

2023

Final Report

PASSWORD MANAGER
(C00246523) - JASON DARLEY

SUPERVISOR Christopher Staff

Contents

Introduction	2
Project Description.....	3
Login Page	3
Forgot Password.....	5
Register	7
Main Menu.....	9
Passwords.....	10
Password Generator.....	10
Strength Tester	12
Store Passwords	13
View Passwords.....	14
Update Passwords.....	16
Training.....	18
Quizzes	18
Account	19
Change password	19
Logout	20
Description of Conformance to Specification and Design.....	21
Description of Learning.....	22
Technical Learning.....	22
Personal Learning.....	22
Review of Project	23
What Went Right and What Went Wrong.....	23
Missing Aspects of the Project.....	24
What I Would Do Differently.....	24
Advice for Next Years Students	25
Review of Technology Choices	25
Was my Project a Success?	26
Acknowledgements.....	26
References.....	27

Introduction

This report will detail the process of making a Password Manager for my 4th year project. This Password manager is intended to be extremely easy to use so it can be available to anyone. The application is intended/aimed at company use as there are built in training and quizzing programs for various cyber-attack topics including Phishing, Business Email Compromise (BEC), Malware, Ransomware and password security. The training and quizzes were implemented to inform users of how to identify these attacks like being able to spot a phishing link from a legitimate link and it also provides information on how to defend against these attacks if they are ever encountered. The training and quizzes were also added in order to provide free education to companies to help advance their employees knowledge on these attacks as it is common knowledge in cyber security that people are the cause for majority of breaches so this training is designed to prevent or lower the possibility of a successful attack that can lead to a breach.

I decided to create a password manager as it would be used for my own personal use and to help aid companies prevent attacks from occurring. From personal experience in the I.T. sector and from what I've heard from classmates they had received cyber security training on their 3rd year work placements whether it be recurring training or a once off training at the beginning of their placement, and from research in 2021 only 35% of Irish small and medium sized enterprises (SME's) use cyber security training in the workplace. Now whether the lack of training is due to companies not wanting to pay for the training my password manager will provide the training for free while also allowing users to store passwords securely in hopes of increasing the percentage of companies using cyber security training.

The password manager stores user's data securely by using hashing or encryption and also only requires as much information from the user to use each service of the application required, nothing more which contributes to the compliancy of the GDPR. In order to ensure the security of this data each user when creating their account is assigned a random key for encryption and decryption which means only the key's owner can access the passwords they've stored. There are also security measures like brute force prevention and Multi factor authentication implemented within the application to help prevent any attackers from gaining access to someone else's data and/or account.

