

## **Password Manager App Project Report**

### **1. Introduction**

In this digital age, maintain strong passwords across various platforms is crucial, as now every website and app requires the signup and login to help you keep track on certain things, like for example Instagram, in order to access it you need to sign up and remember those credentials so that you can use it to login and so that you can view your friend's post or status and keep track of your profile. So as part of day-to-day life we are on multiple websites and its most likely that we sign up on some of them, and tracking all these passwords and usernames can be daunting. For this project, I am implementing a Password Manager Application which is designed to provide users with a secure way to store and retrieve their passwords. This app is convenient and helpful to everyone because it contains features like secure password storage and management for various websites. The Secure Password Manager Application is implemented using Python and Tkinter library in Python for building the GUI (graphical user interface).

It is important for any application to have a password security because in this digital era we can see that there is a high amount of security breach which causes us to choose a strong password to protect the personal and sensitive information from unauthorized access. The use of weak and reused passwords can cause account compromise, because a lot of us tend to use the same password for most of the account and use something like birth dates, graduation dates and other which are easy to be guessed by the attacker. It is important that the users choose password that will be hard to be attacked by the attackers.

The main purpose of the Secure Password Manager Application is to give the users a platform where they can securely store and manage the passwords for multiple websites. This can be done. This application uses strong encryption and hashing functions to help maintain the integrity and confidentiality of user's sensitive information. Overall, the application offers a user-friendly interface for generating and retrieving strong, unique passwords, which will help in enhancing the security. The targeted audience for the Password Manager Application can be anyone, like general users, who want to have an application to store all their passwords and be able to manage it instead of forgetting the passwords and resetting it repeatedly.

### **1.1. Problem Statement**

Managing passwords is tiring and daunting, keeping track of the passwords of multiple accounts can be hard and many individuals struggle due to it, which causes them to have same password for multiple websites/accounts, because its easier to remember the same password that is used for multiple websites than remembering each unique password for each account. For users remembering or creating complex passwords could be hard which will mostly likely result in them using weak passwords, which could pose security concerns. Due to this many users tend to save or store their passwords in an unsecure way, like storing it word file, notes or in any other plaintext format, which is not reliable and secure because there is a chance that it could fall in the wrong hands because there is no secure protection that will prevent the attacker from getting access to the files, which could compromise user information.

This problem is relevant because it is important for a user to save and manage their sensitive information in a proper way, where the security, integrity and confidentiality are maintained. It is important to address and focus on the above problem because reusing and using weak password poses high security risks because this means that the user personal information could be breached or be attacked by any hackers because once the hacker applies attacks and finds out that the user is using same password for all the websites and accounts, then all the personal information would be compromised, which can result in identity theft, losing financial information and other sensitive information. As data breaches and security compromises on online platform are becoming more common the danger of the sensitive information being leaked needed a better solution for protecting user's information.

## **1.2. Functional Requirements**

- The Secure Password Manager application can support the creation of multiple user accounts, and each account has their own database of stored passwords after logging in.
- Each individual can create their own account and can change their password using the security question set up that is implemented during signup process.
- The user can store an existing password for a website or application along with username, website name and password.
- The user can generate a strong random password for the certain websites or application they want.
- This app secure protects the passwords, by adding an extra layer of security by encrypting the passwords before storing in the database using AES – 256 and

Galois/Counter Mode block cipher and decrypts the password before the user derives in the same way.

- This application makes sure of the authenticity by implementing the master password feature where the user sets up one during the signup process, which is hashed and stored in the database and when logging in the user is asked to enter the master password and then that hash value is matched with the initial hash value and if matched then the user can enter the app.
- The master password is also used to derive the key for the AES encryption. The master password is hashed, and the key is derived from this hash value and that key is used for AES encryption and decryption.
- In the password manager screen the user can modify the username, website name and password, and have a delete option, where they can delete the website name, password and username and can copy the password to their clipboard.

## **2. Scope of work**

The features that have been implemented in this Secure Password Manager Application is the user authentication, where the user can sign up and login with the username, password, and master password. The master password is used as an additional layer of authentication and to ensure the integrity of the user. Once the user logs in there a password manager window, where the user can add the information like, website name, username, and password for the websites they want to save password for, and they can click add button which saves it

Aparna Mandapaka

Cryptography

April 26,2024

to the database. The passwords are encrypted using AES-256 and GCM block cipher mode before stored in the database. I have also implemented the password generation feature, where the user can generate passwords for websites if they can't come up with their own password. The password generation feature makes sure that it generates strong and random password by using a special character, at least one capital letter and at least one digit in the password it generated randomly. There also a feature where the user can view the saved password by clicking on the 'Retrieve Password' where it opens a window that shows the saved password by the user, and there is an option where the user can edit the website name, username, and password and there is a delete option. The passwords are retrieved from the database and decrypted using the AES-GCM algorithm. The overall scope of this project is to help the users by increasing online security and protect the user data for anyone that uses this app.

Some of the features that are included in the project are the GUI which is used using the Tkinter library in Python. I have used SQLite as database management for storing the user's information and password. Some of the cryptographic techniques that I have implemented in this application is AES-GCM for encrypting and decrypting passwords, used SHA-256 to hash the user's password when they signup and the master password and I have also used the PBKDF2 to derive the key from the hashed master password. There is the implementation of the user authentication where the app makes sure that the user gets a secure access to the app and also there is the implementation of the random password generator where it helps the user by giving a strong and random generated password. Some of the elements that I wasn't able to implement in this app is having an email integration which can be used to reset the

