

# UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

## Password Manager extension for Firefox

Emilios Savvides



Supervisor: Prof. George Papadopoulos

Co-Supervisor: Dr. Savvas Savvides

In partial fulfilment of the requirements for the degree of Computer Science at the  
University of Cyprus

May 2024

# Abstract

More and more aspects of our day-to-day activities are nowadays conducted online and a lot of personal information is available in various digital services. These services require authentication to keep our information available only to us, and a common way of authenticating to these services is through a username and a password combination. Using a strong password that is different for each such service is therefore very important. In this report, we document the design and implementation of a password manager extension for the Mozilla Firefox web browser that can be used to generate and store these passwords. By doing so, users don't have to come up with their own passwords which are oftentimes weak nor do they have to re-use the same password across multiple services which could be disastrous. We start by conducting an examination of existing password managers, to understand the features they provide and the interface that they use. We then design our password manager by choosing the most important features that are most commonly available in existing systems, and implement them using HTML, CSS, JavaScript, and by following the Firefox WebExtensions API. We then evaluate different aspects of our password manager such as the time needed to generate a password and the storage needs of the password manager. We conclude this report with a discussion of limitations and ideas for future work.

# Table of Contents

<b>Password Manager extension for Firefox</b>	i
<b>Abstract</b>	ii
<b>Table of Contents</b>	iii
<b>Chapter 1: Introduction</b>	1
<b>1.1 Overview</b>	1
<b>1.2 What are password managers?</b>	2
<b>1.3 Why are password managers useful?</b>	2
<b>1.4 About this project</b>	3
<b>1.5 Structure of this report</b>	4
<b>Chapter 2: Password Managers</b>	5
<b>2.1 Overview</b>	5
<b>2.2 Bitwarden</b>	5
<b>2.2.1 Description</b>	5
<b>2.2.2 Usability and User Experience</b>	6
<b>2.2.3 Security</b>	7
<b>2.2.4 Unique Features</b>	8
<b>2.3 1Password</b>	8
<b>2.3.1 Description</b>	8
<b>2.3.2 Usability and User Experience</b>	9

<b>2.3.3 Security</b>	10
<b>2.3.4 Unique Features</b>	11
<b>2.4 LastPass</b>	11
<b>2.4.1 Description</b>	11
<b>2.4.2 Usability and User Experience</b>	12
<b>2.4.3 Security</b>	13
<b>2.4.4 Unique Features</b>	14
<b>2.5 Comparison</b>	15
<b>Chapter 3: Design</b>	16
<b>3.1 Tool Architecture</b>	16
<b>3.2 Workflow</b>	17
<b>3.3 Sign-In Page</b>	18
<b>3.4 Home / Relevant Page</b>	19
<b>3.5 All Page</b>	19
<b>3.6 Create Page</b>	20
<b>Chapter 4: Implementation</b>	22
<b>4.1 Introduction</b>	22
<b>4.2 Manifest</b>	22
<b>4.2.1 Manifest_version</b>	22
<b>4.2.2 Name</b>	22

<b>4.2.3 Version</b>	23
<b>4.2.4 Description</b>	23
<b>4.2.5 Homepage_url</b>	23
<b>4.2.6 Icons</b>	23
<b>4.2.7 Permissions</b>	23
<b>4.2.8 Browser_action</b>	24
<b>4.3 Javascript Logic</b>	24
<b>4.3.1 Necessary Functions</b>	24
<b>4.3.2 Relevant Page</b>	25
<b>4.3.3 All Page</b>	26
<b>4.3.4 Create Page</b>	27
<b>4.4 Development Process</b>	28
<b>4.4.1 Initial Steps</b>	28
<b>4.4.2 Extension Basics</b>	30
<b>4.4.3 Real Progress</b>	31
<b>4.4.4 UI Improvements</b>	33
<b>Chapter 5: Evaluation</b>	35
<b>5.1 Evaluation setup</b>	35
<b>5.2 Memory Allocation</b>	35
<b>5.3 Password Generation</b>	36

<b>Chapter 6: Conclusion</b>	39
<b>6.1 Summary</b>	39
<b>6.2 Difficulties Faced</b>	40
<b>6.3 Plans for future work</b>	41
<b>6.4 Things I have learned</b>	41
<b>References</b>	43

# Chapter 1: Introduction

## 1.1 Overview

As the internet and online social accounts are getting involved more and more in our daily lives, the need for effective security and privacy while being online becomes increasingly more important. For this reason almost every website or application a person uses, it is required for them to use a secure password to enter. However, people tend to reuse the same passwords in several of their online accounts, something which is extremely dangerous, and people also tend to forget old passwords that haven't been used in a while.

To combat this, the use of a secure password manager gives solutions to all of the previously mentioned problems. Online users have the freedom of allowing the manager to take care of every log-in, by automatically filling out the sensitive information without the risk of a wrong input. For these reasons, we have decided to develop this kind of tool as a browser extension available for Firefox.

By examining and evaluating existing password managers we were able to extract the most crucial features that are needed for a secure and efficient password manager. It is important to pay attention to features and design choices that increase the quality of life for the user, as the main reason for someone to use such a tool is the convenience they provide.

This report will showcase in great detail the process behind designing a password manager, all of the implementation phases of the manager itself but also the necessary actions needed to be effective as a browser extension, and a thorough evaluation of storage space that our tool uses and generation times for various features.

## **1.2 What are password managers?**

Password managers are very important tools for a more secure and carefree online life. They aim to streamline security practices by taking the responsibility off of the users, allowing them to be secure online without them having to do much. A few features such tools provide are auto-completion of login forms, randomized generation of secure passwords, and many others.

These capabilities of password managers simplify every login process of users and negate the risk of using weak passwords, or even worse, reusing passwords across several accounts. Most password managers can have multiple devices linked to a user's account, so they don't have to worry about any of their devices being unsafe.

## **1.3 Why are password managers useful?**

Indeed, the hallmark of password managers lies in their ability to alleviate the burden of remembering numerous complex passwords by requiring users to only recall a single master password for access. This eliminates the risk of forgetting passwords or resorting to insecure practices such as using the same password across multiple accounts. Additionally, features like password generation and evaluation empower users to create robust passwords immune to brute force attacks, further fortifying their online security.

By taking into consideration all of the features that consist in a password manager it is quite clear to understand what makes such a tool useful in the daily lives of online users. Having the ability to remember only the master password of the manager and still gain access to every account a person might have, is a huge improvement for anyone. This by itself is an improvement for users, but the ability for managers to also generate a secure and unique password for every account the user has significantly improves security and privacy of users while being online.

These tools also provide a solid encryption protocol for every one of the user's credentials that are stored in their database. Credentials are never stored in plain



text so even in the extremely rare event of a breakthrough, attackers cannot access the sensitive information of users. The most important thing for password managers is providing security for their users, which means that encryption standards are improved often and security breaches are being accounted for the moment they might happen.

## 1.4 About this project

In this project, the primary focus will be on developing a web browser extension for Mozilla Firefox that functions as a basic password manager. We will allow users to add credentials manually if they please to do so, but also provide an integrated randomized password generation tool that can be customized to the user's preferences for a more secure password to be added to their credentials. An autocomplete function will also be integrated into the tool to allow users to more easily connect to their desired applications.

To design the password manager extension, we will first conduct a thorough investigation of existing password managers. This will allow us to discover the most commonly used features, along with any design choices that existing managers have that can be improved.

Upon completing the design and implementation of the password manager extension, we will perform a set of experiments to verify its performance and efficiency in terms of execution time of some functionalities as well as its storage capabilities.

It's important to note that this project will *not* incorporate a database for storing credentials, nor will it implement any encryption mechanisms. As a result, the emphasis will primarily be on developing functional aspects of the password manager rather than focusing on its security features. The aforementioned capabilities are therefore considered outside of the scope of this project and can be tackled as part of future work.

## **1.5 Structure of this report**

This report is structured as follows. After an introduction to the scope and goals of this project in this chapter, in chapter 2 we do a thorough examination of existing password managers including examining their design, ease of use, and features. In chapter 3 we explain the architecture and the overall design of our password manager extension tool. In chapter 4 we provide details about the implementation of our solution. In chapter 5 we talk about the development process followed to successfully complete the project and make it more user-friendly. In chapter 6 we evaluate our solution by running a number of experiments before concluding in chapter 7 with remarks on possible future directions.