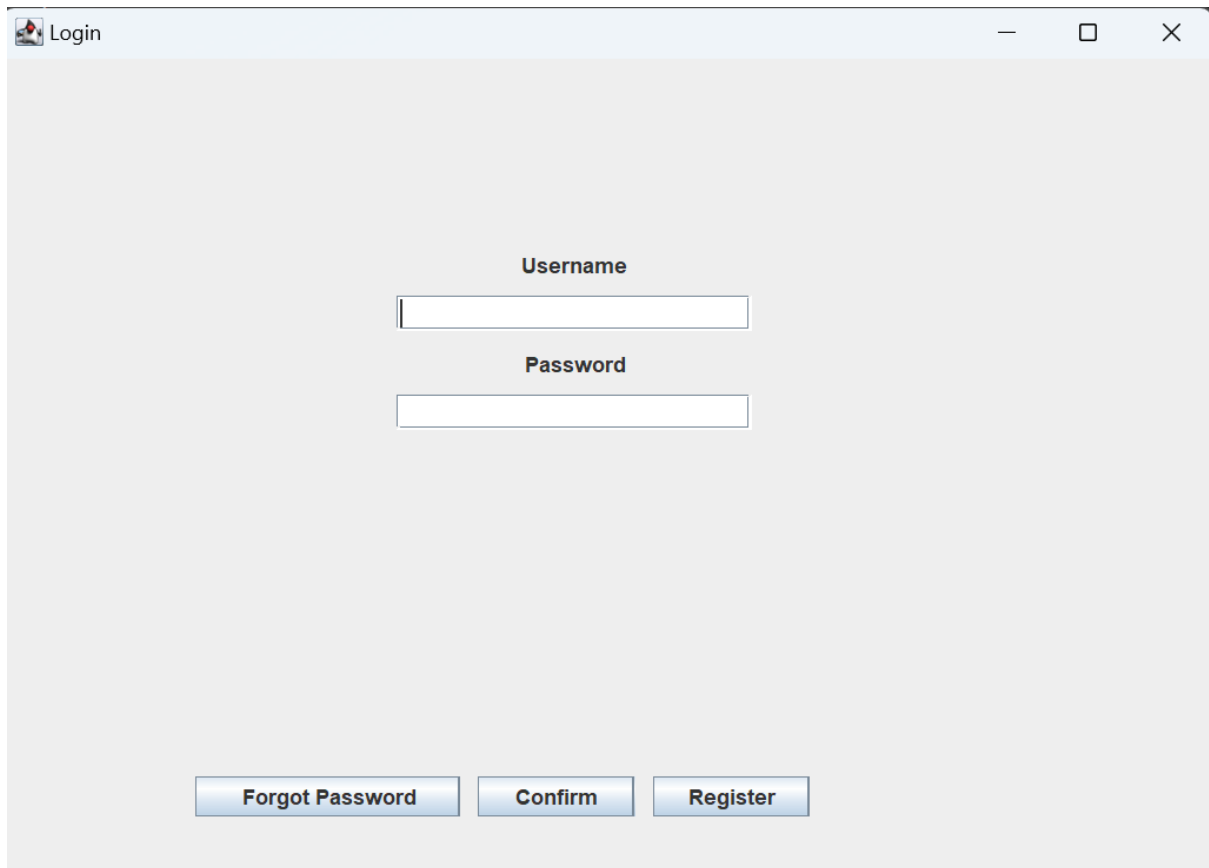
Project Description

This project was made using Java and was coded in Eclipse using MySQL to store the user's data on a local server. Java swing was also used in order to create all of the Gui for the project in order for it to be easy to navigate and visually pleasing to users instead of having a complicated navigation and design which would cause users to stray from using this application. The project also required JAR files for Bouncy Castle, MySQL, and 2 jar files for sending emails through eclipse. The MySql server consists of 4 tables: account, mfa, loggedIn, and password. The account table consists of first_name, last_name, user_name, password, email_id, user_key and mobile_number these are all used to store the data given by the user when creating an account and their randomly generated key for encryption and decryption. The mfa table consists of mfa_code which contains the 6-digit code that gets randomly generated. The loggedIn table consists of Id and user_name and is used to track which user is logged in as this application works remotely so only 1 user can be logged in on each device. The password table consists of website, site_username, password and user_name to store the information of the account of the 3rd party site the user wishes to store and the name of that site along with the username of the account that's storing the details so the application will only show each users details and nobody else's ensuring the CIA model (Confidentiality, Integrity, Availability).

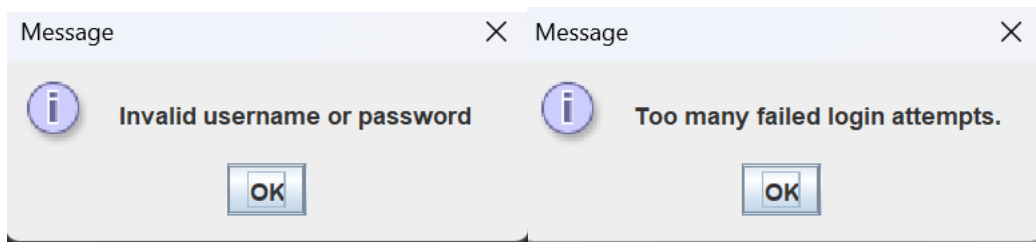
I will describe each part of the project in the order you can access them so the login page first then the registration page and then the main menu etc...

