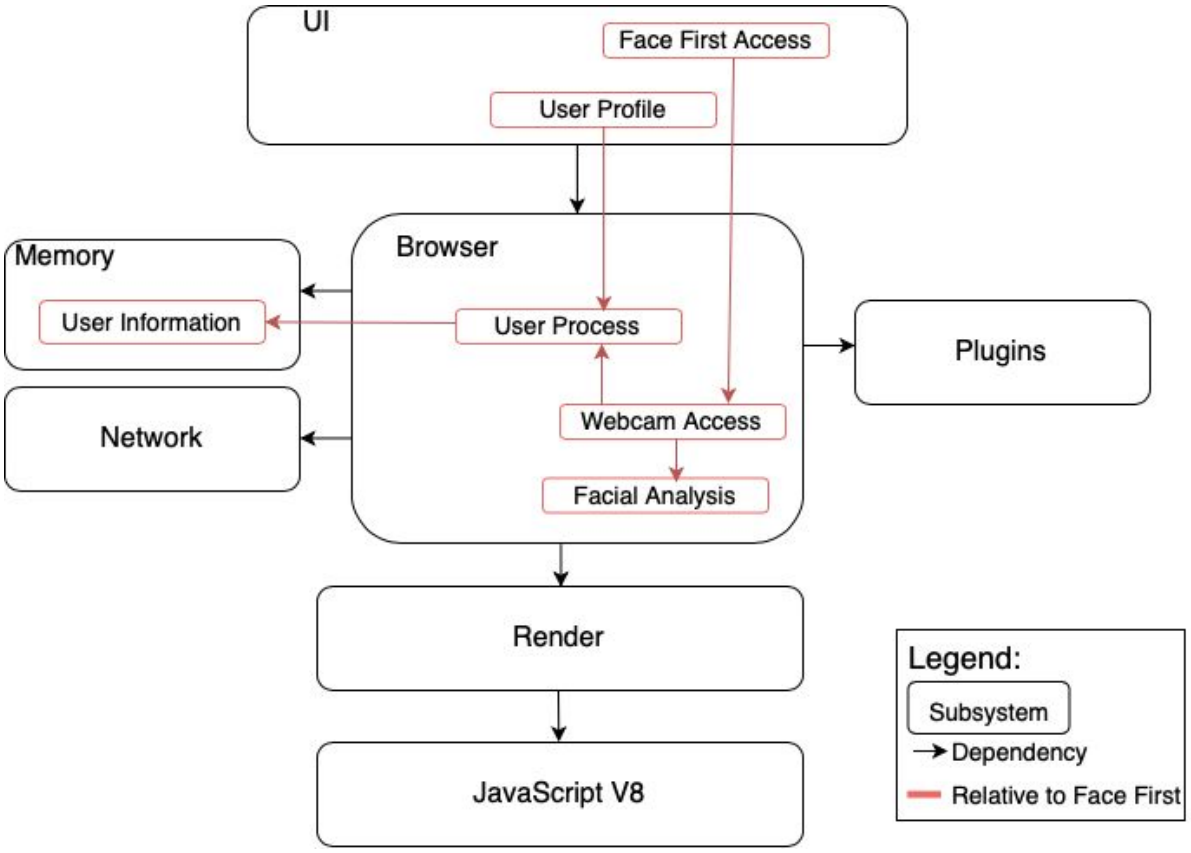

Figure 2. Current state of the conceptual architecture with Face First implemented.

Approach #1

The first approach to implementing the Face First feature was to have the user validation apply to the whole window. The user is validated through facial recognition in the Browser subsystem which then automatically logs the user into their account. Because it applies to the entire window, when the user opens a new tab in that window, they are automatically logged into their account on that tab as well. However, if the user opens a new window while currently logged into another window, they must use facial recognition to

access their account in that window. When the user closes the browser, they are automatically logged out of their account on that window.

