Criterion B: Design

Table of Contents

Software Design

Figure 1 shows a general overview of the program as a whole. Each of the predefined processes in the Figure 1 are then shown further in separate flowcharts.

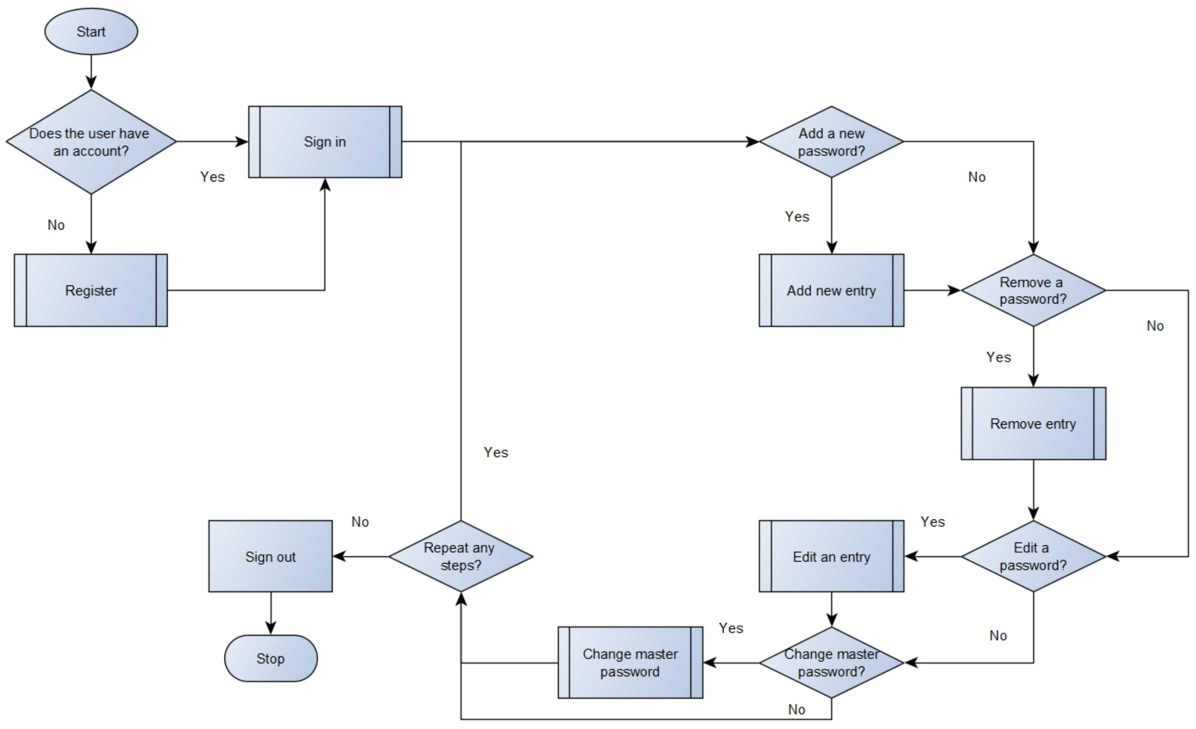


Figure 1: General overview flowchart

When the client starts using the application, he either signs in or registers, depending on whether he has already created an account or not. If he needs to register, he must then sign in once he has created an account.

Once the client is authenticated, he can view his passwords for all applications, and can add new passwords, remove passwords, or edit passwords. Once he is finished, he signs out.