Login Page

The following is a screenshot of the Login page of the password manager:



The login page is the first page a user will see when the program is running as it makes it more convenient to the user as they won't want to see the register page constantly and have to switch to the login page every time. The login page also allows a user to sign into their account if they have created an account already by entering their username and password for the account into the JTextField for the username and the JPasswordField for the password and then clicking the confirm button, the password field makes it so whatever the user enters it will appear as * instead of plaintext in order to prevent shoulder surfing which is a social engineering attack technique. When a user logs in their username will also be updated into the logged in table to track which user is logged in to help other pieces of code work like view passwords. The JPasswordField also prevents someone from copy pasting the text onto a different file in hopes to view the plaintext as it doesn't allow the text to be copied. These pieces of data are then hashed using sha256 and compared to what is stored in the account table on the MySQL server and if they match the details of a stored account, they are granted access and are then forwarded to an MFA page to ensure the user is who they say they are. In the case of a failed login, they will receive an error message stating that username or password is incorrect and if the user has 5 failed logins the program will shut down in order to prevent brute force attacks. The following are images of what a user will see when they encounter these scenarios from unsuccessful logins.

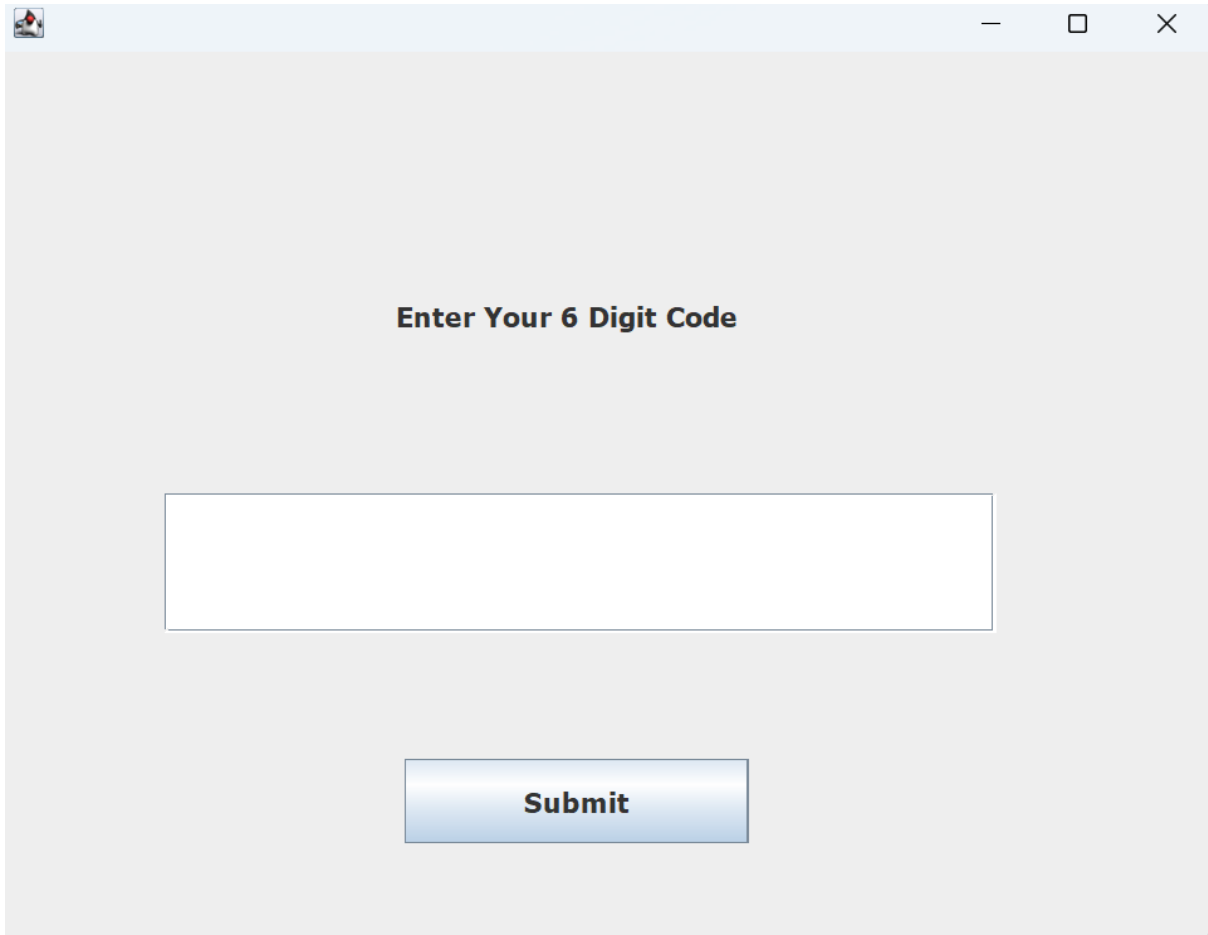


Forgot Password

If the user clicks the forgot password button will redirect the user to the following page

A screenshot of a web page titled "change password". The page has a light gray background. At the top, it says "change password" in bold black text. Below that, it says "enter your email" in bold black text. In the center, there is a white rectangular text input field with a vertical cursor on the left. Below the input field is a blue "Send" button with white text.