# Chapter 2: Password Managers

## 2.1 Overview

In this chapter, we examine the three most prominent password managers, namely, Bitwarden, 1Password, and LastPass and provide detailed descriptions for each. Specifically, for each of the chosen password managers, we assess the usability and user-friendliness of each password manager by evaluating their interface design and ease of navigation. In addition, we list and describe the key features offered by each, identifying the most important ones between them and determining the essential functionalities that are necessary to every and any password manager.

## 2.2 Bitwarden

### 2.2.1 Description

Bitwarden [1] stands as a preeminent password management solution, distinguished for its sophisticated approach to digital security. In an era characterized by a proliferation of online accounts and the associated challenge of managing complex credentials, Bitwarden emerges as a robust and trustworthy ally in fortifying the integrity of sensitive information.

At the core of Bitwarden's efficacy is its commitment to top-tier security protocols. Employing advanced encryption methodologies, the platform ensures the impervious protection of user data. An adherent to a 'zero-knowledge' architecture, Bitwarden guarantees that user credentials remain exclusively in the hands of the user, elevating the platform's trustworthiness and instilling confidence in its user base.

Bitwarden distinguishes itself through its proficiency in password generation, providing users with the tools to create strong, unique passwords for each online account. This proactive approach alleviates the burden of memorizing intricate

password combinations. The platform also excels in simplifying the authentication process by automatically populating login credentials, optimizing user convenience without compromising security. Furthermore, Bitwarden extends its functionality to seamless form-filling, enhancing the efficiency of online interactions.

The accessibility of Bitwarden is noteworthy, with support for various platforms, including browser extensions, mobile applications, and desktop clients. This ensures a synchronized and user-friendly experience across diverse devices and operating systems. Complementing this accessibility is the option for two-factor authentication (2FA), reinforcing the security posture of the password vault.

Beyond password management, Bitwarden encompasses secure note storage, facilitating the safekeeping of sensitive information such as credit card details, Wi-Fi passwords, and important documentation. Additionally, the platform facilitates the secure sharing of login credentials, accommodating collaborative endeavors while preserving the confidentiality of user data.

Bitwarden emerges as an indispensable asset for individuals and enterprises alike, seeking a blend of robust security measures and seamless digital operations. Its intuitive interface and stringent security protocols position it as an attractive option for those prioritizing both user experience and data protection. With flexible pricing options, including free and premium tiers, Bitwarden caters to a broad spectrum of users and organizations. Whether fortifying personal online security or augmenting an enterprise's digital defense strategy, Bitwarden stands as a versatile and reliable choice.

### **2.2.2 Usability and User Experience**

Installation and setup for new users of the manager is easy and simple. Bitwarden provides clear instructions to follow and explains the uses provided by it thoroughly. The User Interface of Bitwarden is easy to understand and is organized in a way that makes sense. Adding passwords and other information to the manager is easy and intuitive. The Auto-fill feature for logins in websites is not obvious how it works and requires a specific key bind to use which is hidden inside the settings.

### **2.2.3 Security**

Bitwarden sets a gold standard in digital security through its comprehensive encryption protocols. Utilizing AES-CBC-256, all data in Bitwarden is end-to-end encrypted, ensuring an impenetrable layer of protection. The master password undergoes encrypted salted hashing with email and PBKDF2-SHA256 encryption, subsequently stretched to 512 bits using HKDF. Notably, the client assumes the pivotal role of key creation and management, employing a zero knowledge-based encryption approach.

Data sharing between Bitwarden users incorporates both asymmetric and symmetric encryption methods, ensuring secure transmission and reception. The platform adheres to the highest standards of transparency and compliance, as evidenced by its open-source code, in conformity with AICPA SOC2/Type 2, Privacy Shield, GDPR, and CCPA regulations.

User authentication is fortified by requiring the master password and the chosen 2-factor authentication method for every login. Client-server communication is fortified through TLS/SSL and HTTP/HSTS, with encryption occurring locally before transmission, fortifying the confidentiality of data in transit.

Bitwarden's commitment to proactive security is evident in its provision of regular security reports, addressing threats such as Exposed/Reused/Weak Passwords, insecure websites, and potential data breaches. Organization creation involves the generation of a symmetric key using CSPRNG, encrypted via RSA-OAEP, exemplifying the platform's commitment to robust security measures.

For user participation in an organization, a stringent approval process ensures the creation of an organization key using the user's public RSA key, encrypted with the organization's symmetric key. Any exported data undergoes encryption using API and CLI, adding a layer of security to shared information.

All data is securely stored in Microsoft Azure, with routine vulnerability assessments conducted using CloudFlare and proxies. Access controls are rigorously implemented, with employees granted the minimum access necessary for

their roles. Any change requests undergo thorough review and approval processes managed by authorized personnel.

Bitwarden proudly holds SOC 2 Type 2 and SOC 3 certifications, reinforcing its commitment to maintaining the highest standards of security and compliance within the industry.

## **2.2.4 Unique Features**

Bitwarden stands out from its competition by having an open-source security model that allows everyone to have access, review, and contribute to its improvement. This allows transparency between user and manager at the cost of possibly being exposed to hackers, but the continuous evolution of the source code makes it hard for them to keep up with, and having every possible user who might know about cyber security makes the developer team work more efficiently and gives them new ideas.

Another feature not found in competitors is that users can have emergency access to their vaults without the help of an administrator or any other user, making it easier for account recovery or in the event of a lockout.

## **2.3 1Password**

### **2.3.1 Description**

1Password [2] stands as a premier solution in the realm of password management, distinguished for its sophisticated approach to digital security. In an age defined by an escalating number of online accounts and the inherent challenges of managing complex credentials, 1Password emerges as a robust and sophisticated ally, dedicated to fortifying the integrity of sensitive information.

The cornerstone of 1Password's effectiveness lies in its unwavering commitment to the highest standards of security. Employing advanced encryption methodologies, the platform ensures the impregnable protection of user data. Committed to a 'zero-knowledge' architecture, 1Password affirms that user

credentials remain solely within the control of the user, thereby enhancing the platform's credibility and engendering trust among its user base.

1Password excels in the generation of strong, unique passwords for each online account, alleviating users from the burdensome task of memorizing intricate password combinations. The platform further simplifies the authentication process by seamlessly populating login credentials, optimizing user convenience without compromising security. In addition, 1Password extends its functionality to streamlined form-filling, enhancing the efficiency of online interactions.

The accessibility of 1Password is noteworthy, with support across various platforms, including browser extensions, mobile applications, and desktop clients. This ensures a synchronized and user-friendly experience across a diverse array of devices and operating systems. Augmenting this accessibility is the provision of two-factor authentication (2FA), bolstering the overall security of the password vault.

Beyond its core function of password management, 1Password encompasses secure note storage, providing a secure repository for sensitive information such as credit card details, Wi-Fi passwords, and critical documentation. The platform also facilitates the secure sharing of login credentials, accommodating collaborative efforts while preserving the confidentiality of user data.

1Password emerges as an indispensable asset for discerning individuals and enterprises seeking a harmonious blend of robust security measures and seamless digital operations. Its user-friendly interface and uncompromising security protocols position it as a compelling choice for those prioritizing both user experience and data protection. With flexible pricing options catering to individual and business needs, 1Password stands as a versatile and reliable solution for fortifying personal online security and elevating an organization's digital defense strategy.

### **2.3.2 Usability and User Experience**

Installation and setup for new users of the manager is easy and simple. The User Interface is easy to use and understand. The manager helps new users get started by showing them the capabilities of it and having messages showing users

how to import data and create new vaults. In the vault itself, everything is organized and easy to find. Every password is reviewed for its strength and any reused passwords or compromised websites can be found easily. 1Password explains how to use the autofill function, which is just a button that appears in the filling box for each website, basically, a button of the manager's logo that auto-fills the data for the user when pressed.

### **2.3.3 Security**

In the realm of security, 1Password boasts a robust model with encryption practices. Employing AES-256, the platform ensures end-to-end encryption for all data. Notably, encryption keys, initialization vectors, and nonces are generated using a cryptographically pseudorandom number generator, rendering brute-force guessing or hacking practically impossible. To fortify against brute force attacks, data undergoes encryption using PBKDF2-HMAC-SHA256, providing an additional layer of formidable protection.

Each user is assigned a unique 128-bit key crucial for account activation and the addition of new devices. This key setup acts as a potent deterrent against remote access attempts by hackers, enhancing the security of individual vaults for website logins. 1Password employs Secure Remote Password (SRP) to ensure that passwords are never transmitted over the internet, mitigating exposure to potential trackers. As an added security feature, the platform supports biometric access if the user's device is equipped with such capabilities.