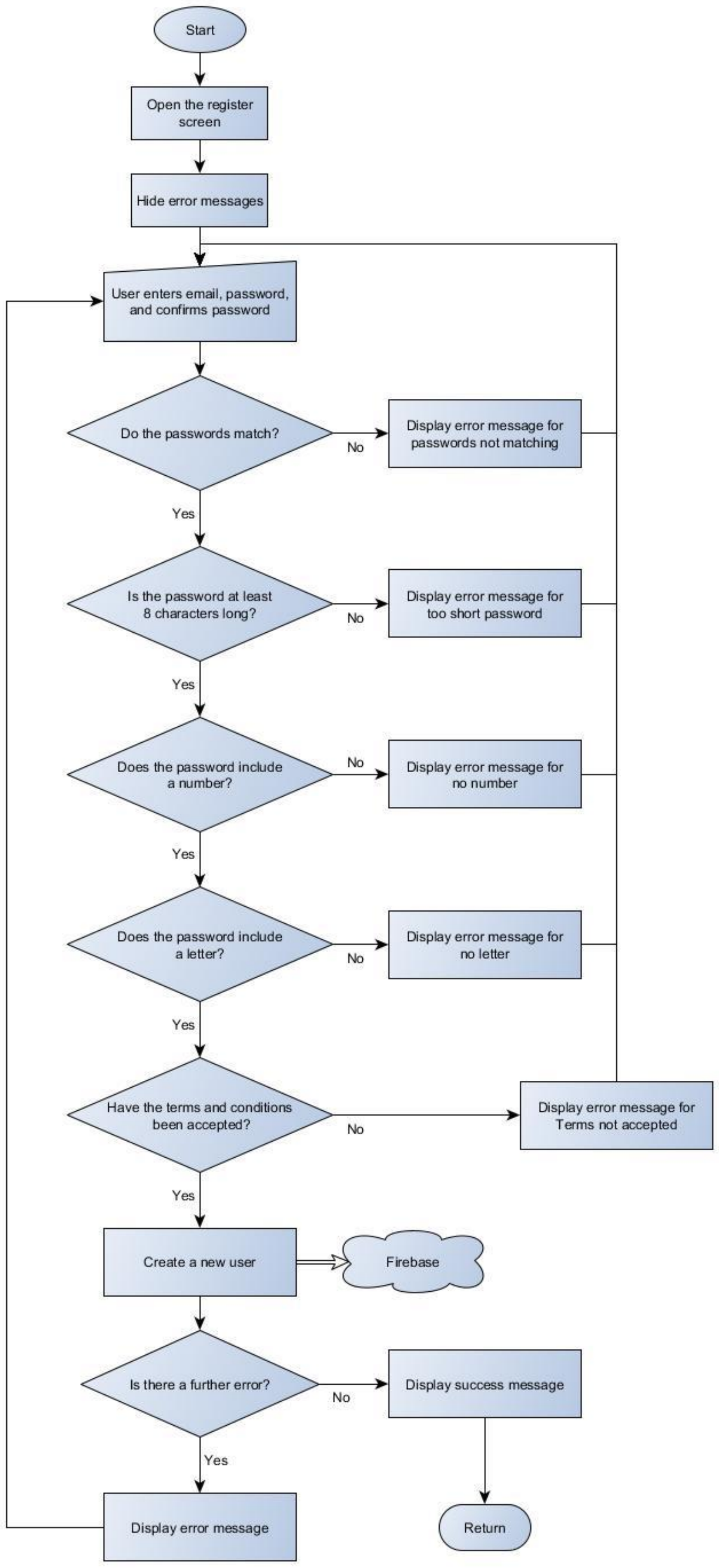

Start
Open the register screen
Hide error messages
User enters email, password, and confirms password
Do the passwords match?
No
Display error message for passwords not matching
Yes
Is the password at least 8 characters long?
No
Display error message for too short password
Yes
Does the password include a number?
No
Display error message for no number
Yes
Does the password include a letter?
No
Display error message for no letter
Yes
Have the terms and conditions been accepted?
No
Display error message for Terms not accepted
Yes
Create a new user
Firebase
Is there a further error?
No
Display success message
Yes
Display error message
Return

Figure 2: Register sub-process flowchart

In the register sub-process, the register screen is first opened and all error messages are hidden. The user enters new user details and these are checked for validity. If any of the checks fail, an error message is displayed and the user has to input new user details. Otherwise a new account is created and a success message is displayed.

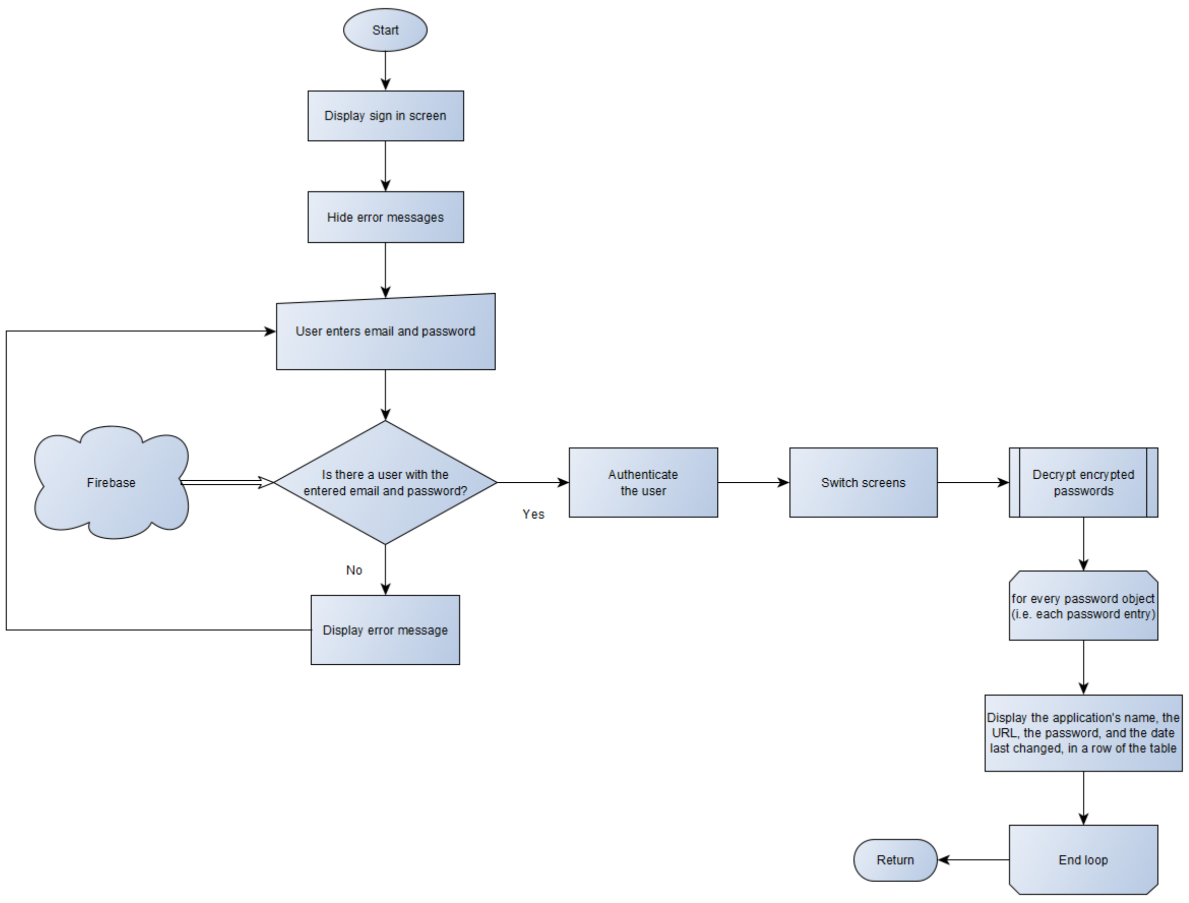

Start
Display sign in screen
Hide error messages
User enters email and password
Firebase
Is there a user with the entered email and password?
Yes
No
Display error message
Authenticate the user
Switch screens
Decrypt encrypted passwords
for every password object (i.e. each password entry)
Display the application's name, the URL, the password, and the date last changed, in a row of the table
End loop
Return