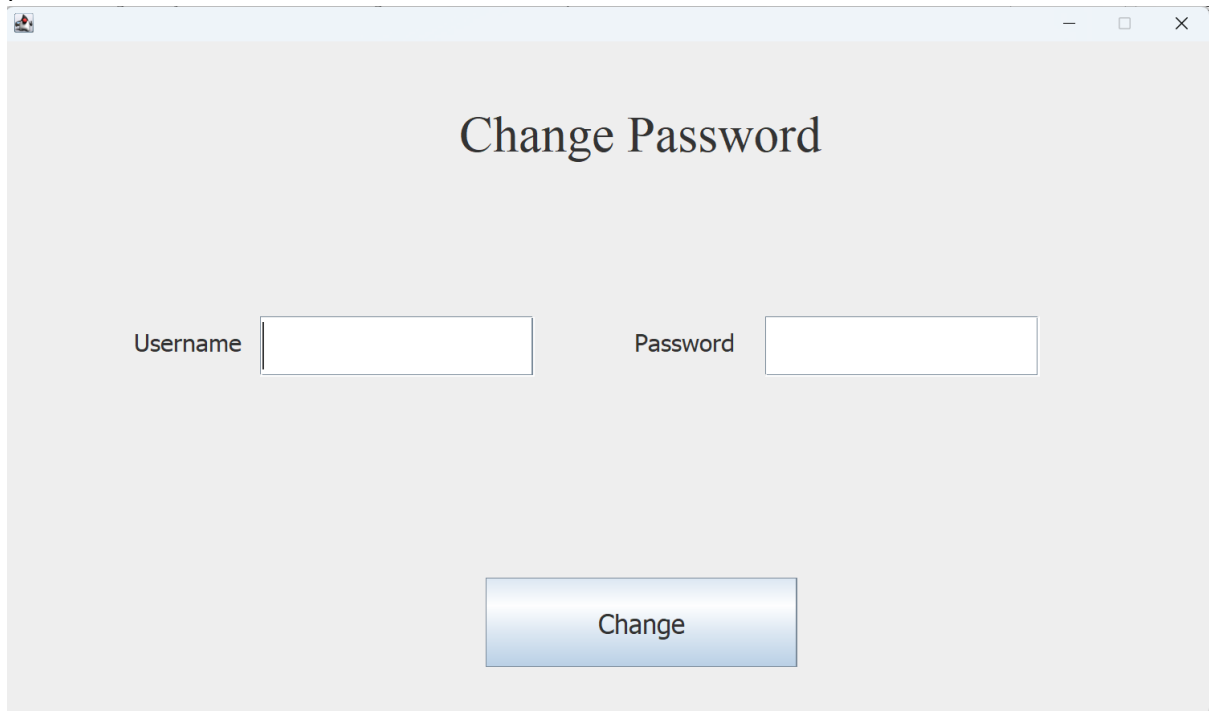
At this page the user will enter their email and will then receive a 6-digit mfa code that has been randomly generated and entered into an email to be sent to the user and the code is also stored in the mfa table in the MySQL server. If the user enters an email that doesn't exist they will receive an error message but if the email sends successfully it will then redirect them again to a page where they can enter their random 6 digit mfa code in order to be granted access to change their password. Here is how that page looks:



The image shows a browser window with a light gray background. At the top, there is a title bar with a small icon on the left and standard window controls (minimize, maximize, close) on the right. The main content area is centered and contains the text "Enter Your 6 Digit Code" in a bold, black font. Below this text is a large, empty white rectangular input field with a thin gray border. At the bottom center of the form is a blue button with a white border and the word "Submit" in white text.