The security architecture of 1Password extends beyond the code itself. Master passwords and user data are stored separately, a strategic measure to thwart potential exploitation in the event of a breach. Importantly, passwords and secret keys are never sent unencrypted to the manager's servers; rather, 1Password retains only encrypted copies. All information is stored locally on each device, reinforcing the decentralized nature of its security infrastructure.

Further enhancing its security posture, 1Password takes proactive measures such as automatically clearing passwords from device clipboards, validating the



authenticity of the user's browser before processing any information, and alerting users to potential entry into compromised websites. These comprehensive security practices collectively position 1Password as a stalwart guardian of user data in the digital landscape.

### **2.3.4 Unique Features**

1Password stands out from the rest by giving developers tools such as signing git commits, generating secure SSH keys and even storing code in the secure vault of the manager. Normal users can also benefit from sharing data with people who do not use 1Password as their password manager making it convenient for data sharing. In terms of security, users can use masked email for their communications making it harder for unwanted third-party trackers to find their email. Finally, in shared vaults other users can help in the event of a lockout by helping in account recovery eliminating the hassle and time needed to go through a special operative from the manager or not being able to sign in at all.

## **2.4 LastPass**

### **2.4.1 Description**

LastPass [3] is a leading-edge password management software and service renowned for its adept handling of digital security. In an era characterized by an ever-expanding digital footprint and the consequent proliferation of online accounts, LastPass emerges as an indispensable solution for the astute management and safeguarding of sensitive credentials.

The cornerstone of LastPass's value proposition is its unyielding commitment to data security. Employing robust encryption protocols, the platform serves as a fortress for users' confidential information. Notably, LastPass adheres to a 'zero-knowledge' approach, which assures that the service provider is devoid of any access to users' stored data, further fortifying the level of security and conferring a profound sense of trust upon its users.

A distinguishing feature of LastPass is its proficiency in password generation. The platform not only safeguards your passwords but actively contributes to their creation. It facilitates the generation of strong, unique passwords for each online account, obviating the need for users to memorize intricate sequences of characters. Additionally, LastPass streamlines the login process by automatically populating login credentials, simplifying and expediting the authentication experience. This extends to form-filling, making e-commerce and web interaction frictionless.

LastPass boasts versatility in its accessibility, being available as a browser extension, mobile application, and desktop software, ensuring a seamless and synchronized experience across a spectrum of devices and operating systems. Augmenting this versatility is the provision of two-factor authentication (2FA), adding an extra layer of security that safeguards the integrity of the password vault.

Beyond password management, LastPass encompasses secure note storage, accommodating sensitive data such as credit card information, Wi-Fi passwords, and software licenses. Moreover, the platform facilitates the secure sharing of login credentials, a boon for individuals and enterprises looking to collaborate without divulging confidential information.

LastPass is an unimpeachable asset for those who demand stringent security protocols and the streamlining of digital operations. Its user-friendly interface, coupled with robust security measures, makes it an attractive proposition for individuals and enterprises alike. Offering both free and premium tiers, it caters to a diverse range of users and organizations. Whether the objective is to safeguard an online identity or fortify a business's security posture, LastPass emerges as a versatile and dependable choice.

## **2.4.2 Usability and User Experience**

Installation and setup are easy like every other password manager, the user only needs to create an account and follow the instructions provided on the screen to create and access a secure vault. The user interface is customizable in terms of the way it shows vaults and passwords to appeal to each specific user. It motivates

new users to increase the security of their vaults by giving them a set of tasks which after completed gives them a small discount on their next subscription. Creating folders and adding data to the vaults is straightforward, the user has to find the button to add an item and do what he needs. To enter a website that is already added to the vault the user only needs to click on the item that stores the passwords and LastPass will automatically redirect the user to the website already logged in alternatively, the user can go manually to each website and inside the username and password boxes there will be the symbol of LastPass when pressed will log the user in. In the account setting, there are a lot of quality-of-life features that greatly help the user in day-to-day tasks. Importing and exporting data from LastPass is easy to do but the option to do this is hidden inside the account setting under advanced options. The manager has browser extensions and a mobile application.

### **2.4.3 Security**

LastPass's security model is based on having zero knowledge of encrypted and decrypted user data. The master password is first hashed with 600000 rounds of PBKDF2-SHA256 and a round of Authentication hash. After that, it is sent to the servers of LastPass where it undergoes another round of hashing using the authentication key and is stored. Data is hashed with 600000 rounds of PBKDF2-SHA256 as well but after it is encrypted using the encryption key using AES-256 encryption. This data is then stored in the local vault of the device and an encrypted vault in the manager's servers.

To log in each user needs to use the two-factor authentication method or use biometric features if the device supports it. Vaults can be accessed in offline mode as well and personnel have the minimum access required to do their jobs, to prevent possible intrusions from the inside. LastPass has the term "Super Admins", who have a copy of a user's local key to help them recover their accounts and reset their passwords in the event of a lockout.

LastPass also has Intrusion detection systems on its servers a local OWASP firewall, proxy servers to prevent DDoS attacks, and regular system and file monitoring.

There have been several incidents regarding security in the past [4], which in some cases resulted in vault theft. In 2015 there was a security breach regarding suspicious activity in the network. No breaches were found from this, only a few account email addresses, password reminders, server-per-user salts, and authentication hashes were compromised. In 2021 it was discovered that LastPass has integrated third-party trackers in its applications to better understand product trends and patterns and to improve user experience and performance. These trackers still exist in the application and LastPass commented that users have the option to disable them in the 'Advanced Settings'. In 2022 there was suspicious activity in the development environment of the manager and an investigation was started immediately. The LastPass team found an unauthorized third party trying to gain access to the source code through a compromised device of a senior developer but the activity has been stopped before any incident could happen. However, a few months later this led to a security breach taking place through a cloud-based storage service which had some account information and metadata stolen from hackers, including company names, user names, billing addresses, email addresses, telephone numbers, and IP addresses, but LastPass has stated that there was no evidence that any encrypted data was accessed. This means that stolen data are still strongly encrypted and is extremely difficult for hackers to use a brute force method to guess master passwords and the keys of the users.

#### **2.4.4 Unique Features**

LastPass stands out from its competitors for the usability and user experience it provides. Logging in to any website directly from the password manager and the customizable interface makes everything easier. It also has some extra Multi-Factor Authentication add-ons that improve security and usability for an extra price and help

businesses provide them with customizable security policies for their users to follow. In general, LastPass is more optimized and offers more to businesses compared to its competitors.

## 2.5 Comparison

Name	Storage	Two-Factor Authentication	Security Reports	Devices	Credential Grouping	Shared Vaults	Encrypted Sharing
Bitwarden	Unlimited	Yes	Yes	Unlimited	No	Yes	Yes
1Password	Unlimited	No	Yes (Paid)	Unlimited	Yes (Paid)	Yes	Yes
LastPass	Unlimited	Yes	Yes	Unlimited (Paid)	Yes (Paid)	Yes	Yes

Name	Notes	Biometric Login	Support	Password Generator	Developer Tools	Open-Source
Bitwarden	No	No	Yes	Yes (Paid)	No	Yes
1Password	Yes (1GB)	No	Yes	No	Yes	No
LastPass	Yes (50MB)	Yes	Yes	Yes	No	No