The main advantage of this approach is maintainability. Since only three subsystems are affected when implementing this feature, the feature itself has low coupling when implemented in this way into our architecture. Decreasing the number of dependencies allows for easier implementation and code updating. The advantage of validating the user across the entire window allows for higher performance, as facial recognition is not needed for each tab that is opened. However, this results in decreased security, so our team had to determine whether performance or security was the most important characteristic of Face First before deciding on the final implementation.

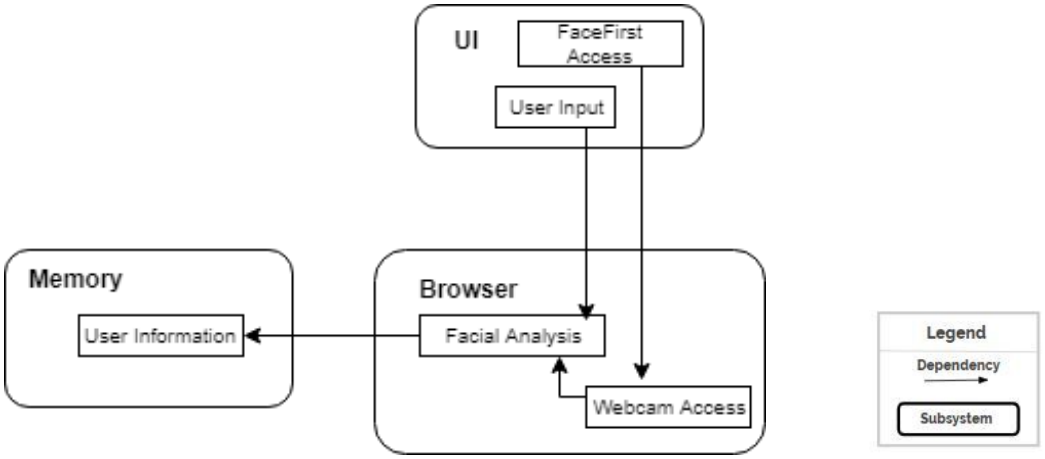

Figure 3. Approach #1 implementation of Face First in affected subsystems of the conceptual architecture

Approach #2

Our second approach for a possible implementation of the Face First feature for Chrome involves the feature being constructed as a plugin. In the second approach, the user is required to use Face First to log in to their Chrome account each time a new tab is opened, in addition to a new browser being opened. Chrome users are then automatically logged out of their Chrome account, each time a tab is closed.

The advantages of this implementation of the feature are primarily centered around security. By enforcing a new login with each new tab that is opened, the browser and the Face First feature do not assume that the same user is still using the browser, perhaps it may be a different user opening a new tab. Thus, increasing the security of the user's information. The tradeoff of this increased security for the feature is that it slows down the feature process. By forcing the user to login with each new tab, it adds time to the feature and the user. As well, as can be seen in the proposed architecture diagram, this implementation requires more subsystems to be involved. The involvement of more subsystems results in increased coupling in the architecture, which is not ideal. But, due to the increased security, our team explored the possibility of this implementation before deciding on a final implementation that

is still secure, but does not result in as many dependencies and will operate at an appropriate speed.