Here the user will enter the code and it will compare what was entered with what is stored in the mfa table and if they don't match, they will receive an error saying invalid mfa code and if the codes do match, they will be redirected again to a page where they can change their password for their account. The following is the page where the user can change their

password:

A screenshot of a web browser window titled "password:". The window displays a "Change Password" form. The form has a light gray background and a white border. At the top center, the text "Change Password" is written in a large, black, serif font. Below this, there are two input fields: "Username" on the left and "Password" on the right. Each label is followed by a white rectangular input box with a thin black border. At the bottom center of the form, there is a blue button with a white border and the text "Change" in white. The browser window's title bar shows standard minimize, maximize, and close buttons.

This page works by the user entering their username and then the new password they want to change the password to for their account. This code implements password complexity rules in order to ensure user's have strong passwords to keep their accounts safe. The passwords must be at least 8 characters long and must include a number and at least 1 upper case and 1 lower case character otherwise they will receive a message stating the password requires whatever aspect is missing. In terms of actually changing the password the code will take the username entered and hash it and then compare this hashed value to the account table, if the hashed username is found in the table, then it will update the password for this account with that username with the hashed new password entered by the user. After successfully changing the password of an account the user is then redirected back to the login page.

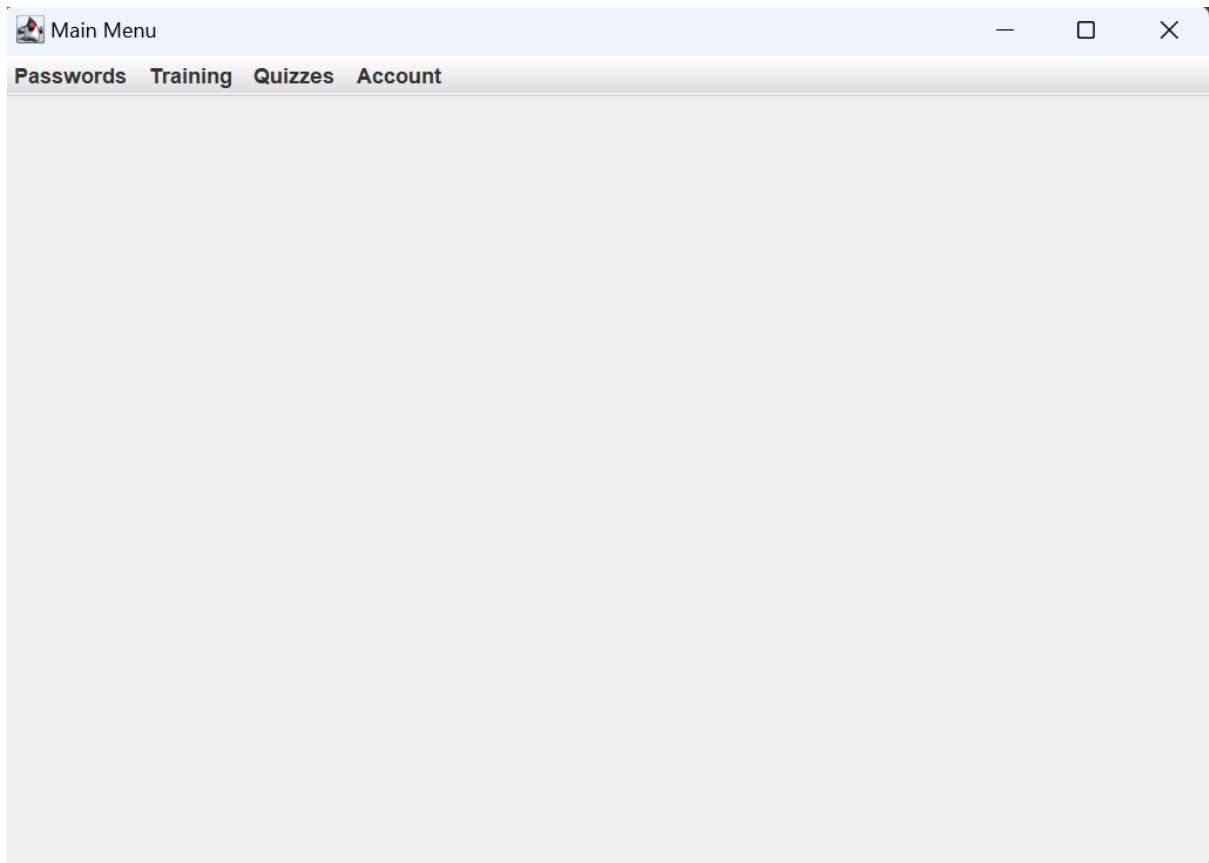
[Register](#)