Figure 3: Sign in sub-process flowchart

In the sign in sub-process, the sign in screen is first displayed, and error messages hidden. The user enters their credentials. If such a user exists, the user is authenticated and taken to a new screen. The user's password list is decrypted, which yields an array of Javascript objects that contain names and passwords of applications. This array is then iterated through and the passwords for different applications are displayed on the new screen.

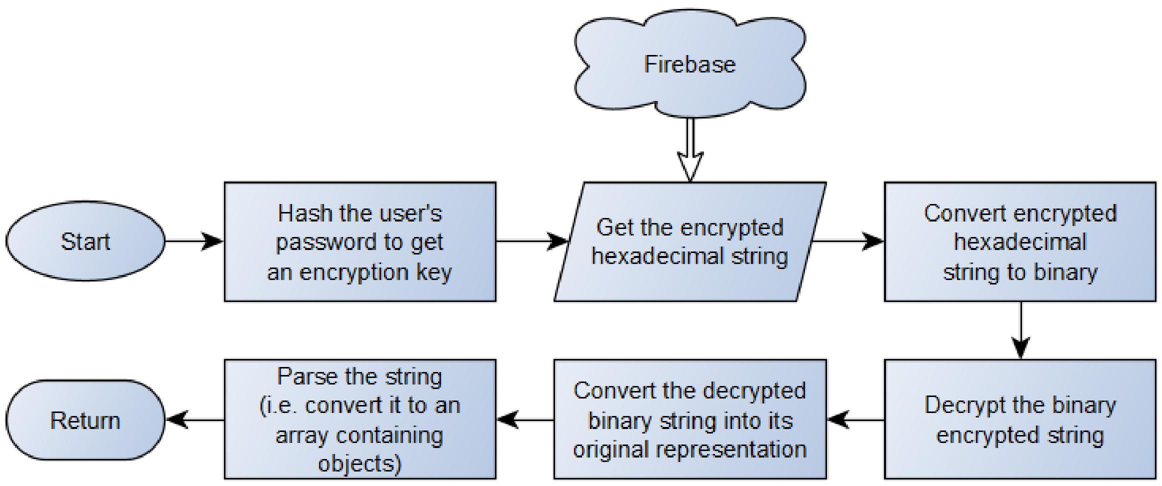

Firebase
Start
Hash the user's password to get an encryption key
Get the encrypted hexadecimal string
Convert encrypted hexadecimal string to binary
Decrypt the binary encrypted string
Convert the decrypted binary string into its original representation
Parse the string (i.e. convert it to an array containing objects)
Return

Figure 4: Decrypt encrypted passwords sub-process flowchart

The user's passwords are stored in encrypted form in the cloud using Firebase. The same encryption key is used to encrypt and decrypt them. This encryption key is obtained by hashing the user's password. Next, the encrypted string is retrieved from Firebase, and converted into binary. The binary string is decrypted, converted into text, and then into Javascript objects, ready to be used.

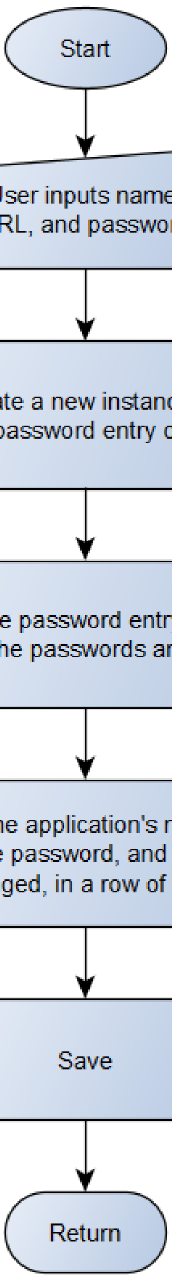
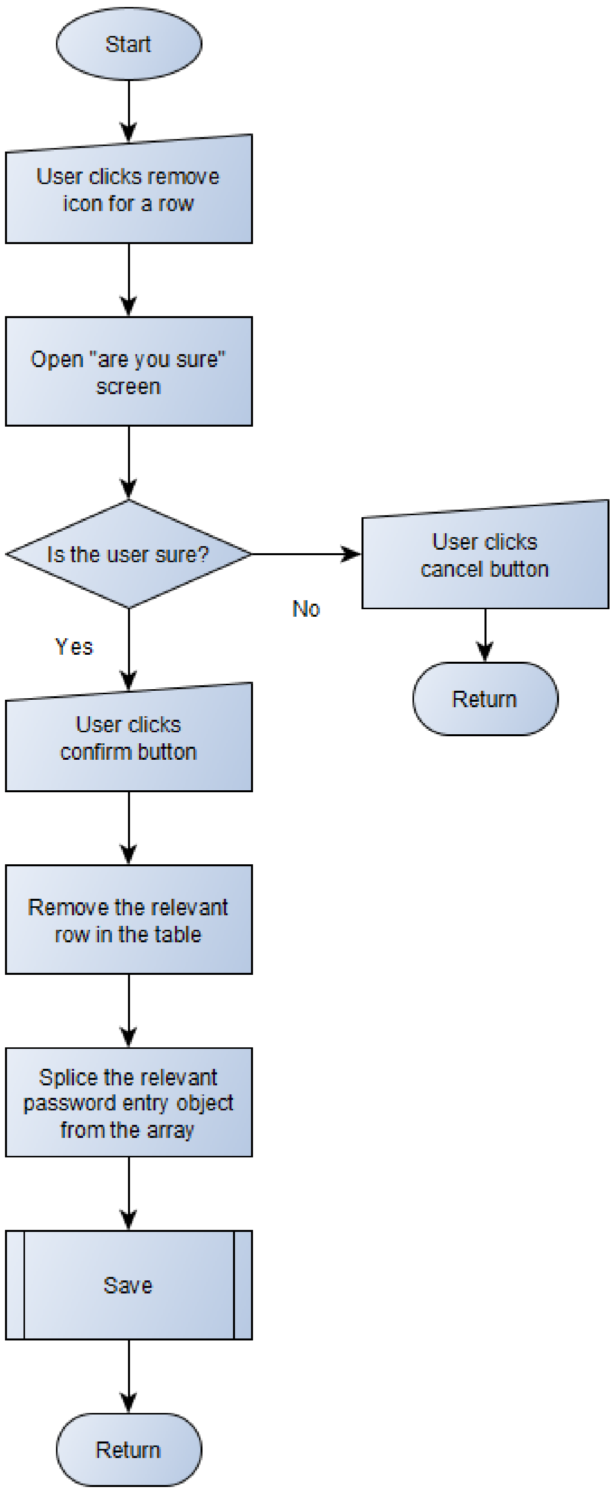