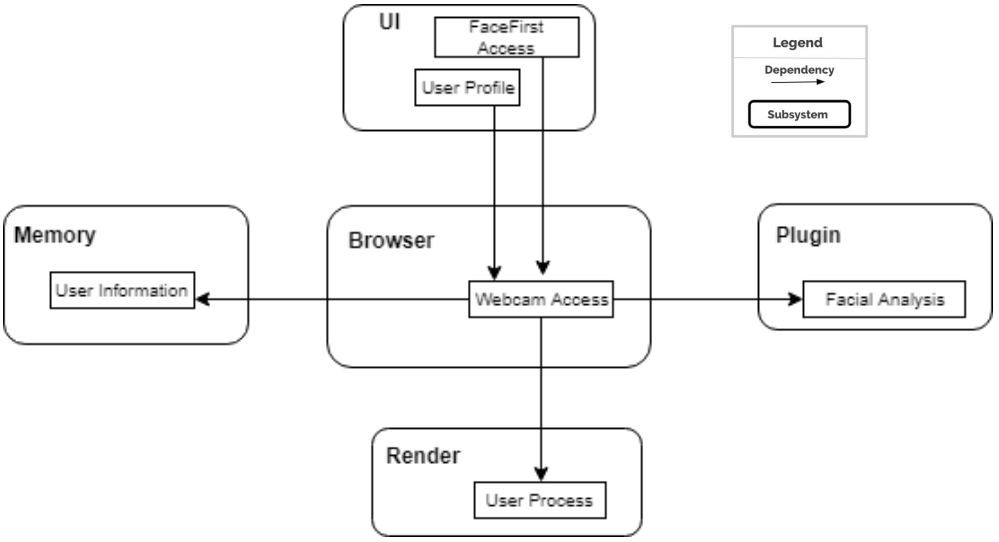

Figure 4. Conceptual architecture of Face First feature implementation approach number two.

SAAM Analysis

NFR	Effect (Approach #1)	Effect (Approach #2)
Security	Moderate. The authentication applies to the whole window, which means it is assumed that it is the same user opening new tabs within a window. If the highest level of security possible was the goal, this would approach would not be adequate.	High. By enforcing subsequent logins by the user, each time a new tab process is started, this approach is more secure. It does not assume that the same user will be opening a new tab in the same window.
Performance	High. The user does not need to authenticate their identity each time a new tab is opened. They only need to use facial recognition when a new tab is opened, which results in higher performance.	Low. Increased logins by the user each time a new tab is opened add time and negatively impact performance. Making this approach slower than the first approach.
Maintainability	High. This approach is implemented into three subsystems, which automatically increases the number of dependencies between and within the subsystems. However, relative to approach 2, this does not affect as many subsystems and therefore has higher	Low. This approach requires more subsystems to be involved in the feature's process, therefore increases the coupling on the system.

	maintainability as there is less coupling added to the system.	

Table 1. Effects of NFRs on each of approach to implementation.

Stakeholder	NFR Investment in Project (Approach #1)	NFR Investment in Project (Approach #2)
Users	This approach assumes users are concerned with having adequate security of their personal information while still maintaining the high performance of Chrome. They will be unlikely to use Face First if it negatively affects the speed at which they browse.	This approach assumes that users are more concerned with having high security of their personal information and having adequate performance of Chrome. They will be unlikely to use Face First if it does not have high security for their personal information.
Developers	Developers are most concerned with increasing the maintainability of the feature so that they can easily fix and enhance it in the future. At the same time they still want to maximize security and performance so that Google is satisfied with the product.	Developers will be concerned with the maintainability of the feature. They will want to minimize coupling and increase cohesion in order to create good software for the user to enjoy. As well, developers will be concerned with the security as they want the feature to be as secure as possible for users.
Investors	Investors would likely be highly concerned with security as they wouldn't want to be investing in something that will end up leaking users personal information. They would also be concerned with performance since they know people will not use the feature if it negatively impacts the speed of their browser.	Investors will be concerned with the security mostly, as they want users to feel safe using the feature, as well as for the feature to be secure to avoid possible issues that may arise if the feature is not secure.
Google	Google would be concerned with everything that their investors, developers, and users are concerned with, as this feature would not be possible without them. They want a feature that is easily maintained by their developers, but that also has high security and performance for its users and investors.	Google's primary concern is to create a successful feature for users, that is easily maintained by developers and meets the requirements of stakeholders. Thus, they will want the feature to be secure, and not impact the total architecture of the system. It must be maintainable in order to minimize coupling and increase cohesion.

Table 2. Effects of the project and NFRs that Stakeholders will have investments in.