The register button will redirect the user to the register page where they can create an account for the application. The following is a screenshot of the register page:

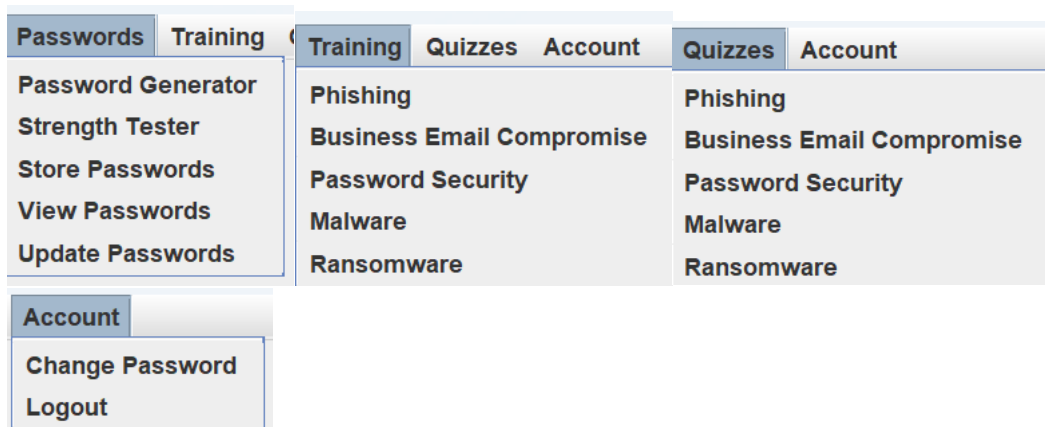
The image shows a web browser window with a light gray background. At the top center, the title "Create Account" is displayed in a large, bold, black font. Below the title, there are six input fields arranged in two columns. The left column contains "First name", "Last name", and "Email address". The right column contains "Username", "Password", and "Phone number". Each field is a simple white rectangle with a thin gray border. Below the input fields, there are two blue buttons with white text. The left button is labeled "Register" and the right button is labeled "Go To Login Page". The browser window has standard window controls (minimize, maximize, close) in the top right corner.

Here the user has the option to register or go back to the login page in the event that they accidentally clicked the register button, and they have a way back instead of having to restart the program again. The register page works by having the user enter their details into the text fields and password field and then clicking the register button. Like the forgot password page to change an accounts password the registration page implements the same password complexity rules to ensure the user has a strong password for their account. If any of the fields are left empty the user will receive a message saying, "please fill in all fields" and won't allow the user to create an account until the fields are filled with valid data. It also ensures that a valid phone number is entered in the phone number field also printing an error message to the user to enter a valid number. It also ensures that no 2 users can have the same username by telling a user if the username they tried to use for their account exists already and that they must choose another. If the user has valid data in all the fields and they click the register button all their information will be hashed using sha256 and stored in the account table. The code will also use a random generator in order to create a unique key for each user in order to store and view passwords using aes-cbc mode. Only the owner of this key can use it to store and view their passwords in other parts of the application which will be shown later in this document. After successfully registering the user will then be brought to the login page where they can authenticate using the details they chose and gain access to the full application and its services. Here is the page the user is redirected to after successfully authenticating at the login page and confirming the mfa code:

Main Menu



Here is the main menu for the application. This frame consists of a menu bar with 4 dropdown boxes called passwords, training, quizzes, and account. The password dropdown consists of 5 buttons to be redirected to a password generator, a strength tester, the store passwords frame, the view passwords frame, and the update passwords frame. The Training and Quizzes dropdowns both consist of Phishing, Business Email Compromise, Password Security, Malware and Ransomware. Finally, the Account dropdown consists of 2 buttons one for changing the user's account password and one to logout. The following pictures are of these dropdown boxes:



Each of these buttons will open their respective frames except for the logout button that will bring the user back to the login frame and prevents the user from gaining access to the application unless they re-authenticate.