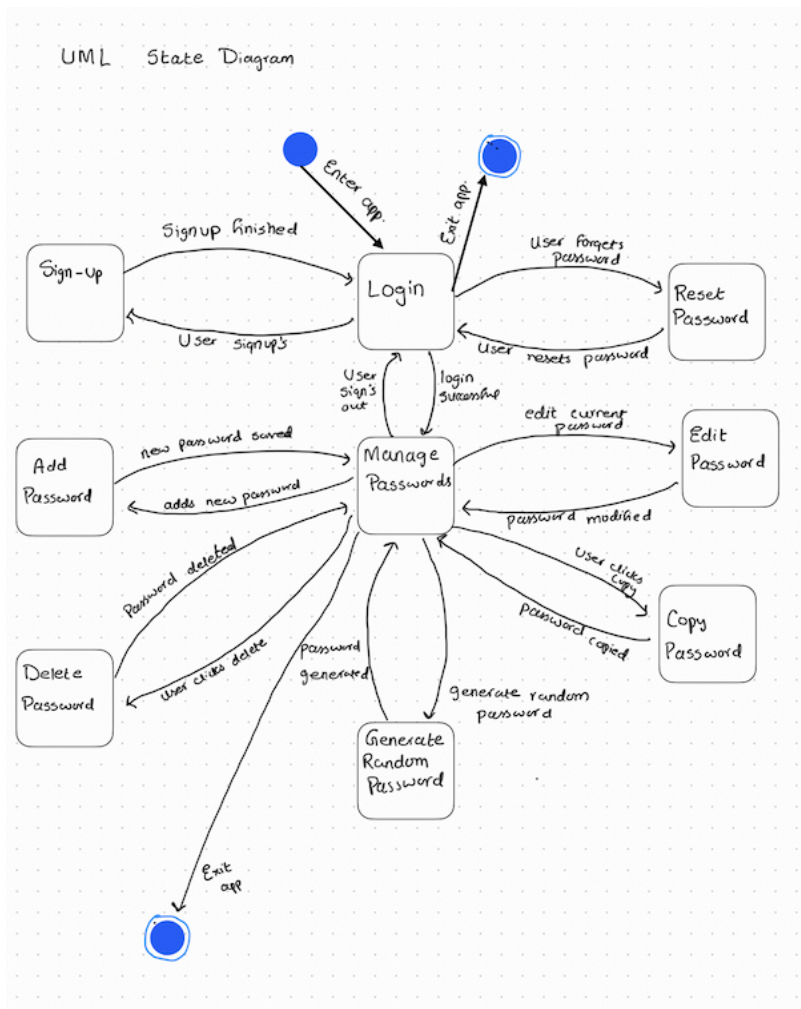
Aparna Mandapaka

Cryptography

April 26,2024

user's password if they forgot, my initial plan was to use flask and integrate the emailing system in this app so that if a user forgets their password or master password they can reset it via the reset link that will be sent to the email, but unfortunately due to timing constraints I wasn't able to implement that feature. Another feature that is excluded is the implementation of 2FA (two-factor authentication) because it was hard to implement this feature using the local database. Some of the limitation for this application is that currently, this app only runs on the computer and not on mobile devices, so in future I want to work on this app to make it run on the mobile platform.

### **3. System Architecture**



## 4. Design

### 4.1. Database Design

The database design for the Secure Password Manager Application is important for security and the integrity of the user's information. For this project I have used SQLite database to store the user's signup, login information and the encrypted passwords. For this project I am using two database files, one is users.db and another database file is passwords.db. In the users.db database file there is a single table called 'users', which stores the users information, the table contains the following fields, username- which is the unique

identifier for each user, hashed password – it's the hashed representation of the user's password, which is combined with salt to enhance the security, hashed master password – it's the hash of the user's master password which is used for deriving the encryption/decryption keys, salt- randomly generated salt value used to combine the user's password for hashing, hashed security question answers – which is the hash of the security question answers that are provided by the user during signup process. The password.db database file which contains the table named 'passwords' , which is used for storing the encrypted passwords that go along with the specific user account. The schema for this database file has the fields, website – name/URL of website associated with stored password, username- username that is associated with the stored password, password – shows the encrypted user's password using the AES-GCM encryption, user username – uses the foreign key for referencing the username of the user who the password belongs to. The database operations in this application include the user signup, login, password retrieval, adding, modifying, and deleting passwords, usernames and websites. The database also handles errors when the user enters incorrect login information, if they provide existing username during signup or when they provide incorrect answers to the security questions during password reset. In this app the database is implemented in a way that it prioritizes security, integrity and the user's privacy by providing a reliable platform.

#### **4.2. GUI Design using Tkinter**

The GUI for this application is built using the Tkinter library for Python, this feature provides the users a unique way to use the application. The key components that I have



implemented using the GUI is login window, this is the primary interface that user sees when they run the application, this window contains the text field for username, password and some buttons, like signup and forgot password. When users click on the signup button, it opens a new window where the user can create a new account it contains the text fields for username, password, master password and it has the answer field for two security questions which will be used when the user wants to reset the password. After filling in all the fields and when the user clicks the signup button, the user gets displayed a dialog box which says that the user has logged in successfully, and using these credentials the user can login. Once the user is logged in they will see the password manager screen, where there is a text field for website, username and password and it contains four buttons add password, generate random password, retrieve passwords and logout. When the user fills all the text fields and adds password it gets added to the database, passwords.db, but it does not store the plain text in the database, the passwords are securely stored using the encryption techniques AES-GCM to protect the user's information and prevent unauthorized access. When the user clicks on Generate Random Password button it generates secure random password which contains the variety of capital letters, digits, and special characters. When the user clicks on Retrieve passwords button, if there are not password stored it displays no passwords stored. But there are stored password then it displays the website name, username and password and there is an option to modify the password, delete and copy it. When the password is modified it gets modified in the database and the encrypted version will be stored, when the user deletes, it gets deleted from the database. When the user clicks the logout button, it logs the user out and take the user back to the login screen When the user clicks on the Forgot Password

button, it opens a Reset Password screen, where there are three text fields, username along with the security questions, favorite food and elementary school and Reset Password button, when the user fills in all the fields and clicks reset button, in the background the code compares if the hash value of the answers of the security questions during the sign up match the hash value that is calculated for the recently entered security answers if it matches then it lets the user, set the new password for the account following the password requirements, if not it says wrong answers.

#### **4.3. Encryption and Decryption Design**

For this project in order to achieve the security of the user's information, I have implemented the encryption and decryption of saved passwords using the AES-256 along the block cipher GCM algorithm, using this combination provides the confidentiality and integrity for the stored passwords. Here is the basic overview of encryption process, as the user adds the password in the password manager screen, in the backend a random nonce of 12 bytes is generated, then the AES-GCM is activated with the generated nonce, then a symmetric key is derived from the hashed user's master password and the GCM block mode. Then the plaintext, which is password is then encoded into UTF-8, and encrypted using the initialed cipher and then return with the authentication tag.

The decryption process takes places when the user wants to retrieve the stored password, so during the decryption the nonce and authentication tag are extracted form the stored ciphertext. Then the AES-GCM cipher is initialized with the extracted nonce, since AES is symmetric algorithm same key is used. Then the ciphertext and authentication tag are fed to

the cipher for decryption and authentication, if this process happens successfully then it returns the plain text password if not it throws errors.

#### **4.4. Key Derivation Process using PBKDF2**

For deriving the keys from the hashed master password, I have used the PBKDF2 algorithm, which derives the encryption and decryptions keys to use for the AES-GCM algorithm. The PBKDF2 algorithm takes the hashed user's master password, along with the randomly generated salt and does the key derivation with the high number of iterations to derive strong encryption key.