Start
User inputs name, URL, and password
Create a new instance of the password entry class
Push the password entry object to the passwords array
Display the application's name, the URL, the password, and the date last changed, in a row of the table
Save
Return

Figure 5: Add new entry sub-process flowchart

To add a new entry, the user enters the details he wishes to add. Then, a new instance of the password entry class is created, and this object is added to the array containing all the passwords. In addition, the details are displayed in a row of the table on the screen. The passwords are then saved.



Start
User clicks remove icon for a row
Open "are you sure" screen
Is the user sure?
No
User clicks cancel button
Return
Yes
User clicks confirm button
Remove the relevant row in the table
Splice the relevant password entry object from the array
Save
Return

Figure 6: Remove entry sub-process flowchart

As per the client’s wishes, clicking on the delete button will not instantly delete a password entry. Rather, the user is prompted to confirm the action. If the user confirms, the relevant password entry is removed both from the table on-screen, and from the array containing all the passwords, and the array is then saved.

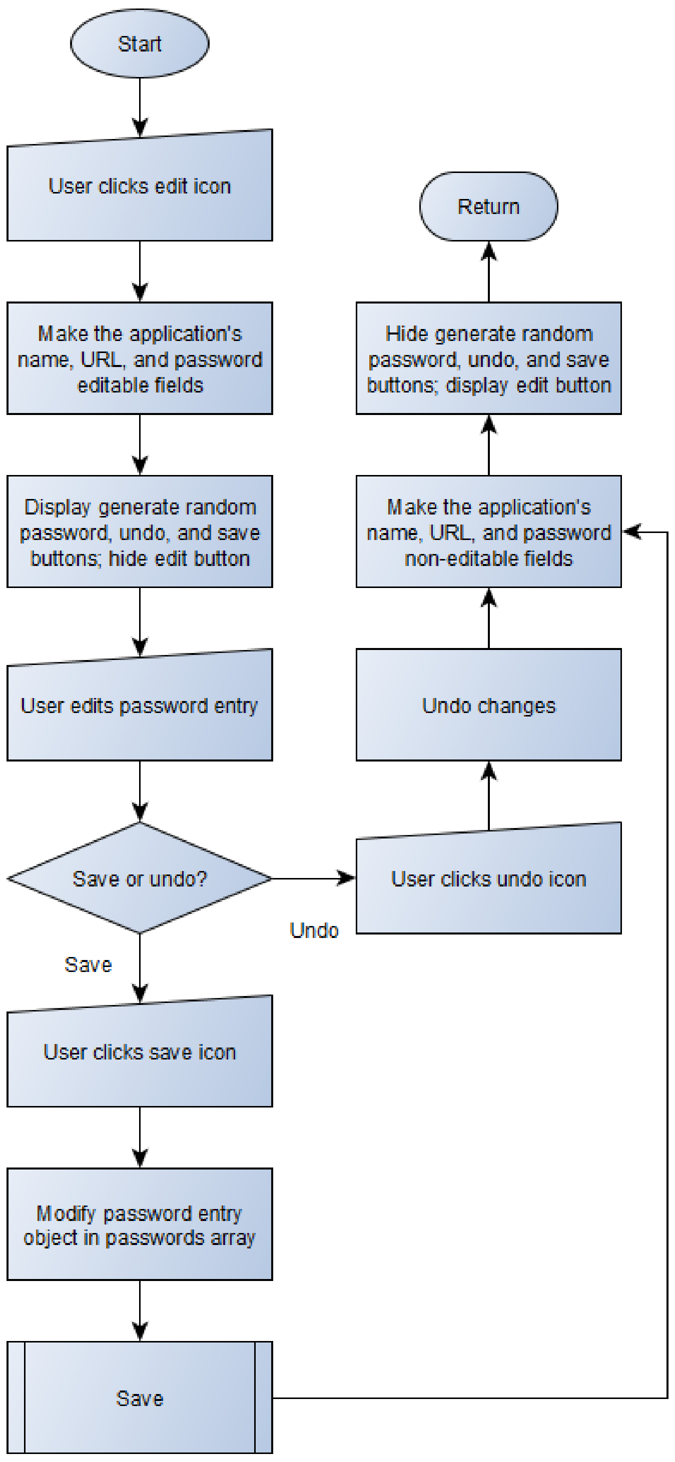

Start
User clicks edit icon
Make the application's name, URL, and password editable fields
Display generate random password, undo, and save buttons; hide edit button
User edits password entry
Save or undo?
Save
Undo
User clicks save icon
Modify password entry object in passwords array
Save
User clicks undo icon
Undo changes
Make the application's name, URL, and password non-editable fields
Hide generate random password, undo, and save buttons; display edit button
Return