After performing the SAAM analysis, we needed to decide whether high security or high performance was our main priority in implementing this feature, as these were the main differences between our two approaches from the users' perspective. Ultimately, we decided that Approach #1 was our best option as we believe users would not want to have to authenticate themselves each time they opened a new tab, even if it is more secure. By choosing the first approach, we also help the developers as this is a much easier feature to maintain once it is implemented. This will make code updating and enhancing much easier in the future.

Effect on Conceptual Architecture

Our feature will impact 4 components from our conceptual architecture: User interface, browser, memory and plugins.

User Interface

The user interface displays a request to the user to login to their account through facial recognition. If the user selects yes the UI will request access to the systems camera, then forward the request to the browser component to facilitate the authentication. In accordance to our original architecture the UI continues to only depend on the browser component.

The expected impacted files and folders for the user interface are: ui/display/manager, ui/login, ui/messae_center

Browser

Once the browser receives the Face First authentication request from the UI it will access the systems camera to capture an image of the user. The browser must then retrieve the original image of the user for comparison. The browser accomplishes this by sending a request to the memory component. Once the browser receives the original and captured images, they will be passed to the plugin component for validation. The browser's component dependencies remain unchanged from our original architecture, however the browser now has access to the systems camera resource.

The expected impacted files and folders for the browser are: memory_coordinator_impl.h, image_capture_impl.h, content/browser/permissions, components/password_manager/core/browser

Memory

The memory component will handle the request from the browser and return the users original Face First ID. All user specific memory such as bookmarks, search history and passwords are restricted until the user is authenticated. The memory component dependencies remain unchanged from our original architecture.

The expected impacted files and folders for memory are: content/child/memory, child_memory_coordinator_impl.h, child_memory_coordinator_impl_android.h

Plugin

The plugin component will receive the original and captured images from the browser to perform the validation of the user.

The expected impacted files for the plugin component are: about_signin_internals.h, account_fetcher_service.h, account_info_fetcher.h, child_account_fetcher.h.

Potential Risks

The implementation of our feature Face First comes with potential security risks. Giving the browser access to the systems camera resource opens up the browser to the possibility of being accessed without the user's permission. During development it is important to keep this in mind and create a sufficient test suite to insure unauthorized access is not permitted. A second security risk of our enhancement is the facial analysis function for authorizing user's. It is necessary that the quality of the facial analysis is sufficient enough to only authorize the user of the account. Developers will need to ensure that the facial analysis can not be tricked into giving access to the wrong user.

Testing Implementation

To determine if our feature has been implemented correctly its effect on Google Chrome and the functionality of the feature need to be tested. First, it is important to test that the feature is functional on the different operating systems Chrome is available on. For example, the feature should work on Linux, Mac and Windows operating systems. As well, it should be accessible through mobile devices like, Android and iOS. It is also important that the feature is adjustable based on the size of the page. If the page is scaled down or up the feat should adjust.

Second, the existing password manager should not be comprised. Previously saved password should be accessible but still secure. This can be tested by comparing the list of password before and after implementation.

Third, the browser speed should not be compromised. It is important to ensure that the user's browsing experience with Chrome is as efficient and easy before the feature was implemented. This will be tested by comparing the speed of the browser before and after the feature is included.

Lastly, the actual functionality of Face First will be tested. Test cases will be created based on our specifications to ensure all needs are met. It is also important that the Face First feature works for all types of users and can differentiate between users. As well, the feature should be able to recognize users wearing accessories like, hats and glasses.

Interactions with Other Features

Our Face First feature added to Google Chrome will interact with other features that have already been implemented in the system. The Face First system utilizes the Plugin subsystem to communicate with a facial recognition verification feature. This feature takes the information regarding the user's facial profile and returns a verification value for the system. We opted to use a third party facial recognition verification application to facilitate our overall feature.