#### **4.5. Password Hashing Process**

In any app, the security is the most important aspect. For my application to protect the user's information from unauthorized users I have implemented the hashing process. The password hashing is the cryptographic technique which converts the user's password into a unique string of characters, and this is a one way function, it makes sure that if the hashed values are compromised, they cannot be reverse-engineered to obtain the plaintext. For this application I have used hashing algorithm for salt generation, before the passwords are hashed a random salt is generated for each user, this helps in making sure that there are no attacks like the rainbow table attacks by making sure that even if two users have the same password, their hashed password will be different due to the unique and random salt value. I have used SHA-256 algorithm to enhance the security and protect the app against brute force attacks. I have used hashing algorithm for password hashing, so that when a user creates or

updates their password, it will be undergone through hashing process, where the user's plaintext password will be combined with the salt and the combine string is passed to the SHA-256 algorithm to generate the hashed password. The hashing algorithm is also used for the storage, where only the hashed passwords and its corresponding salts are stored in the database, in this app we never store the plaintext in the database for security purposes. The hashing algorithm is used for verification purposes, when the user logs in, the entered password is hashed using the same salt and hashing algorithm, then the hashed password is compared to the stored hashed password in the database, if it matches then it logs the user in, if not it prompts a error message.

## **5. Implementation Details**

### **5.1. Signup Process**

The signup process/window allows the users to make a new account by providing the text fields, username, password, master password, and security questions like favorite food and elementary school name. Here the password that user sets up should follow the password requirements, like the password should at least be 8 characters long, should contains at least one uppercase letter, one digit, one special character and shouldn't be same as username. When the user enters all the fields and clicks on signup button, the code in the background checks the username already exists, if it does, then it displays the message saying that the username exists and it checks the password to see if it meets the password requirements, if all conditions are met then it successfully creates an account for the new user. The master password does not have any requirement because it can be anything like it can be a

paraphrase, a word or anything that the user wants it to be and can remember. When the user signs up the password and the master password are saved in the database by securely hashed by the SHA-256 algorithm and the randomly generated salt values so that even if the attacker gets access to the database file, I cannot get the password because they are being stored as hash values.

## **5.2. Login Process**

The login process acts as an important step because this provides the access control and security for the user accounts. When the user is at the login screen, they have to enter the username and password, in the backend the code will be performing the login validation where the app retrieves the stored hashed password and salt from the database based on the given username, then the entered password and stored salt is combined and then hashed and this is used to compare it with the stored hashed password, if the credentials match, there is another verification step which is the master password verification, the entered master password is hashed with the salt retrieved from database and then used as a comparison with the stored hashed master password, if it matches then the user can view the password manager screen.

## **5.3. Password Storage**

In the Secure Password Manager application the stored password are secured using cryptographic techniques. In this app, the PBKDF2 algorithm is used to derive keys from the hashed master password, this algorithm applies a pseudorandom function, like SHA-256 and iteratively generates a secure cryptographic key, thus it makes brute-force attacks more

difficult and time consuming. For encrypting and decrypting the user data, I have used AES algorithm in the GCM, as it provides authenticated encryption by making sure that both confidentiality and integrity of the data is achieved. Since AES is symmetric algorithm same key is used for encryption and decryption. In the user's interface, when displaying the stored passwords, the encrypted password data is decrypted using the derived key before showing to the user, this make sures that the passwords are only accessible in their decrypted form during the session and never stored in plaintext. To view the saved passwords, we can click on the retrieve passwords button, which has features like edit, which helps the user to modify the website name, username and password, the modified password when clicked saved button, replaces the old existing password and uses the same encryption technique before being stored in the database, the delete option deletes the entry and the copy button copies the password to the clip board.

#### **5.4. Random Password Generator**

The random password generator helps the user by generating random password for specified length, for this project 12 character is default. Using the python library secrets the generator randomly selects the characters for the specified length and then the select characters are concatenated to form the final password string. The generated passwords include the uppercase letters, digits and special characters. The function for the random password generator combines all the characters to form a pool of characters from which the random password will be composed to enhance the complexity and security of the password, and when this is stored in the database it is also encrypted using AES-GCM.

### **5.5. Reset Password**

If the user forgets their password for the Secure Password Manager Application they can reset the password. When user clicks on forgot password they are taken to a window where they enter the user name and answer the security questions, that they have answered during the signup process, the answers during the signup process are hashed using random salt value, so when the enter the security answer before resetting the password, the entered answer is hashed with the combination of salt value if the initial hash value matches the current hash value calculated then it lets the user to the new password page where they can enter the new password for the associated username and the new password replaces the old passwords in the users.db by following the same hashing(SHA-256) and salting techniques.

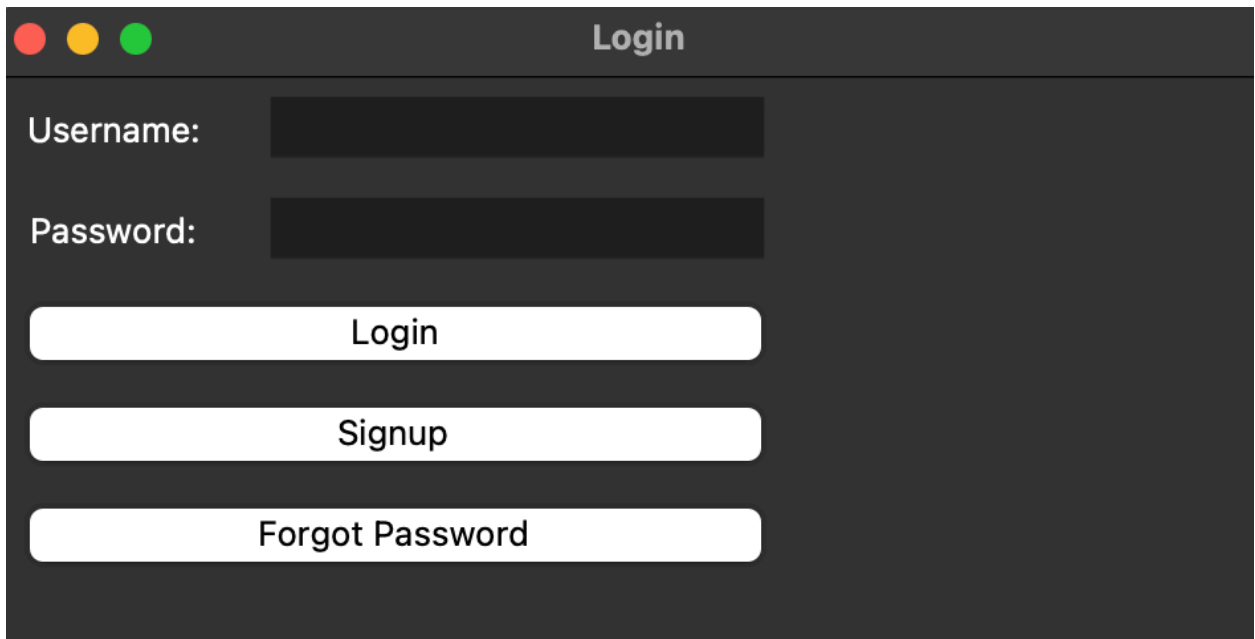
## **6. Mockups (screenshots)**

**Screen when code is ran**

Aparna Mandapaka

Cryptography

April 26,2024



A dark-themed login window with a title bar containing three colored circles (red, yellow, green) and the word "Login". The window has a dark gray background. It contains three input fields: "Username:" followed by a dark gray rectangular box, "Password:" followed by a dark gray rectangular box, and a "Forgot Password" link. Below the input fields are three white buttons with rounded corners: "Login", "Signup", and "Forgot Password".