Figure 7: Edit entry sub-process flowchart

If the user wishes to edit a password entry, he clicks the edit icon. This makes it possible to edit several fields. A save and an undo button will replace the edit button. The user can then edit the fields. To save changes, the save icon should be clicked. The password entry object in the passwords array will then be modified and saved. Alternatively, if the user changes his mind, he can click the undo icon, which reverts all changes. Ultimately, all fields are made non-editable and the buttons are reverted.

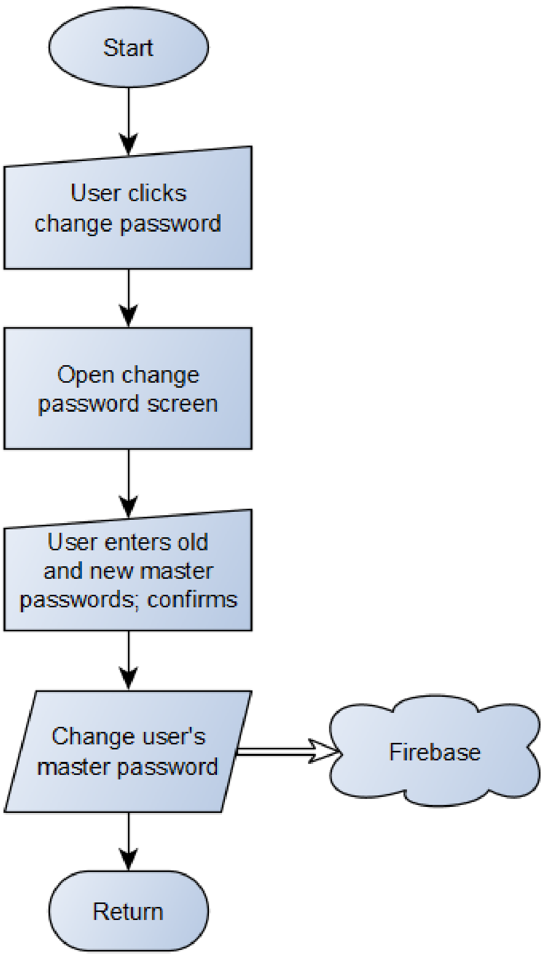

Start
User clicks change password
Open change password screen
User enters old and new master passwords; confirms
Change user's master password
Firebase
Return

Figure 8: Change master password sub-process flowchart

The user can change the master password by clicking on a button which opens a dedicated screen. Once the user makes changes and confirms, his password is changed and Firebase is updated.