Architecture Style and Design Patterns

When this feature is added to our conceptual architecture, the style remains Layered. We modelled our feature to be able to seamlessly fit into our conceptual architecture so that we would not have to create any entirely new subsystems. We maintained our layered conceptual architecture style in order to insure that we can increase cohesion within subsystems while decreasing coupling between our subsystems. It was valuable following the layered design pattern to insure we were implanting our new feature in a way that was compatible with the entire Google Chrome system.

Sequence Diagram

The sequence diagram below describes the use case of a User who wants to use Face First to login to their Chrome account. When a user first opens Chrome, they will receive a pop-up (as shown in Fig 1.) asking them for permission for Face First to use their webcam. The user will click *allow* and the message will be transmitted to the UI. The UI will make a function call to Browser to start Face First. Browser will call Memory to get the images (IDs) of the users who have allowed this computer access to their Chrome account previously. Memory will return the users. Browser will then enable the user's webcam and take a picture of the user. It will use this picture to perform facial analysis and compare the current user to the user's stored on the computer. In this use case, there is a match found for the user, so Browser passes the matched user ID to memory to retrieve the user's information. This information is then returned to the UI so that the user can view all the information associated with their Chrome account.

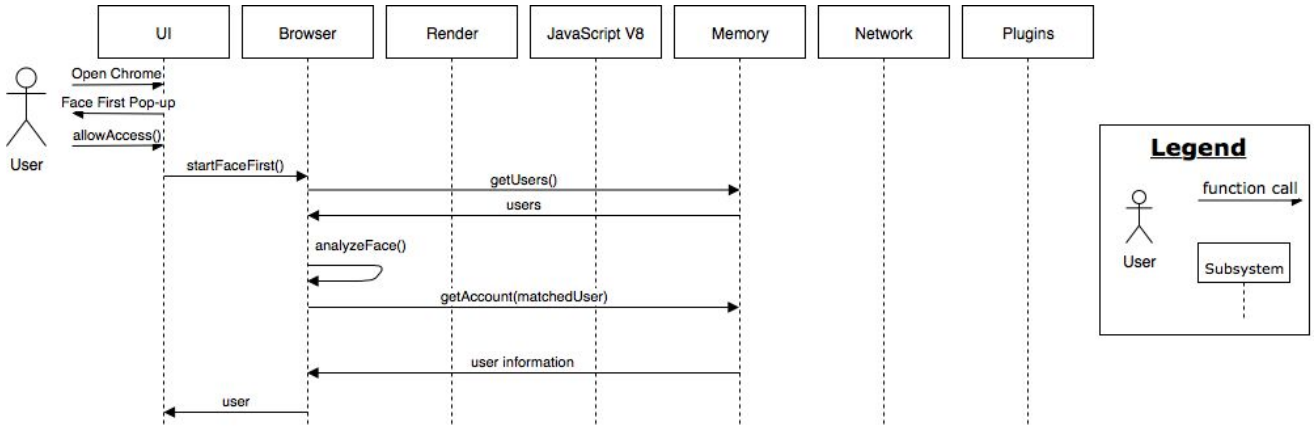

Figure 5. Sequence diagram for a user wanting to use Face First

Conclusion

After a significant amount of time spent brainstorming our new feature, we created a new feature called Face First. We investigated 2 approaches that maintained the overall functionality of Chrome and chose the approach that impacted Chrome in the most positive way possible. Our feature was focused on several non functional requirements including Security, Performance and Maintainability. We impacted 3 subsystems along with several code directories within these subsystems. As a team, this project showed us the steps for developing a new feature and the importance of collaboration as a group.

References

Uncovering Password Habits: Are Users' Password Security Habits Improving? (Infographic). (2018, February 05). Retrieved from https://digitalguardian.com/blog/uncovering-password-habits-are-users-password-security-habits-improving-infographic

Johnson, T. (n.d.). Forgot your password? You have too many and stores are losing business over it. Retrieved from https://www.kansascity.com/news/business/article156636084.html