Username:

Password:

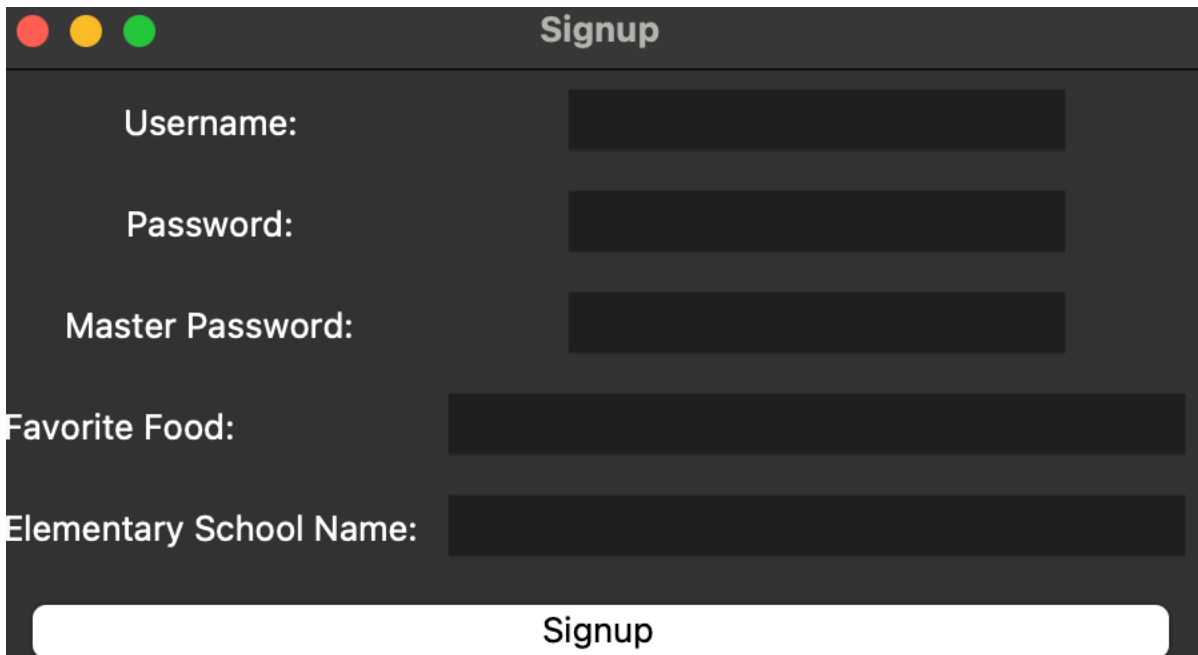
[Forgot Password](#)

Login

Signup

Forgot Password

Sign up screen



A dark-themed signup window with a title bar containing three colored circles (red, yellow, green) and the word "Signup". The window has a dark gray background. It contains five input fields: "Username:" followed by a dark gray rectangular box, "Password:" followed by a dark gray rectangular box, "Master Password:" followed by a dark gray rectangular box, "Favorite Food:" followed by a dark gray rectangular box, and "Elementary School Name:" followed by a dark gray rectangular box. Below the input fields is a single white button with rounded corners labeled "Signup".

Username:

Password:

Master Password:

Favorite Food:

Elementary School Name:

Signup



Aparna Mandapaka

Cryptography

April 26,2024

**Signup**

Username: Kevin


Password: \*\*\*\*\*

Master Password: \*\*\*\*\*

Favorite Food: pizza

Elementary School Name: kilmer

**Signup**



**You have successfully signed up!**

**OK**

**If username already exists**

Aparna Mandapaka

Cryptography

April 26,2024

Signup

Username:

Aparna

Password:

\*\*\*\*\*

Master Password:

\*\*\*\*\*


Favorite Food:

pizza

Elementary School Name:

kilmer

Signup



**Username already exists. Please  
choose a different username.**

OK

Login screen with master password

Aparna Mandapaka

Cryptography

April 26,2024

The image shows a login application interface. The main window, titled "Login", has a dark theme. It contains two input fields: "Username:" with the text "Aparna" and "Password:" with masked characters "\*\*\*\*\*". Below these fields are three buttons: "Login", "Signup", and "Forgot Password". A modal dialog box, also titled "Login", is open in the foreground. It prompts the user to "Enter your master password:" and features a single text input field with a blue border. At the bottom of the modal are two buttons: a blue "OK" button and a white "Cancel" button.

Username: Aparna

Password: \*\*\*\*\*

Login

Signup

Forgot Password

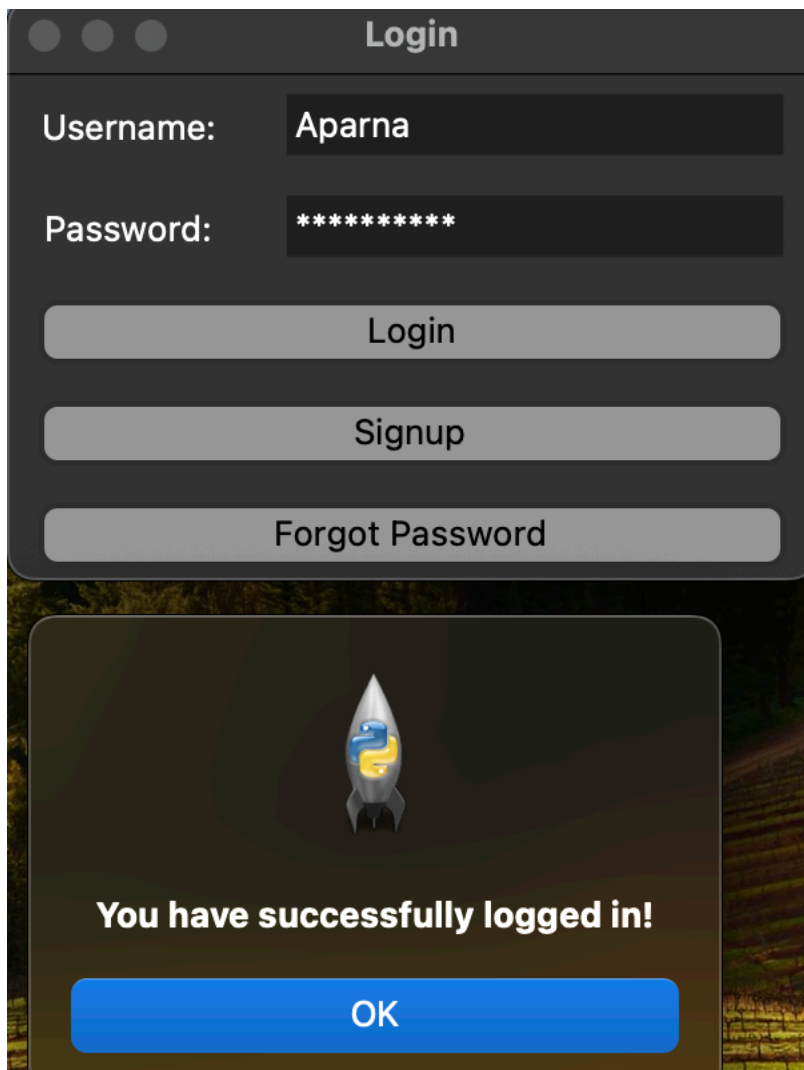
Enter your master password:

OK Cancel

Aparna Mandapaka

Cryptography

April 26,2024

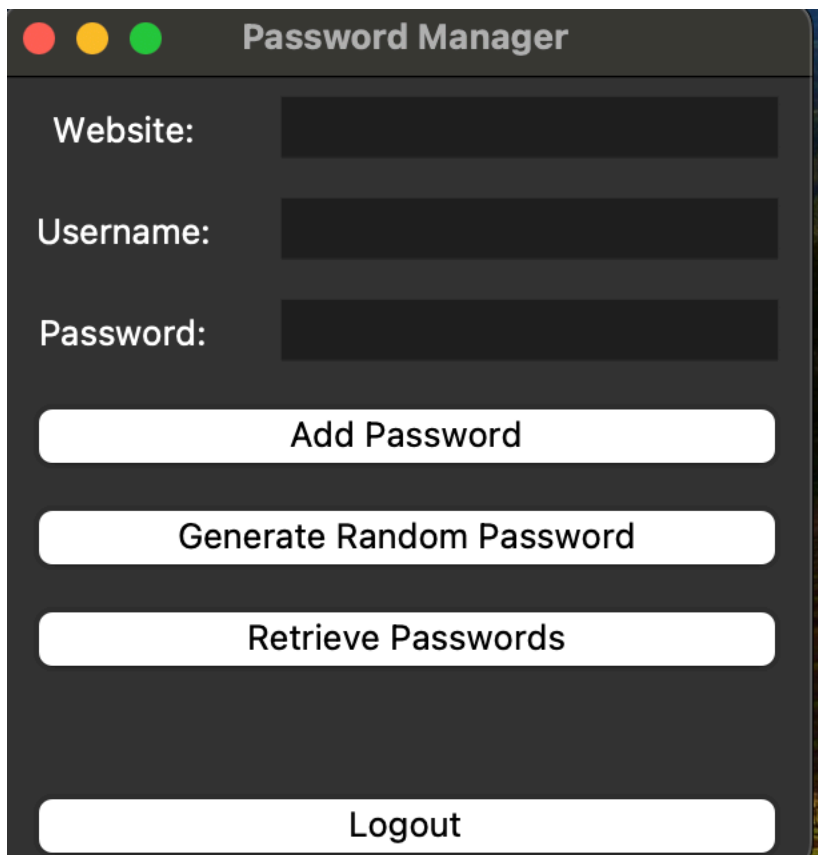


**Password Manager screen**

Aparna Mandapaka

Cryptography

April 26,2024



The image shows a screenshot of a web application titled "Password Manager". The interface has a dark gray background. At the top, there is a header bar with three colored circles (red, yellow, green) on the left and the title "Password Manager" in white. Below the header, there are three input fields labeled "Website:", "Username:", and "Password:" in white text. Each label is followed by a dark gray rectangular input box. Below these input fields, there are four white buttons with black text, stacked vertically. The buttons are labeled "Add Password", "Generate Random Password", "Retrieve Passwords", and "Logout".

Website:

Username:

Password:

Add Password

Generate Random Password

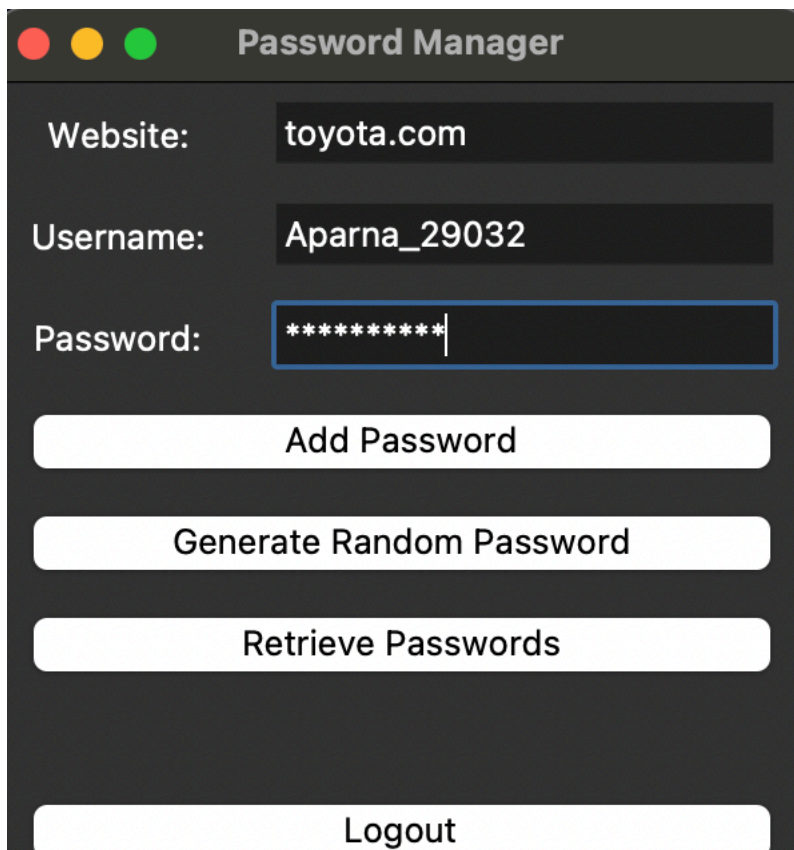
Retrieve Passwords

Logout

Aparna Mandapaka

Cryptography

April 26,2024



The image shows a screenshot of a web application titled "Password Manager". The interface has a dark theme. At the top, there are three colored circles (red, yellow, green) and the title "Password Manager". Below the title, there are three input fields: "Website:" with the value "toyota.com", "Username:" with the value "Aparna\_29032", and "Password:" with the value "\*\*\*\*\*". The "Password:" field is highlighted with a blue border. Below the input fields, there are four buttons: "Add Password", "Generate Random Password", "Retrieve Passwords", and "Logout".