# Chapter 3: Design

## 3.1 Tool Architecture

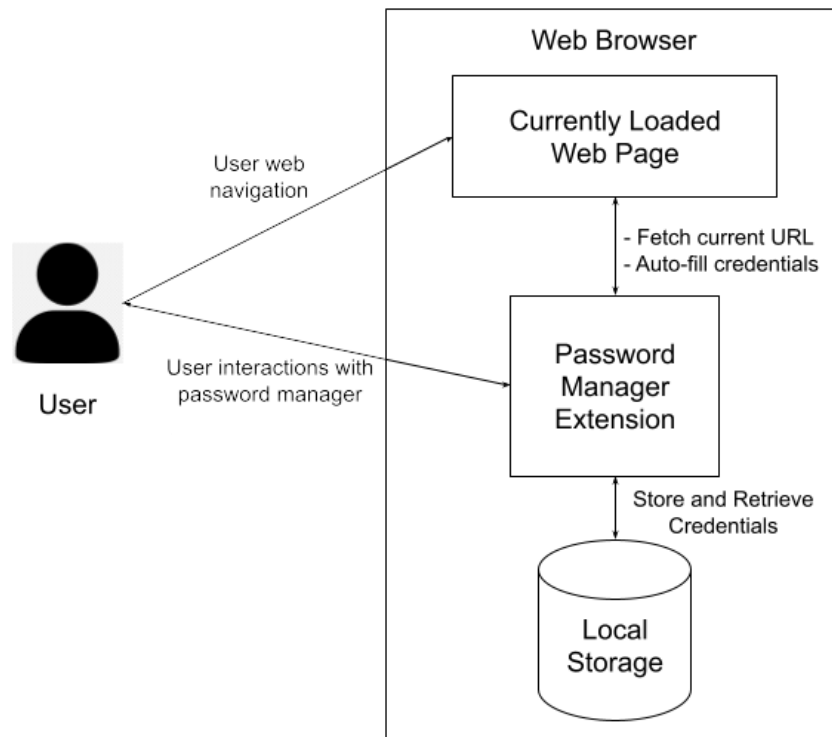


Figure 3.1 The extension's architecture

To develop our system we will follow a simple design based on the architecture show in figure 3.1. The user will be navigating any web page they want, and our system will extract useful information from the web page they have currently loaded. Now the user can communicate with our password manager where they can execute functions directed to their web page, like the autofill function, or store and retrieve their personal credentials from the browser's local storage.

## 3.2 Workflow

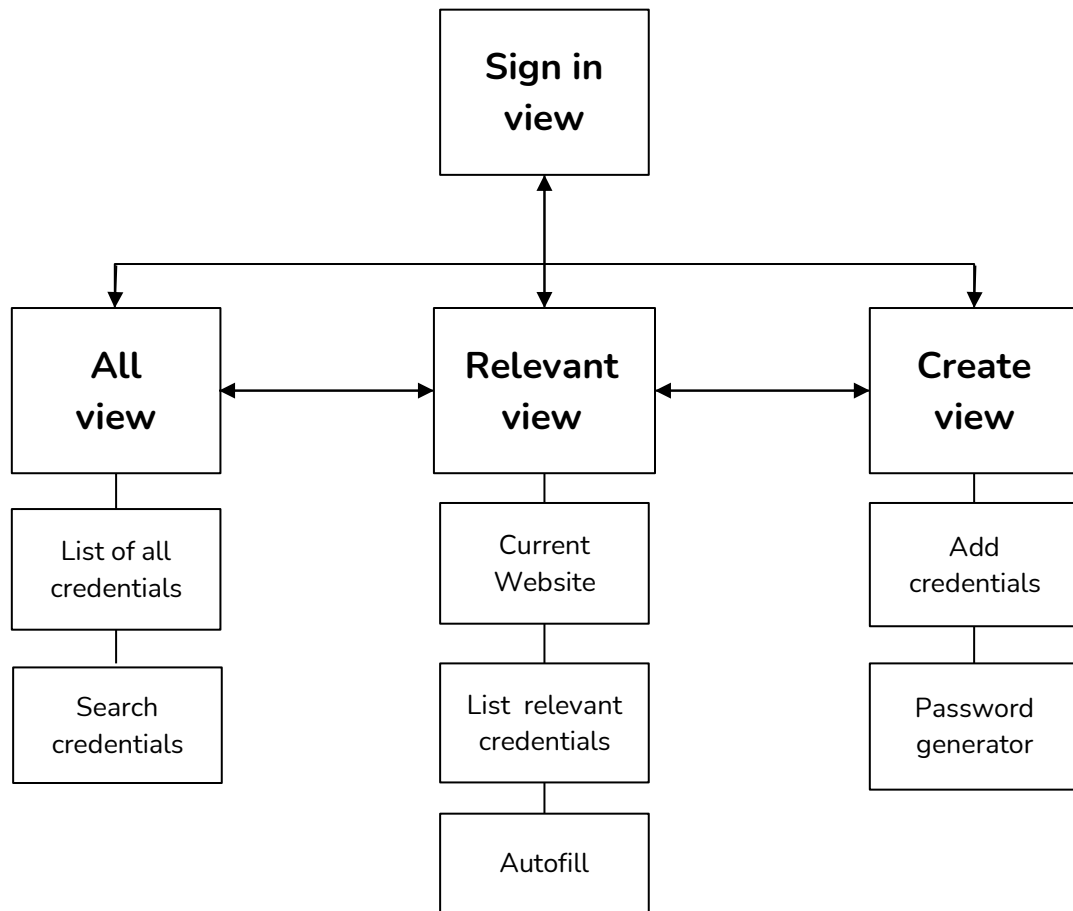


Figure 3.2: The workflow of the Password Manager

Based on the exploration we have done in the previous chapter, we determined that a good structure for our password manager extension is one that contains four distinct views, with each view designed to serve one or more key features. The first view is a simple one that allows users to sign-in, and then seamlessly redirects the user to the Relevant view. The Relevant view is the first view shown to the user when the user signs in because it contains the most commonly used features. Specifically, this view displays the relevant credentials pertinent to the user's current web page, and it provides the essential feature of auto-filling the credentials to forms detected on the current web page.

Next to the Relevant view, the All view provides a full view of all user credentials. This view provides search functionality to enable users to easily locate specific credentials within their stored list of credentials.

The next view is the Create view which allows users to create and store new sets of credentials. The interface of this view is a simple one made primarily of a single form to facilitate the seamless addition of new credentials. A crucial feature contained within this view is the password generation tool. This tool is customizable so that it can satisfy the requirements of individual users, both functional and security requirements.

### 3.3 Sign-In Page

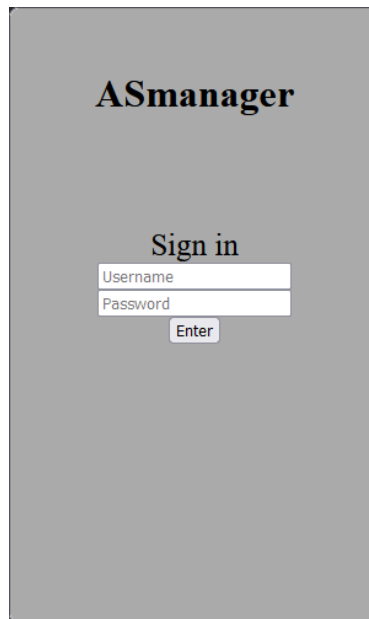


Figure 3.3: The 'Sign In' page the user sees upon opening the extension

This page is isolated from the rest as its purpose is to provide a secure access point to the functionalities of the password manager. Users are required to fill out both input fields to authenticate them, with the correct error messages being displayed in the event of a mistake or a missing field. After a successful login, the user is redirected to the 'Relevant' page, where they can access the password manager from.



### 3.4 Home / Relevant Page

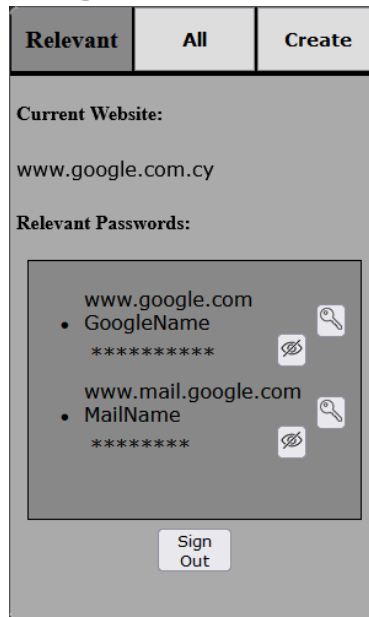


Figure 3.4: The 'Relevant' page with all the credentials relevant to the user's web page

Upon entering the extension, users are presented with the 'Relevant' page, which showcases all of the user's credentials that are relevant to the website they are currently on. For each of the credentials that are deemed relevant, an autocomplete button is created for the user, which allows the automatic completion of login forms. Along with the previously mentioned button another password visibility button is created to show or hide the password for each entry of credentials.

Since this page also functions as the 'Home' page of the extension, at the bottom of the page the 'Sign Out' button is located to allow users to safely exit the password manager, without fear of any security leaks. This button will redirect users back to the 'Sign In' page where their login credentials will be asked again for re-entry.

### 3.5 All Page

The 'All' page is designed to be a simple page that showcases all of the credentials the user has stored in the password manager. The moment the extension

is loaded in the browser, a single list is created with all of the information of the credentials and is sorted based on the URL of each entry. Each entry also has a button integrated with it to show or hide the password, but unlike the previously mentioned page, there is no feature for auto completion of web forms to prevent any accidental credential sharing.