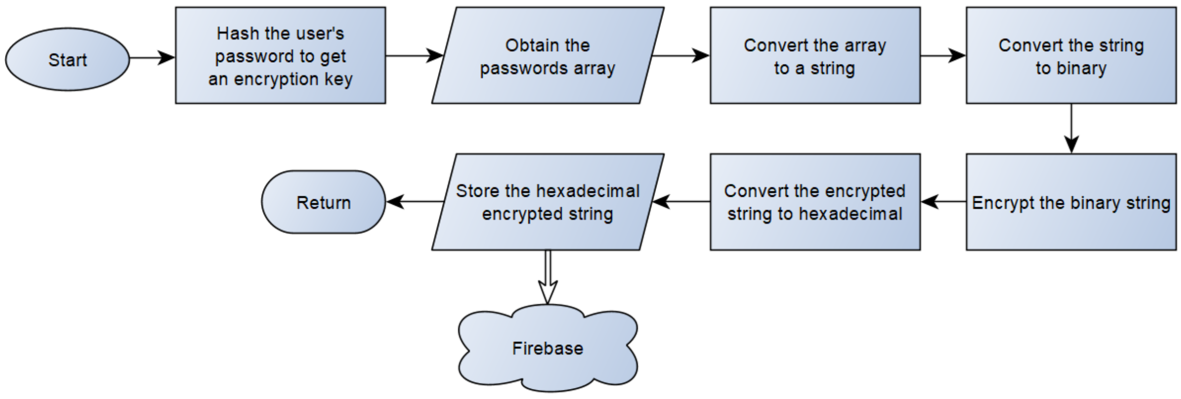
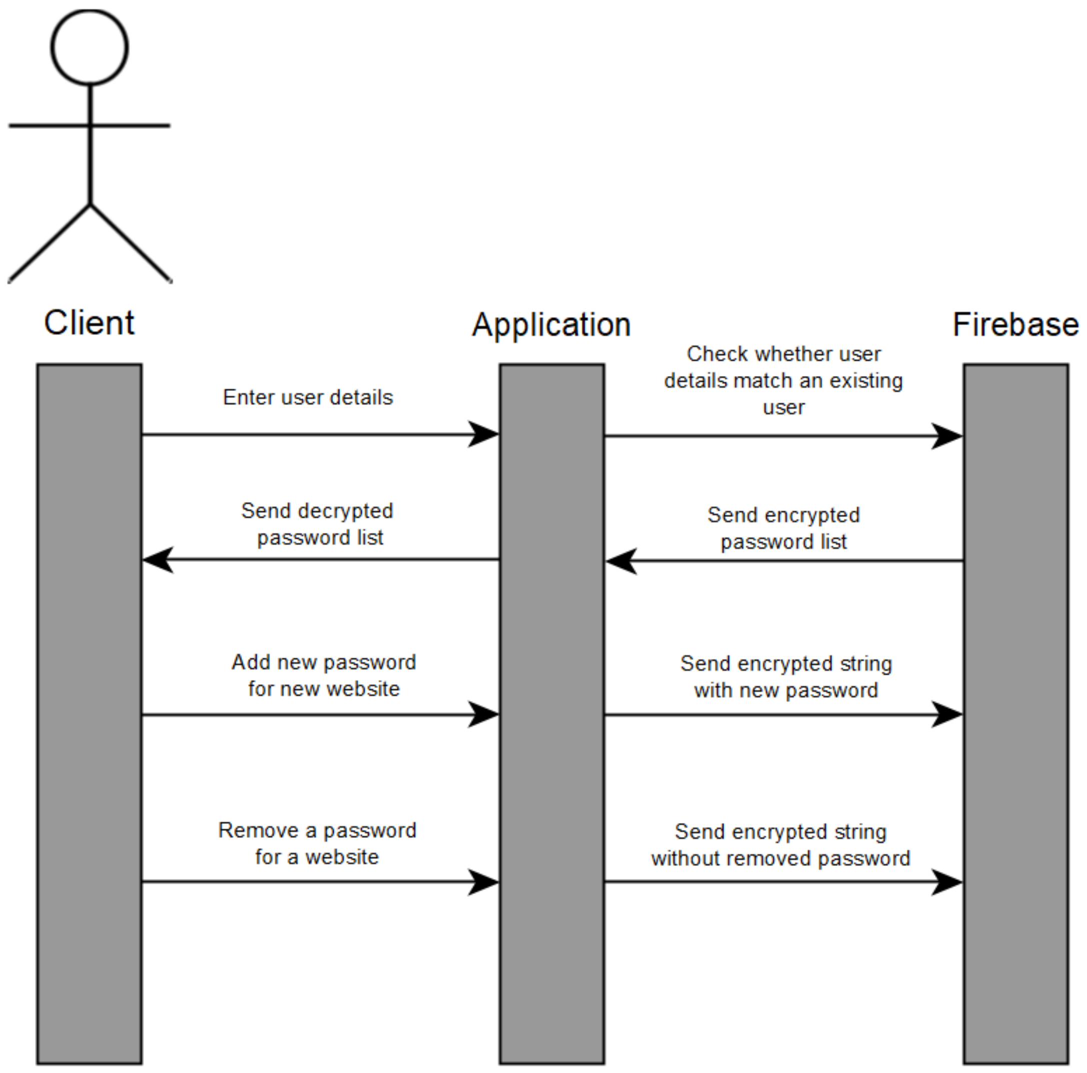

Start
Hash the user's password to get an encryption key
Obtain the passwords array
Convert the array to a string
Convert the string to binary
Encrypt the binary string
Convert the encrypted string to hexadecimal
Store the hexadecimal encrypted string
Firebase
Return

Figure 9: Save and encrypt sub-process flowchart

In the save and encrypt sub-process, the user's password is first hashed to obtain the encryption key. The passwords array is converted to a binary string, and is then encrypted using the key. The binary string is changed to hexadecimal form and stored in Firebase.



Client
Application
Firebase
Enter user details
Check whether user details match an existing user
Send decrypted password list
Send encrypted password list
Add new password for new website
Send encrypted string with new password
Remove a password for a website
Send encrypted string without removed password

Figure 10: UML diagram

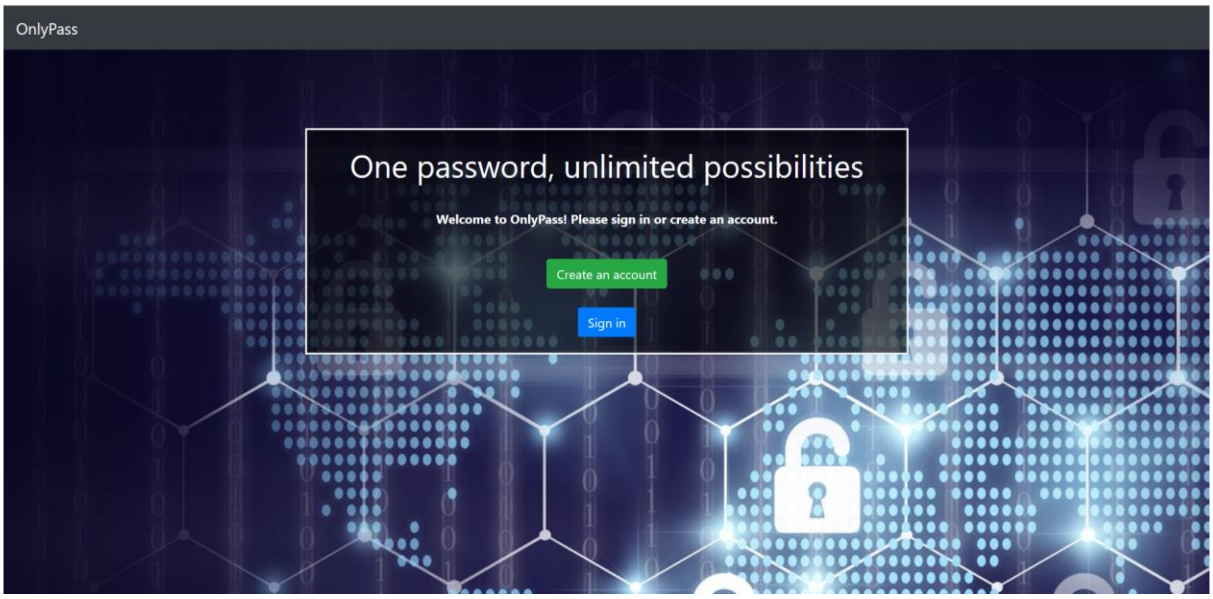
Firebase will be used for user authentication and storing the encrypted strings. Firebase thus acts as the server in the program. The UML diagram shows some of the information flow between the client, the front-end application, and back-end Firebase.