Website: toyota.com

Username: Aparna\_29032

Password: \*\*\*\*\*

Add Password

Generate Random Password

Retrieve Passwords

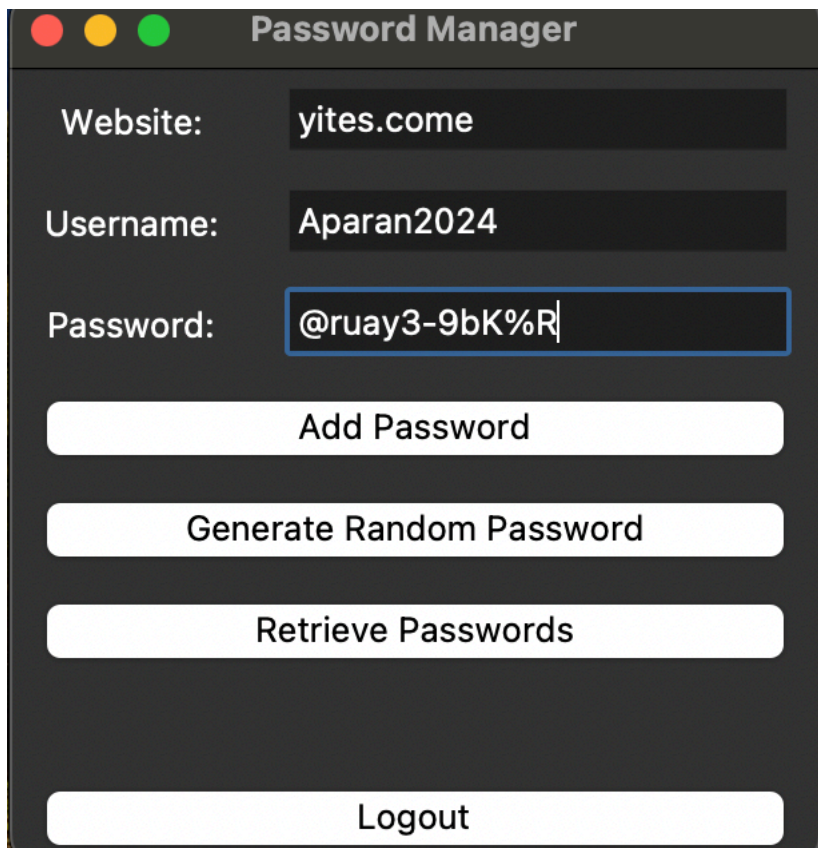
Logout

**Random generated password**

Aparna Mandapaka

Cryptography

April 26,2024

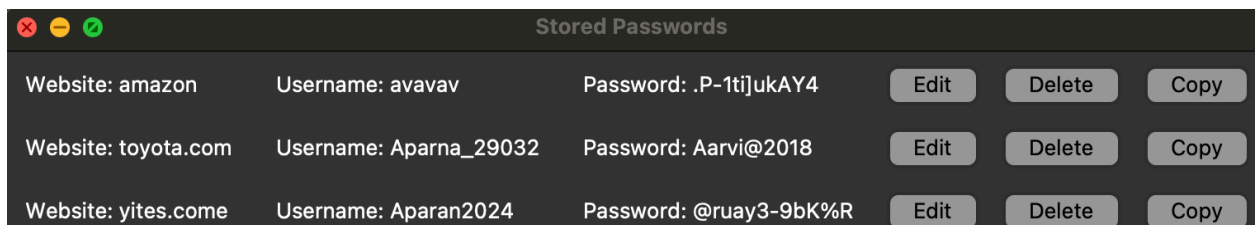


The screenshot shows a 'Password Manager' application window with a dark theme. It features three input fields for 'Website:', 'Username:', and 'Password:'. The 'Website:' field contains 'yites.come', the 'Username:' field contains 'Aparan2024', and the 'Password:' field contains '@ruay3-9bK%R'. Below these fields are four buttons: 'Add Password', 'Generate Random Password', 'Retrieve Passwords', and 'Logout'.

| Field     | Value        |
|-----------|--------------|
| Website:  | yites.come   |
| Username: | Aparan2024   |
| Password: | @ruay3-9bK%R |

Buttons: Add Password, Generate Random Password, Retrieve Passwords, Logout

### Retrieve passwords window



The screenshot shows a 'Stored Passwords' window with a table of saved credentials. Each row includes the website, username, password, and three action buttons: 'Edit', 'Delete', and 'Copy'.

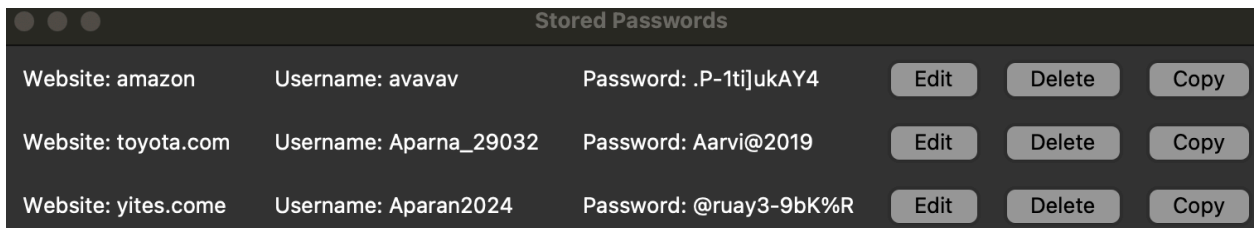
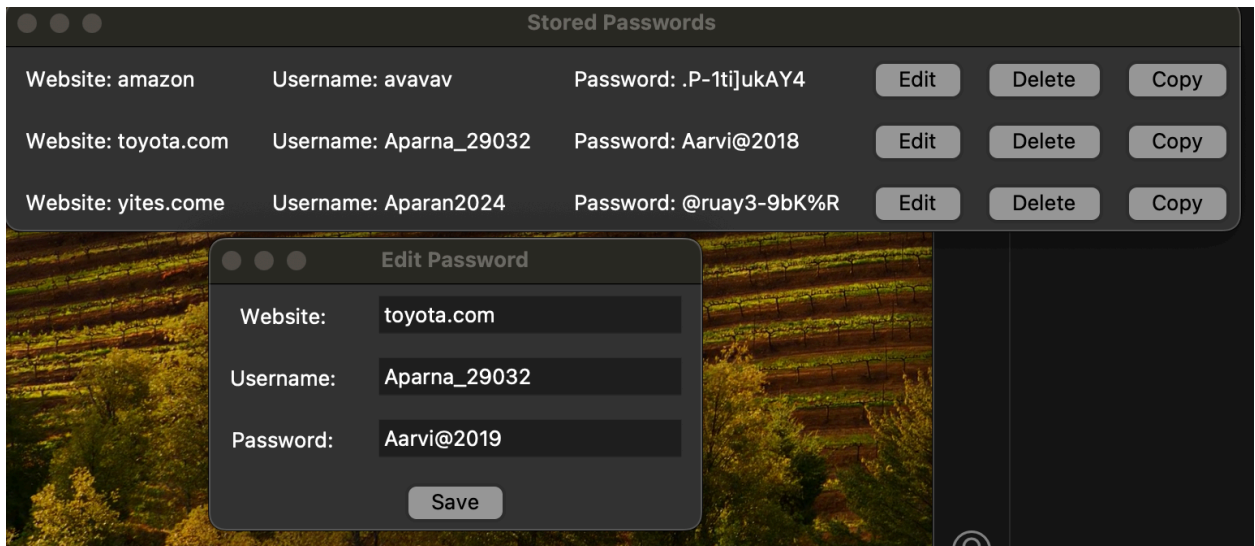
| Website    | Username     | Password     | Edit | Delete | Copy |
|------------|--------------|--------------|------|--------|------|
| amazon     | avavav       | .P-1ti]ukAY4 | Edit | Delete | Copy |
| toyota.com | Aparna_29032 | Aarvi@2018   | Edit | Delete | Copy |
| yites.come | Aparan2024   | @ruay3-9bK%R | Edit | Delete | Copy |