Passwords

Password Generator



Your Password Is:

Ko!2uUZ\$VX{KXeE

This Password Is Very Strong

Weak Moderate Strong

10 15 20 25 30

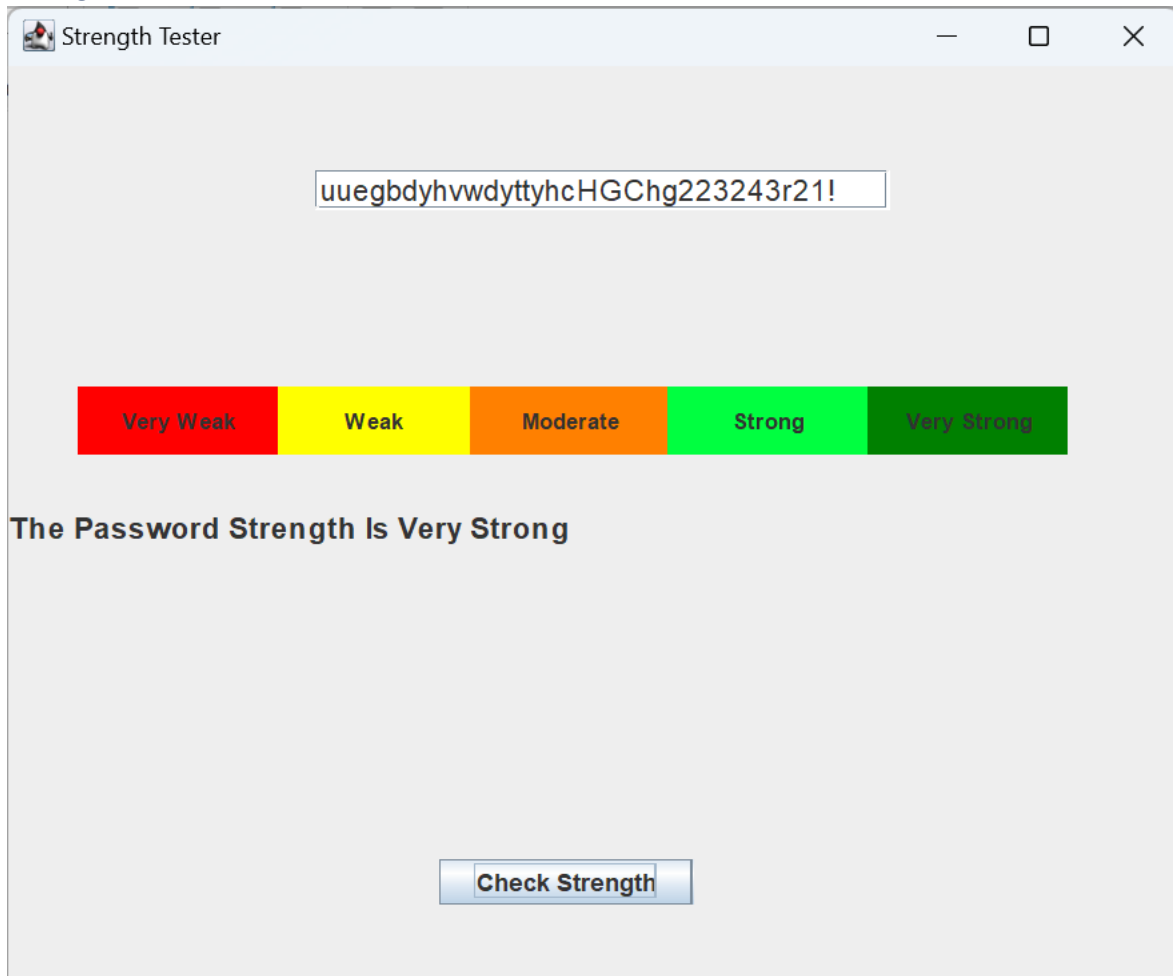
A Strong password should be random and contain Upper case, Lower case and Special characters and should also include Numbers. If one of the generated passwords says its weak you should generate a new password but, know a weak password on this generator is still a lot stronger than using a word even if you think of a random word for a password. By using a word hackers can perform dictionary attacks which is where a computer will enter every dictionary word into the login system. and if your password is a word found in a dictionary it takes very little time to hack into your account.