Users also have the ability to search for their credentials based on the URL of the entries. To achieve this a sublist is shown by the manager with all of the credentials that contain the searched string. Search through username or password is not supported for security reasons.

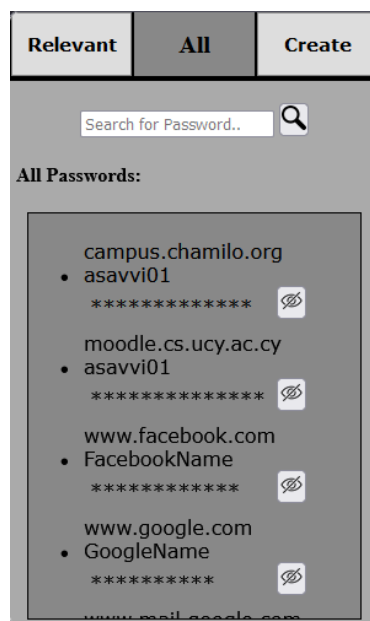
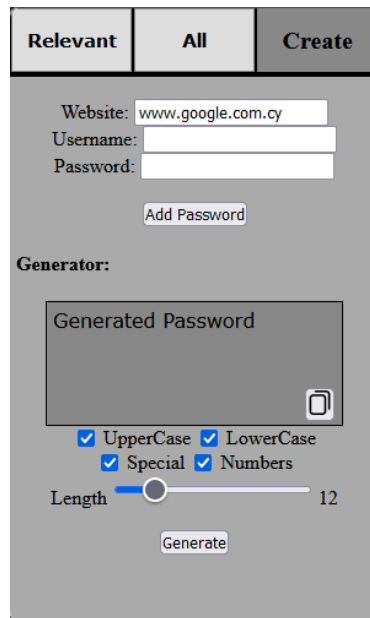


Figure 3.5: The 'All' page where all of the user's credentials can be viewed

### 3.6 Create Page

The final page of the password manager is where users can add credentials to their account by filling out the appropriate forms. The website URL is automatically extracted from the current web page the user is located in but can be manually changed, the username is free to be customized as the user pleases and so is the password of the credentials, but the password generation tool located just beneath this encourages users to create a more unique and secure password, which can be copied in the form with the click of a button.

The randomized password generation feature provides users with a set of several 'modes' they can run the generation to better suit their preferences. These modes consist of Uppercase and Lowercase Latin characters, single-digit numbers, and a set of special characters, with a customizable length that can be set to any value between eight and thirty-two characters. The numbers were chosen to provide users with at least the minimum amount of security from brute force attacks in the lowest values.



The screenshot shows a web interface with three tabs: 'Relevant', 'All', and 'Create'. The 'Create' tab is active. It contains a form for adding credentials with fields for 'Website:' (pre-filled with 'www.google.com.cy'), 'Username:', and 'Password:'. Below these fields is an 'Add Password' button. Underneath is a 'Generator:' section with a 'Generated Password' box containing a copy icon. Below the box are four checked checkboxes: 'UpperCase', 'LowerCase', 'Special', and 'Numbers'. A 'Length' slider is set to 12, and a 'Generate' button is at the bottom.

Figure 3.6: The 'Create' page which contains the password generation tool and the ability to add credentials

# Chapter 4: Implementation

## 4.1 Introduction

To implement our system we generally used HTML, CSS and JavaScript. The system also must be implemented as a Firefox extension, and to achieve this a manifest file has to be created which contains all of the extensions options and properties. An object oriented design was also followed during the development process, as well as focusing on creating the different pages dynamically, to accommodate for any changes in the list of credentials that will be displayed on the screen at any time.

## 4.2 Manifest

### 4.2.1 Manifest\_version

This field indicates the version of the manifest that the extension will be running on. By choosing version 2 for our project, we ensure stability without the fear of any bugs occurring from newer versions, which is important as we are working with sensitive information. Everything needed to create a password manager can be done with version 2 so we are not limited by the older technology.

### 4.2.2 Name

The name indicates the name which will be visible to users using the extension. We have chosen to use the name ASmanager, using my initials to show that it is a personal project and to keep it simple.

### **4.2.3 Version**

Unlike the `manifest_version` this field shows in what version of development the extension is currently at. Currently, it's only at version 1.0 because this is the first working version of the manager.

### **4.2.4 Description**

This field gives a brief description of the password manager and all of the capabilities of the extension, like the ability to add and remove any account to the manager, generate random passwords depending on the preferences of the user, and auto-fill sign-in forms on websites with the correct information.

### **4.2.5 Homepage\_url**

This field is reserved for the website of the extension. In this instance, the link directs users to the GitHub page of the project, which gives viewing access to all of the source code and materials used to create it.

### **4.2.6 Icons**

This is a link to the icon, which the extension will display for the users for easy recognition on their extensions page. A shield in the colour scheme of the password manager was chosen to indicate a service which offers protection.

### **4.2.7 Permissions**

The permissions field is the most important in the manifest file, as it is the declaration of all the permissions that the extension requires to function properly in a web browser. We use the `activeTab` permission [5] as it is important to get access to the user's current tab they are working on specifically so we can use the autocomplete function and the `all_urls` permission as we do not want to be restricted into which websites the extension can work and give the freedom to users to use the extension in any scenario they want.

## **4.2.8 Browser\_action**

Here are the initialized values that the extension will use at the beginning of its execution. Specifically, the icon is shown on the top right among the other extensions, the title is what is shown in the extensions tab of the browser's bar and the default popup is the HTML page which the extension will open when clicked.

## **4.3 Javascript Logic**

### **4.3.1 Necessary Functions**

Each of the user's credentials are stored as a javascript object with three variables, which consist of the URL where the account is supposed to be used, the username of the account, and the corresponding password. The constructor for this object only works when there are parameters for each field as there cannot be a default value for any of them. Finally, an array called 'info' is created where each user's objects of type credentials are stored. This is done so it is easier to access all of the information for each entry, making sorting and selecting credentials in all functions much cleaner and more efficient.

For testing purposes, this array of objects is initialized with some dummy information the moment the window is loaded on the screen. All of the information is stored in the localStorage of the browser where it is kept safe and consistent every time the user logs off or closes the extension window. In the case there is already information in the array then the extension just loads the information from the localStorage to the array to use.

On window load time some necessary functions are also executed to get supplementary information either on screen for the user or for other functions to work properly, specifically getting the current URL the user is in, to use it later on in other functions. All of the event listeners are created here as well and are ready to be fired whenever they are called.

To get the URL the user is in, a function that uses the browser.tabs API is used to get access to it and save it in a variable as a string. That string is later called

in other functions or displayed in the specified fields of the extension as a hostname. Also, to get the maximum use out of the URL on the 'Relevant' page it has to be cleaned, which means removing all of the URL extensions from it and leaving the domain name.

The functions that allow the user to traverse the three pages of the extension are fairly simple. They are almost identical in the sense that every 'div' element which holds the information for each page from the HTML file is given the hidden attribute, except for the 'div' that the user wants to view.

### **4.3.2 Relevant Page**

The 'Relevant' page, other than being the home page of the extension, also has one of the most important functionalities of the project. That functionality is the autocomplete feature. To let the user have this ability first a list of the relevant credentials in the info list needs to be displayed on the screen.

To do this three dynamic arrays are created, each with a specific use. The first one is used to hold the credentials objects that are relevant to the current URL of the user, the second one is used to be able to know whether the user has their password visible on the screen, and the final one is used to store the password in hidden form, meaning that a string of '\*' characters are stored that have the same length as the original password. These three arrays function parallel to each other, meaning that the n-th element of each array corresponds to the n-th element of the other arrays.

The function gets as a parameter a cleaned version of the user's current URL and after emptying the list it saves all of the credentials from the info list that include the string of the cleaned URL, initializing all of the passwords as not visible and saving the length of the password for each entry. Then all of the relevant items are dynamically added to the list of the page.

To achieve this some variables are created which are HTML elements and appended in a 'li' element to be viewed in order on the screen. The size of each list item is determined by the size of the password as that is the only field that can vary

in length between different credentials. The password for each item is contained in a separate 'span' element, to have the ability to change the contents of it without disrupting the rest of the information. A button is appended to this element which is created inside the function and has the functionality to replace the password with a string of '\*' characters and vice versa. Each time the button is pressed the value in the 'eye' array of the corresponding object is flipped to keep track of whether the password is visible or not.