### Password edit

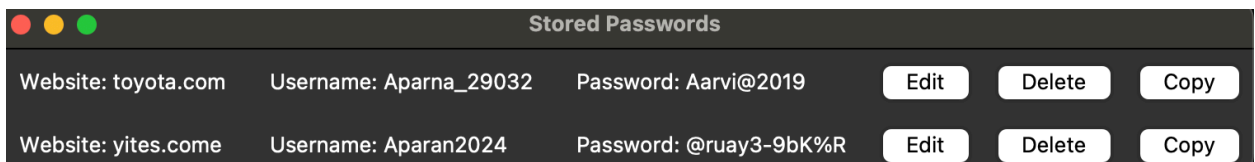
Aparna Mandapaka

Cryptography

April 26,2024



### Password delete



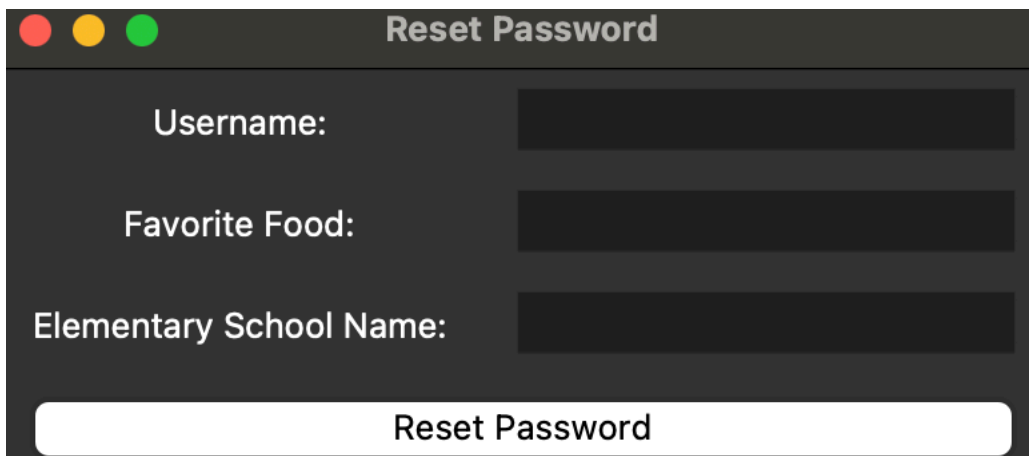
Clicked on forgot password button



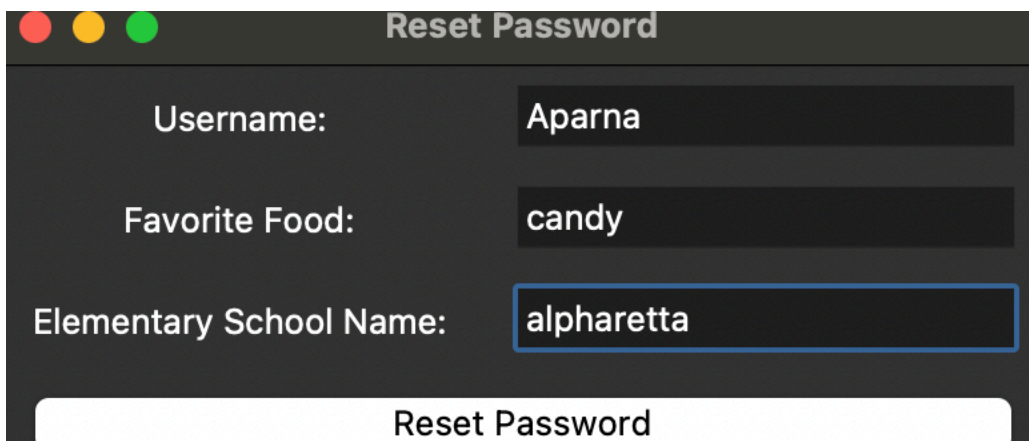
Aparna Mandapaka

Cryptography

April 26,2024

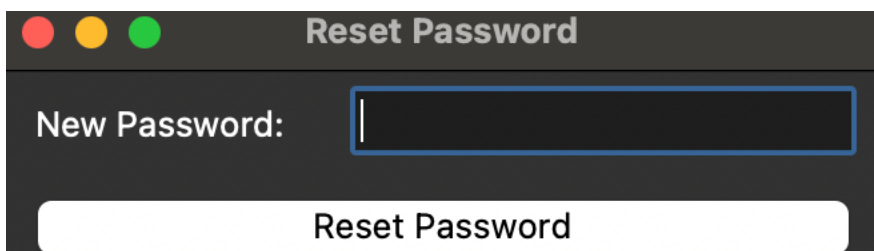


A screenshot of a 'Reset Password' window. The window has a dark gray title bar with three colored window control buttons (red, yellow, green) on the left. The title 'Reset Password' is centered in the title bar. The main area of the window is dark gray and contains three labels with corresponding input fields: 'Username:' with an empty text box, 'Favorite Food:' with an empty text box, and 'Elementary School Name:' with an empty text box. At the bottom of the window is a white button with the text 'Reset Password'.



A screenshot of the 'Reset Password' window with the input fields filled. The 'Username' field contains the text 'Aparna', the 'Favorite Food' field contains 'candy', and the 'Elementary School Name' field contains 'alpharetta'. The 'Elementary School Name' field is highlighted with a blue border. The 'Reset Password' button remains at the bottom.