For example, when the client wishes to sign in, he enters his user details. The application then sends a request to Firebase to check whether there is a user with the entered credentials. If so, Firebase finds the relevant encrypted password list, and send it to the front-end application. The application then decrypts the list, and displays it to the client.

Should the client wish to add or remove a password, this likewise first goes through the application, which modifies the password list and encrypts it. The modified encrypted list then gets sent to Firebase to be stored. It is important that only encrypted information ever gets sent to Firebase, so that the security of the program is not compromised.

User Interface Design

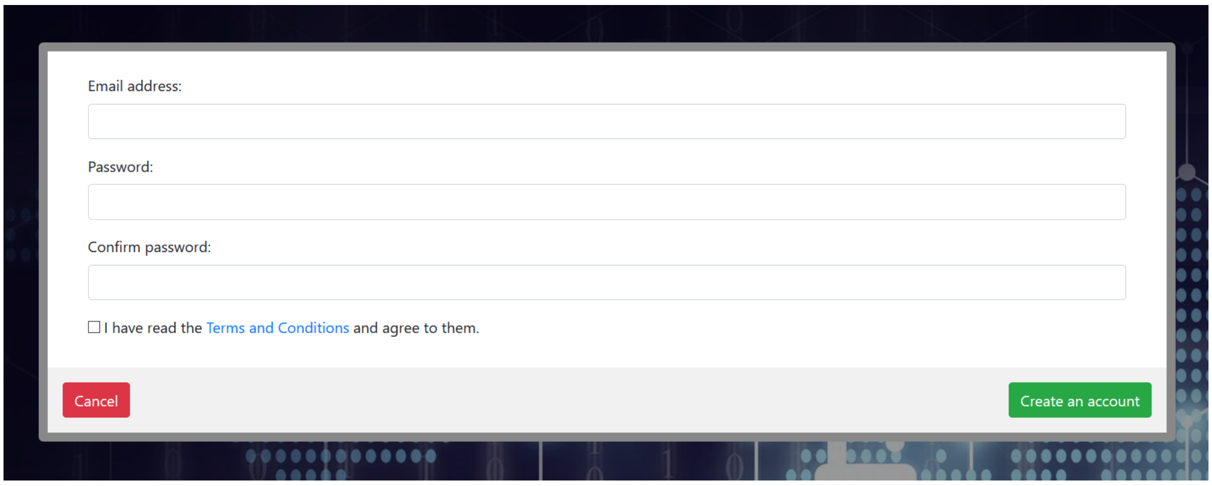
The figures below show how the application will look like. Figure 11 shows the screen the user will first see upon opening the application. Figure 12 shows the popup that will appear when the *Create an account* button is clicked. Similarly, Figure 13 shows the popup produced by clicking on *Sign in*. Clicking the cancel button, or anywhere outside of these popups, will close them.



One password, unlimited possibilities

Welcome to OnlyPass! Please sign in or create an account.

Figure 11: Starting screen

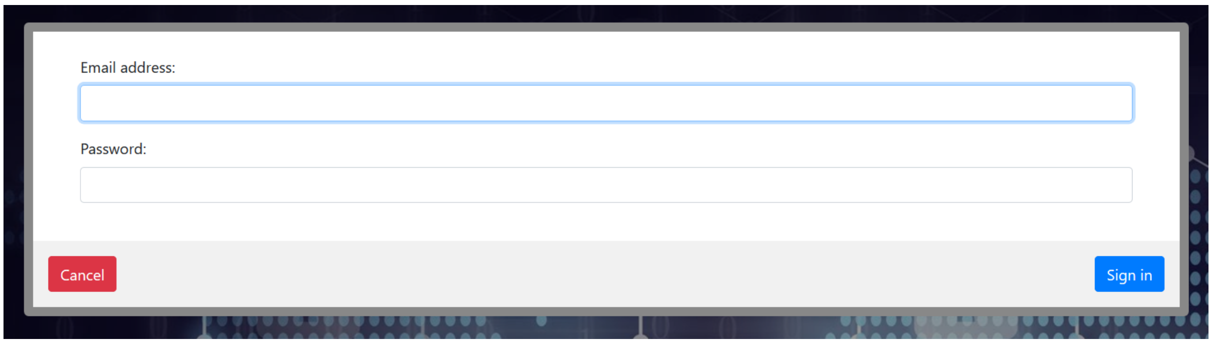

Email address:

Password:

Confirm password:

☐ I have read the Terms and Conditions and agree to them.