A second button is also created in this function which is appended at the end of each list item and has the functionality to autocomplete forms on the user's website with the credentials that are stored in the current list item. This is done using the browser.tabs API which allows the extension to inject and execute a script in the web page. Specifically, a variable is created in the script which holds the information of the credentials for each list item and later calls the 'login' function with the credentials as a parameter. This function is nothing more than a search on the web page to find common names of elements that might be input fields for login information and it gives them the values taken from the credentials.

This page also has the sign-out button since it is the home page. This button is located below the relevant list displayed on the screen and only has one functionality. Exit the manager go back to the sign-in page and set the session in the localStorage as 0.

### **4.3.3 All Page**

The simplest page of the three is only used to display all of the currently saved credentials of the user and sort them, if needed, according to the user's search input. The show password function is called when the window is loaded, as previously mentioned, and just sorts the credentials alphabetically as indicated by the compare function, and then generates the list that contains them.

The compare function is a simple check to see which two inputs, meaning the website from each credential, are first in the alphabet and returns the choice accordingly for sort to work properly.



The methodology of the list generation is very similar to the one above, only in this instance there is no need for the creation of the autofill button, as the user only has that option with relevant passwords for increased security. The same HTML elements are created where each list item has span elements containing the information for each entry and everything is appended together to create a list. The size of each element is determined by the size of the password and still has the button to show and hide the password on the screen.

The user can search for their credentials based on the website they are looking for by typing a string of characters that matches their search in the input bar. This works by creating a new list with all the credentials that have the inputted string, included in the string of the stored URL, and generating the new list with the function mentioned above. In the case where the input bar is empty, the function is called with all of the stored credentials and displays all of them on the screen. The search is executed by pressing the button or pressing the key 'Enter' in the search bar, which calls the aforementioned function.

#### **4.3.4 Create Page**

The third and final page of the extension is used to add credentials to the user's account and generate randomized passwords with the preferences of the user. The function that creates a new credential is fairly simple, checking whether the three required fields are filled and the password is at least eight characters long, if so, then a new object is created with the three fields as the input and added to the list and localStorage.

The creation of a randomly generated password is done by first finding the user's preferences depending on the checkboxes they selected and the length of the password they want according to the slider. The four checkboxes correspond to the Latin letters, both lowercase and uppercase, one-digit numbers, and special characters. The slider can be modified with values between 8 and 32, with the specific number currently selected being shown on the screen simply by changing the value of an HTML element on the screen.

If there is no checkbox selected an error message warning the user is shown on screen and the function is terminated, otherwise depending on the selection a set of randomized numbers are created, which are given padding and range to correspond to the correct range of characters in the ASCII code. This allows the function to have a random character from each set of strings and then randomly decide between them to select only one. The selected character is then added to the string of the generated password, and the process is repeated as many times as the slider for the length indicates. All of the possible combinations of the four checkboxes are in separate if statements but the logic behind them is always the same.

The user can also copy the generated password directly in the input field of the credentials addition form with the press of a button. The function first has to make sure a valid string is contained in the generated password box and no error messages are being displayed there. This is done to make sure the user is not accidentally adding a predetermined string as their password. If everything is in order, then a simple value change of the input field takes place to put the generated password in the field. Getting the password is done with a for loop and gets each character one by one because there is a bug, where for the '&' character, four characters are padded next to it, in the process of copying the string. There is no clear reason this happens, so code has to be added to ignore the next four characters, to avoid possible misunderstanding and mistakes.

## **4.4 Development Process**

### **4.4.1 Initial Steps**

At the beginning of the project, we started by creating an HTML page for each page of the extension and giving them basic functionalities and connections between them. At this point, there is no information being stored and everything on the web pages is static information just for testing purposes and not giving too much thought

to the extension part of the project which in hindsight was a mistake as we were forced to change quite a few things in my code because of it.

Sign-in functionality by using a specific password was implemented, as well as the list that shows all of the URLs of the static accounts of the user on the 'All' page. The 'Relevant' page was redundant at this instance of the project as there was no webpage where it could be opened so we could not implement it properly. The final page was the 'Create' page where the fields for creating an account were added and the necessary fields for generating a randomly generated password with settings given by the user.

Password generation was made simply by utilizing the `Math.random()` function of javascript and editing the numbers for them to be in the range of the ASCII code of the character array that was selected. Specifically, there are 4 checkboxes indicating the use of Uppercase, Lowercase, Numbers, and Special characters, along with a slider ranging from the numbers 8 through 32, which indicates the desired length of the generated password. According to which of the 4 checkboxes are selected a variable is created with `Math.random()`, it is then multiplied by the number of acceptable characters of each field (e.g. \* 26 for Latin characters) and then padding is added to each of them to bring them to correct ASCII code number which represents the characters (e.g. + 65 for Lowercase letters as the ASCII code of 65 is the letter 'a'). After all of those numbers are generated, a new `Math.random()` function is called to decide which of the characters produced by the previously mentioned method will be added to the final password. This process is repeated in a for loop which is executed as many times as the length slider in the html file indicates.

To allow the user to add accounts to their profile we have created a separate javascript file where 3 parallel arrays will store the information for the URL, the username, and the password for each account. This approach was chosen to have access to the information on any page of the project as at this point every html file was independent and had its javascript file. This method should have worked fine by using the modules of javascript, but for reasons that we cannot explain, we could not

get the modules to work even after checking syntax, documentation, and tutorials a thousand times.

With this in mind, the only solution we could find was to create a single javascript file for all of the pages of the project and put all of the functionality there for easier editing, all of the HTML files were compressed into a single one where the different pages were displayed by hiding the others from the screen. This allowed us to connect the 'All' and 'Create' pages by finally being able to create and add accounts to the list displayed in the former.

#### **4.4.2 Extension Basics**

Having a working function that allows the user to add accounts to their profile was the cue for us to start working on making the project work as an add-on to Firefox. First of all a manifest.json file was created with all the necessary fields such as the manifest version, the name and version, as well as the permissions we figured that the project would need. Description and icons were some fields that were not needed, but we filled them anyway with some basic information. For the icon displayed in the toolbar of the browser, we decided to create a basic shield shape icon to make it obvious to the user that this was the password manager. At first, the idea was to have 2 separate icons for dark and light modes, but after realizing that the different modes on Firefox don't change the colour of the toolbar one icon was sufficient to do the work. Finally, the browser actions for the file were given, such as the name and icon which will be displayed on the extensions page, and the default popup (html file) which will be opened when the user opens the extension.

By creating this, the extension should at least open the html file but we were getting runtime errors that did not allow my functions to work properly and as a result, could not even change pages between the single html file. This was caused because we had the HTML elements call the javascript functions from their file. A simple fix was needed where we removed all the inline calls of functions from the html file and event listeners were added in the javascript file to execute the functions to the correct elements.

Now that the extension opened, and all of the functionalities implemented so far worked properly it was time to work on the 'Relevant' which was probably the most important page of the extension. Reading the current webpage the extension was opened at was easy and we displayed it at the top of the page at first for testing to make sure it was read correctly but decided to leave it there as it is useful information for the user as well. For now, the relevant list was created by checking whether a URL from the array stored in the javascript file is included in the current website's URL. This is a bad approach, but it was made like this mostly to make sure the list items could be created dynamically from the javascript file without any errors. A button to each list item was added as well which allowed the user to copy the password of the selected account to their clipboard. Autofilling was not yet possible in this stage of implementation and the user had to sign in each time the extension was closed.

#### **4.4.3 Real Progress**

We decided to fix the easy problem first and worked on the sign-in page. Added an input field in the page to indicate the username of the user as it was much needed, and now both username and password fields are checked by the program to allow the user to access their profile and make it possible to press the button 'enter' after filling out the password to sign in to improve user experience. Alerts were added as error messages to indicate to the user if any fields were empty and needed filling and for wrong credentials. For the user to stay signed in when the extension is closed a value in the local storage called session is set to 1 whenever the user signs in. That value is set to 0 whenever the newly added sign-out button is pressed. That button was added below the list in the 'Relevant' page as that was the main page of the extension.