Takes to new password screen

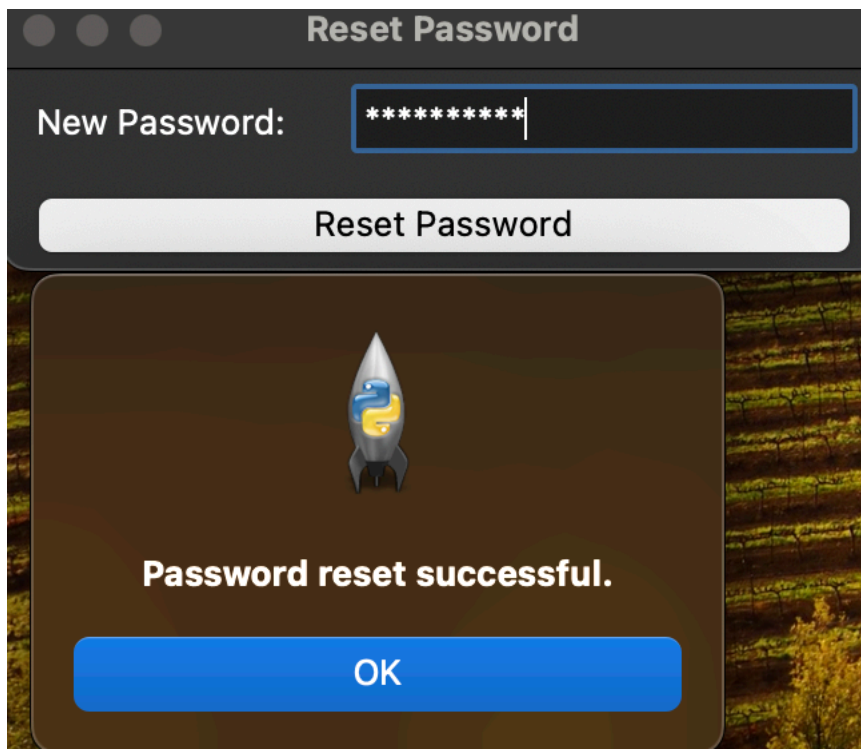


A screenshot of a 'New Password' window. The window has a dark gray title bar with three colored window control buttons (red, yellow, green) on the left. The title 'Reset Password' is centered in the title bar. The main area of the window is dark gray and contains a label 'New Password:' followed by an empty text box. At the bottom of the window is a white button with the text 'Reset Password'.

Aparna Mandapaka

Cryptography

April 26,2024

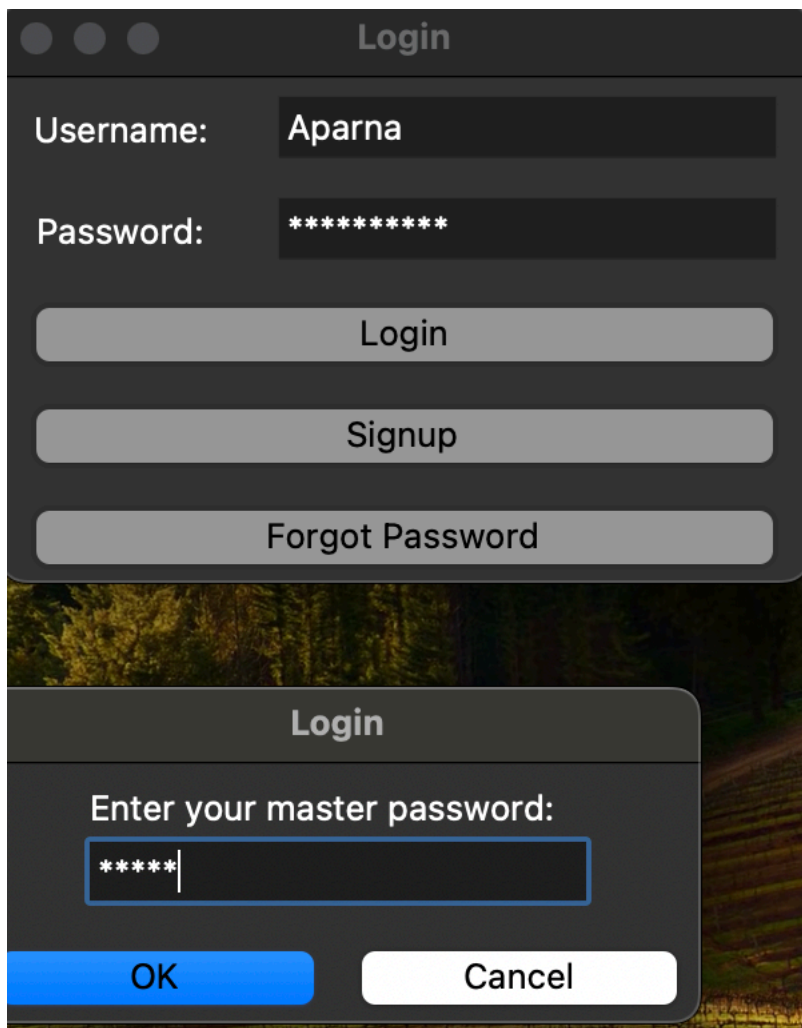


**Login with new credentials**

Aparna Mandapaka

Cryptography

April 26,2024



## 7. Security Measures

The security measures that have been implemented in this app to protect the sensitive data and also to make sure there is secure authentications are storing the passwords that are used for signup and login using the SHA-256 combined with the random salt before storing in the database to prevent the unauthorized users from accessing the plaintext password. Implementing the master password which is used as an extra layer of security and also to derive encryption keys for encrypting and decrypting the stored passwords. The master

password is hashed and stored in the database. I have also used the key derivation function to derive the keys from the master password and salt value, to strengthen the encryption by making it computationally intensive for the attackers to perform brute force. Overall, in this app I have taken enough security measures to increase the confidentiality and integrity of the sensitive data that is stored in the database and also the authentication and authorization aspect by protecting the access to the data.

## **8. Future Enhancements**

Some potential features I want to add in the future version is to have a mobile app using Flutter and dart technology and integrating firebase database, which has already established functions like authentication using email, biometric and other types of authentication techniques to protect the users information. I also want to implement a feature that deals with the browser, like creating browser extension for this app where it enables the autofill feature, so that when the user is on certain website or app the password manager can auto fill the information, but before that it has to check if the correct user is logged in, like enabling fingerprint authentication before performing autofill feature. I want to implement a feature of adding the password expiry notifications and for this I want to use firebase database because it is more efficient and easier to use firebase to send push notifications because firebase has Firebase Cloud Messaging, which is a cross-platform messaging which helps the developers to send notifications and messages to the users that use their app, it could be either IOS, Android and/or web. It would be efficient to use the FCM because it is more reliable in delivering the notifications. Using the firebase database, which is the Firebase

Aparna Mandapaka

Cryptography

April 26,2024

Firestore/Realtime database, we can keep track of the password expiry and then implement a service that constantly checks for the expiring passwords. Then we can use the FCM to send out the notifications to the user's password manager app when the passwords are about to expire.

Certain improvements that can be implemented in the Password Manager Application is enhancing the password generator feature, by giving the user additional options to customize the password, like letting the user decide how long they want the password to be, able give a feature to the user where they can decide how many characters or numbers they want to include in their randomly generated password.

## **9. Conclusion**

Overall, the Secure Password Manager Application makes sure that the user gets the secure storage and management for their passwords using the cryptographic algorithms for authentication, data encryption and decryption and key derivation.