Figure 12: Create an account screen

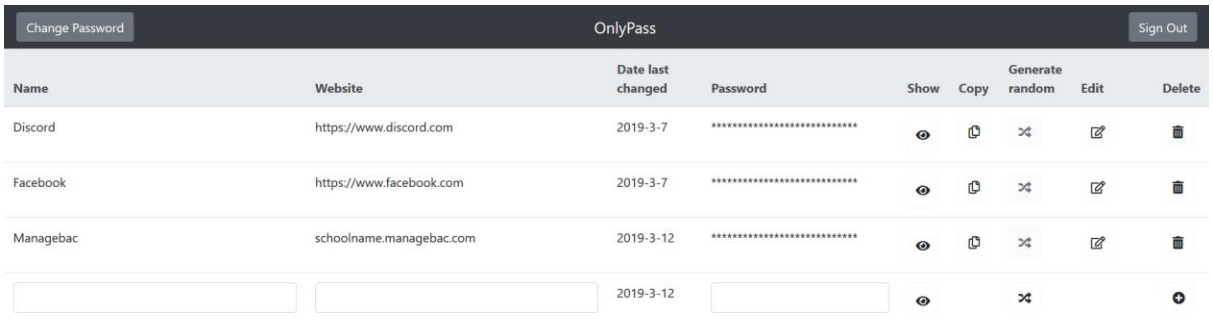


Email address:

Password:

Figure 13: Login screen

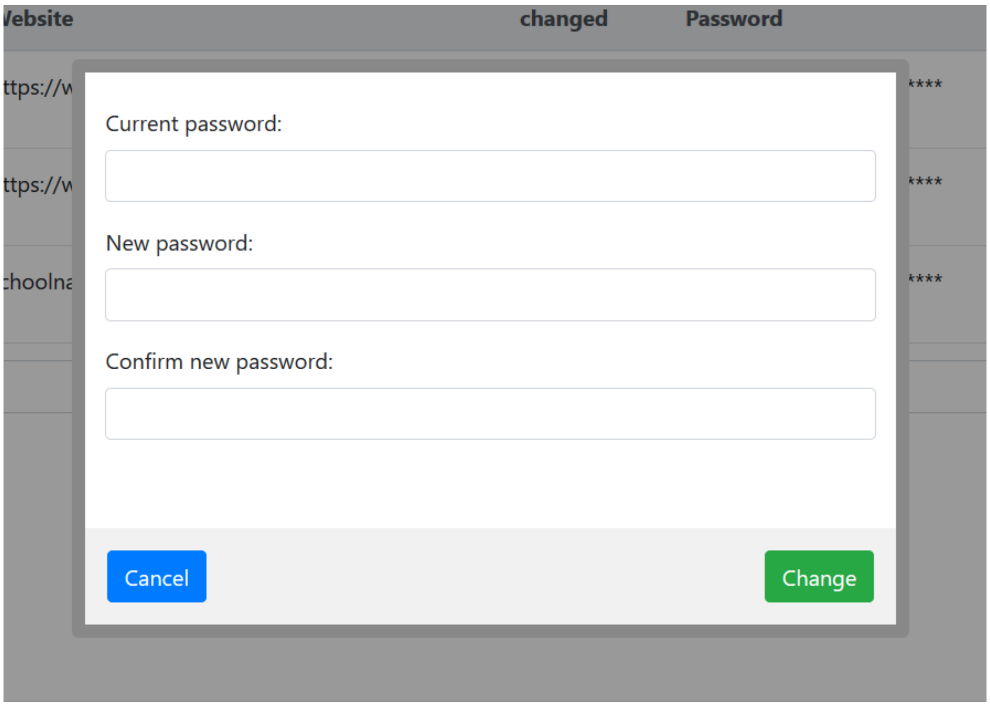
Once the user signs in, the screen shown in Figure 14 will be displayed. This is where the user will be able to view passwords and make changes.



Name	Website	Date last changed	Password	Show	Copy	Generate random	Edit	Delete
Discord	https://www.discord.com	2019-3-7	******************************	👁	⧉	⤨	✎	🗑
Facebook	https://www.facebook.com	2019-3-7	******************************	👁	⧉	⤨	✎	🗑
Managebac	schoolname.managebac.com	2019-3-12	******************************	👁	⧉	⤨	✎	🗑
		2019-3-12		👁		⤨		⊕

Figure 14: Password list screen

To change the master password, the *Change Password* button in the top-left corner needs to be clicked. This will open the popup in Figure 15.



Current password:

New password:

Confirm new password:

Figure 15: Change master password screen

Test Plan

Action to be tested	**Test method**
The program stores different passwords for different websites under AES-256 encryption	Check whether the program produces an unrecognizable hexadecimal string when the passwords are encrypted.
The client finds the product easy to use and user-friendly	Give the product to the client to try out for a few days, and listen to feedback on whether the client found it easy to use.
The client can add new passwords for different websites	Attempt to use the add function and add passwords for new websites.
The client can delete passwords, but is prompted to confirm deleting so that a password doesn't get deleted by accident	Attempt to press the remove button and remove websites.
The client can edit passwords for already added websites	Attempt to press the edit button, change the password and then save. Sign out and sign back in to see if the changes have been saved.
The client can change the master password	Attempt to change the master password, then sign out and try to log in with the new master password.
Each password has the date that it was last changed displayed next to it, so that the client knows when he should change it	Change an old password and see whether the date changes.
The program can be used across multiple devices, and the passwords saved on one device will show up on the other	Add a password for a new website on one device, and then log in on another device and try to obtain the password.
The client can create an account with an email and passphrase	Attempt to use the register function multiple times with various different inputs, to see whether the system ever breaks down depending on what combination of inputs the client enters. Then try to login to prove that the account has been created.
The program has an inbuilt password generator	Attempt to use the inbuilt password generator, and see whether it produces a password at all, and whether that password is sufficiently secure.