Moving on to a bit more complicated problems, the accounts should be sorted alphabetically in both lists ('All' & 'Relevant') but having three arrays and sorting them based on one of them is very tedious and unnecessarily complicated. So in the name of simplicity, a javascript object was created called credentials, which holds

three values. Those values are the same as those held by the three parallel arrays, only now the relevant information of each account is stuck together instead of relying on their position in each array to find the correct information. Now only one array is needed which holds instances of the credentials object for each account the user adds. Now that single array is sorted alphabetically by taking into consideration the URL of each object in it, and the rest of the information will follow.

While being on the topic of messing with the account information, we decided to give out all of the information for each account to the user. To achieve this instead of just creating a list item in javascript and just passing the URL of the account, we would create and pass a span element which will contain all the information of an account. The password could not be visible to the screen unless the user chooses to make it visible, so after adding the URL and username of each account, another span element had to be created to add a button to hide and show the password, along with the actual password. This function was made by creating an array with the length of all the accounts where each position will hold a string of "\*" characters with the length being the same as the actual password. Now the previously mentioned button will switch between the string of the password of each account and the string of the "\*" characters giving the ability to the user to hide and view their passwords. The same implementation was given to the 'Relevant' page with the only exception being that at the end of the span element, the button to copy the password to the clipboard is added since those are the accounts that they will need.

Going back to the 'Relevant' page, it was time to fix the bad implementation of finding the relevant accounts according to the current website. It was suggested to us to use a library that extracts the domain name of the website, but none of those that we found were exactly the thing we were searching for, so we decided to make our own way of extracting the domain which was far simpler to do than we originally expected. After getting the url as normal, it is passed to a function where the beginning and ending of it is removed, to keep only the critical part of it. Specifically, there is a list of strings that contain the most common domain extensions and replace them with an empty string in the URL. Now the way the relevant list is chosen changes, as now we can check whether the cleaned URL is included in the string of

the URL of each object in the info list, improving the range of relevant accounts shown and making it more consistent.

While we were working on all of the previously mentioned methods, we were researching how to implement the autofill function and manipulate the input fields of a webpage from the extension. All roads were pointing towards using the `browser.tabs.executeScript` method of javascript, which allows us to execute a javascript file as if it was executed by the webpage instead of the extension, basically meaning that we can have access to fields and elements of the webpage and give them the values that we want. We were running tests with this while working on everything else to check whether it would work, but everything failed, and had to leave it to the end. After a lot of frustration and disappointment, we decided not to execute a file in the webpage and just execute a single line of code for testing purposes. To my surprise, this worked even though it was impractical and we had to execute nested `executeScripts` to pass the information of the relevant accounts dynamically, as we only had one line of code to work with. However, during the later stages of implementation this was revisited and the error that didn't allow us to run code from a separate file was found, so now we can use this more practical solution. This concludes the implementation of all the basic functionalities which the project will have and so version 1.0 is completed and it is time to move on to make the manager more user-friendly and improve the user interface.

#### **4.4.4 UI Improvements**

The most obvious element that needed improvement in the manager was the lists that display the accounts to the user. A simple 'div' element was created in both 'Relevant' and 'All' pages to contain the list and make a scrollable interface in which the user can find their accounts. Centered everything inside those boxes to keep consistency as well as the buttons of each list item. Passwords now have consistent break points in lines to keep them from overflowing from the container at 14 and 28 characters long.

In the 'Create' page we added a copy button in the generated password which allows the user to copy the generated password to the password field. This created a strange problem where the '&' character when copied added the string of characters 'amp;' after it. The even stranger part of this was that if an account was created successfully with a '&' character in its password which was added with the copy button, the actual password string would not contain the 'amp;' string in it, but the string of '\*' characters in the lists would have extra characters in it to accommodate for the extra bugged string. As difficult as it was to spot this kind of bug, the solution was relatively simple, as it is now hard coded that each character is sent one by one to the password field, and when the character is '&' the next 4 characters are ignored in the generated password. This works as it is only when the copy function is called that the string adds the bugged string and takes into consideration the possibility of the generated password adding by chance the same string of characters without the use of '&'.

To finish off the project, an icon was added for the copy function in the password generation page to clean up the user interface, and error messages were added to the sign-in page in a separate space from the input fields to be easily distinguishable, along with cleaning up the line breaks in the error message of the password generation.



# Chapter 5: Evaluation

## 5.1 Evaluation setup

We run all of our experiments on a single machine with the AMD Ryzen 7745HX CPU that has 3.60 GHz. The main memory of the machine is 32GB. The operating system running on this machine is Windows 11 Home, and the browser used for the experiments is Firefox version 125.0.3.

For all tests where we measure time, we run 100 warmup rounds of generation and then a million rounds where we measure the time needed to execute them to calculate the time of generation more precisely.

## 5.2 Memory Allocation

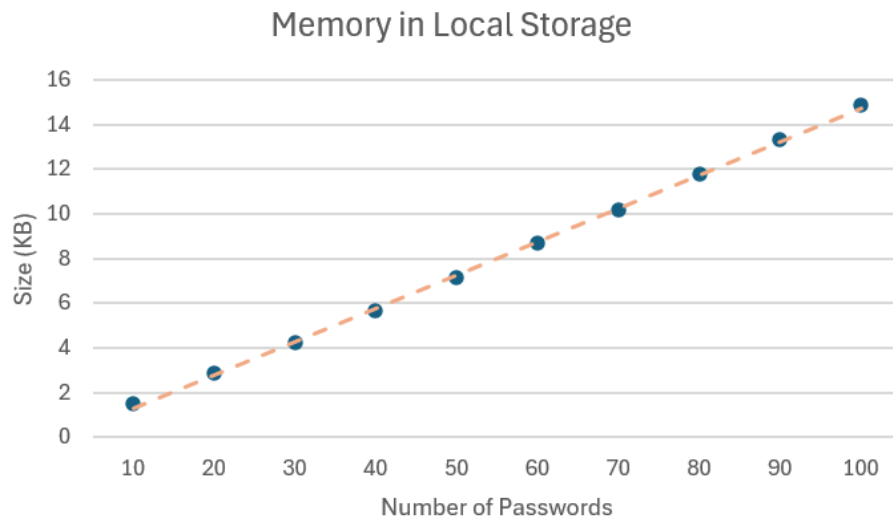


Figure 5.1: The storage size of the local storage as the number of stored passwords increases from 10 to 100.

The password manager to function utilizes the browser’s local storage to store the user’s credentials and account status. A variable to check whether the user is currently signed in or not is used but it occupies a negligible amount of space at 0.02KB so it can be ignored for these tests.

To be able to assess the storage space needed for users correctly, we gathered information on the storage space occupied in intervals of 10 for up to 100 separate credentials stored with the maximum amount of password length allowed. As shown in Figure 6.1, the storage space needed increases linearly as expected, with each of the credentials requiring an average of 0.15 KB of storage space.

It’s noteworthy to mention that the maximum storage space provided by the browser for local storage is 5 MB, so we can safely assume that even when users store credentials that take the maximum space they can, the ceiling of storage space will never be realistically reached by a single user.

### 5.3 Password Generation

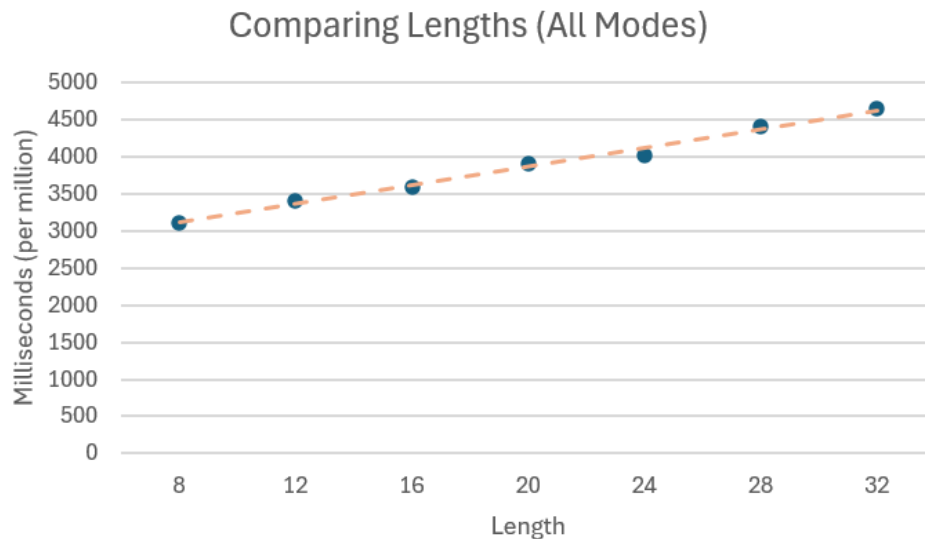


Figure 5.2: The time needed to generate passwords as the length of the passwords increases.

To correctly evaluate the time needed for our password generation tool, a series of tests were conducted where, as mentioned above in the setup, we calculated the time of generation for a million passwords. This test was run ten times for every instance shown in Figures 6.2 and 6.3, in order to more accurately calculate the time needed for a single password through the average of our results.

First, we will compare the generation times of the different lengths the tool can generate passwords, specifically starting from the minimum length and going to the maximum by advancing in intervals of four. As expected the time needed to generate a password increased linearly as the length of the password increased, with an approximate increase of 200 milliseconds per million passwords for every interval.

However, when comparing the modes of generation for this feature the results were not what we had initially expected. By choosing fewer modes for password generation the time needed to generate the passwords was reduced, which was a safe assumption to make even before taking the tests, but when comparing the different modes in single selection results varied significantly. When generating passwords with only Uppercase or Lowercase characters generation time recorded was the lowest, and when generating passwords with only numbers there was a slight increase but nothing too significant. The same cannot be said though for the time needed to generate passwords while using only special characters almost surpassing the time needed to generate passwords that feature all of the generation modes. Figure 6.3 shows the results when generating a password with the maximum length but this trend persists throughout all of the tests taken and can be considered a norm instead of just an anomaly.

In the most extreme scenario, the time required to generate a million passwords was calculated to be an average of 4800 milliseconds, but users will want to generate one password at a time which brings the time of generation down to nanoseconds. Generation time will have no impact on the user's experience, so we can safely evaluate our password generation tool as efficient.

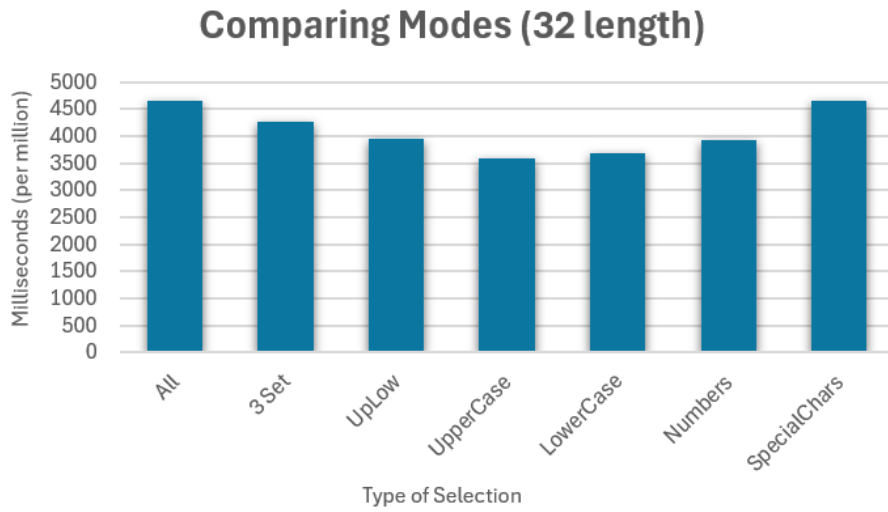


Figure 5.3: The time needed to generate passwords as the mode changes to include different sets of characters with a fixed length of 32 characters.

# Chapter 6: Conclusion

## 6.1 Summary

In this report, we examined several existing password manager and their functionalities to better understand their differences and the key features that are required to create a useful and safe password manager. We focused more on the user experience for each of them rather than the security protocols and distinguished the most useful features.

From our examination, we decided that the features we will use to create our password manager will be an auto-completion tool that allows users to fill out login forms easier and faster, without the risk of errors. Users will also benefit from unlimited storage space for all of their credentials and a password generation tool to create more secure passwords.

These features will be separated into three different pages to be more distinguishable to users, which are the 'Relevant' page which holds a list of all credentials that are deemed relevant to the user's current web page, along with the autocomplete feature, the 'All' page that contains a list of all the user's credentials with the given ability to search for credentials based on the URL, and finally the create page where users can add more credentials to their account with the benefit of a password generation tool for easier creation of secure passwords.

Implementation of the password manager was executed using Javascript and taking advantage of the tabs API for the necessary functions it provides for communication between web page and extension. Credential data are structured as objects consisting of three string elements, those being the URL, username, and password, for easier use of the credentials in functions of the extension. All functions are initialized the moment the extension is loaded and wait for certain actions from the user, by using event listeners.

Evaluation of our system produced relatively expected results in storage space needed for credentials and time needed for password generation. The only

notable result that is worth mentioning is the fact that when generating passwords using only special characters the generation time increased to almost match the time needed when selecting all modes of generation. Storage space for a singular password required an average of 0.15 KB, significantly smaller than the 5 MB maximum storage space, safely assuming that the limit will never be realistically reached by a single user. Generation times were also very efficient since our results showed that for a million passwords, the time to generate was at an average of 4800 milliseconds, bringing the times needed for a single password down to nanoseconds, which is very negligible.

## 6.2 Difficulties Faced

Creating a password manager from the ground up was a relatively simple project to undertake thanks to my prior knowledge of basic HTML and JavaScript. However, challenges started to arise when trying to use APIs that focus on working on webpages through extensions.

Accessing the user's web page was a straightforward task, but constructing a method to correlate the URL with the stored credentials was more challenging. There were a few functions built-in JavaScript, but none of them were quite what we were looking for. To find a solution for this we had to create a custom function that 'cleans' the URL that we extract from the user's web page, leaving only the domain name for comparisons in our functions.

Injecting script to the web page from the extension was also a task that proved to be more challenging than expected. Specifically, to make an effective auto-complete function we need to have access to input forms on the web page to fill them out with the user's information. Initially, we tried to transmit data using the JSON format, however, we encountered a lot of bugs and errors so we decided to change to a simpler approach. Using the tabs API we can use a built-in function that allows us to execute a separate script on the web page we desire. With this function, we can call custom functions from a file that contains the user's credentials and fill out login forms in any given web page.

## **6.3 Plans for future work**

With the extension now fully operational, the next logical step is to connect the password manager with a secure database, with advanced encryption standards. Drawing inspiration from our analysis of previously existing systems, we could encrypt users' credentials using RSA, an asymmetric type of encryption that is extremely difficult to decrypt without the key, and even hash the credentials before storing them in the database for even more security against malicious actors.

The implementation of our autocomplete function could have some limitations, since it tries to find common names for the input fields in web pages. In future plans the addition of support for more websites is one of the top priorities. Increasing security could also be a result of adding a two factor authentication method, where users use a code sent to them via an external method during account login.

Additionally, some quality of life changes could be added to the password manager, by adding secure notes for users, organized groups of credentials, automated monthly report of account security, and even credential sharing across trusted parties. All additions will be made with security in mind and how well they improve user experience.

To better improve the user experience, we could also have real end users evaluate our work, to get real feedback for our system so we can better understand the users' needs and improve our system accordingly. This can be done from focus groups, or by sending a questionnaire to all of our end users where we can get the feedback from.

## **6.4 Things I have learned**

Working on this project has allowed me to use online source code management platforms like GitHub, which made me learn the value these platforms provide in terms of sharing source code, and getting feedback in real-time.

Additionally the ability to roll back to previous version was a very useful tool to use in the case of a big error occurring.

Through the development process I was able to refresh my previously acquired knowledge on web development, but most importantly I learned about everything that goes in the development of a web extension. Specifically, managing permissions, injecting scripts from extension to web page, and accessing information from a web page were the key functionalities I had to learn in order to develop a web extension.

Additionally, I learned how to conduct a proper evaluation of a system, by being able to properly extract useful information that allowed me to design a password manager which is user friendly, but also as equally important, secure through its design without relying just for encryption methods. I was also able to correctly evaluate my own system's performance and properly showcasing the results in an orderly fashion where we can extract and explain them.



# References

- [1] The Bitwarden password manager. <https://bitwarden.com>.  
Accessed: 7/10/2023.
- [2] The 1Password password manager. <https://1password.com/>.  
Accessed 14/10/2023.
- [3] The LastPass password manager. <https://www.lastpass.com/>.  
Accessed 21/10/2023.
- [4] Information on security breaches of LastPass <https://www.makeuseof.com/>.  
Accessed 21/10/2023.
- [5] Usability of the tabs API in Firefox <https://developer.mozilla.org/>.  
Accessed 20/02/2024.