Generate



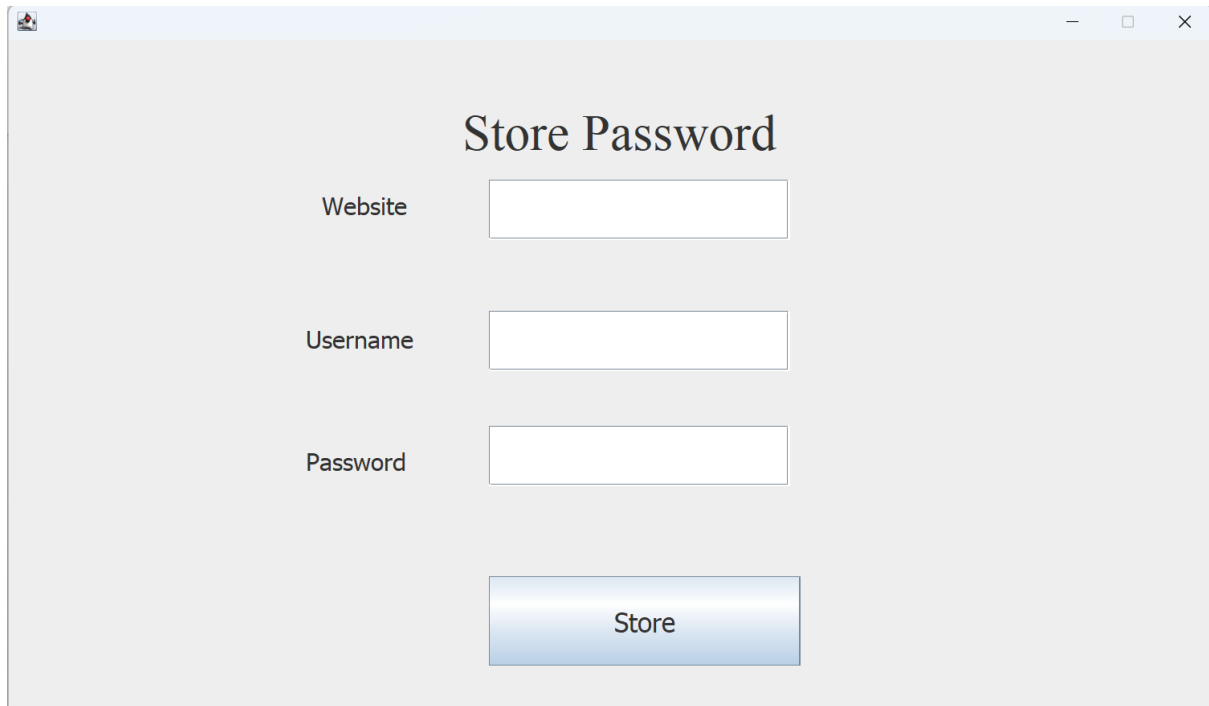
The previous 2 pictures show the password generator. This code works by using a random generator to randomly select a character from specific characters such as upper, lower case character special character and digits. The user will move the slide bar to their desired length and this length will be the length the password is generated so in the first picture a 15 character password is generated as the slider is set to 15 and in the second picture it is set to generate a 25 character password. When the password is generated, text will appear at the bottom explaining what makes a strong password. The strength meter will also be displayed on each password generated as shown as weak/moderate/strong. The code will ensure only strong passwords will be generated as the code makes it so each password must include each character type specified earlier. The strength meter works by using an internal strength tester that verifies which character type is present in the password and if it includes all types then the password will be considered strong. The password is generated and then displayed in the text field where the user has the ability to copy and paste the password for easy transfer to a different application/website to use the password for their account and also to store the password in the password manager application. The slide bar also has little indicators shown as blue | on the slide bar. The bigger | means its an increment of 5 and the smaller | means its an increment of 2 so 10, 15, 20 etc will have a big | and then 12,14,16,18,22,24,26,28 will have a smaller |.

Strength Tester



Here is the strength tester frame. This code also uses an internal strength tester in order to check the strength of the user's password. This code works by having a user enter a password and clicking the check strength button, it will then display the strength meter based on the strength so it can show whichever strength the submitted password is to the user, so they know to make it stronger. This works by giving 5 points for a password and for each point it will unlock a new strength, in order to get the 5 points the password must be at least 10 characters long and includes each character type stated previously. If all 5 points have been identified in the password it will earn a very strong mark indicating to the user, that they can use this password for sites and applications, and it will be a lot safer than most passwords. The user again can copy and paste the password giving them convenience when trying to make an account instead of having to retype the entire password into the site.

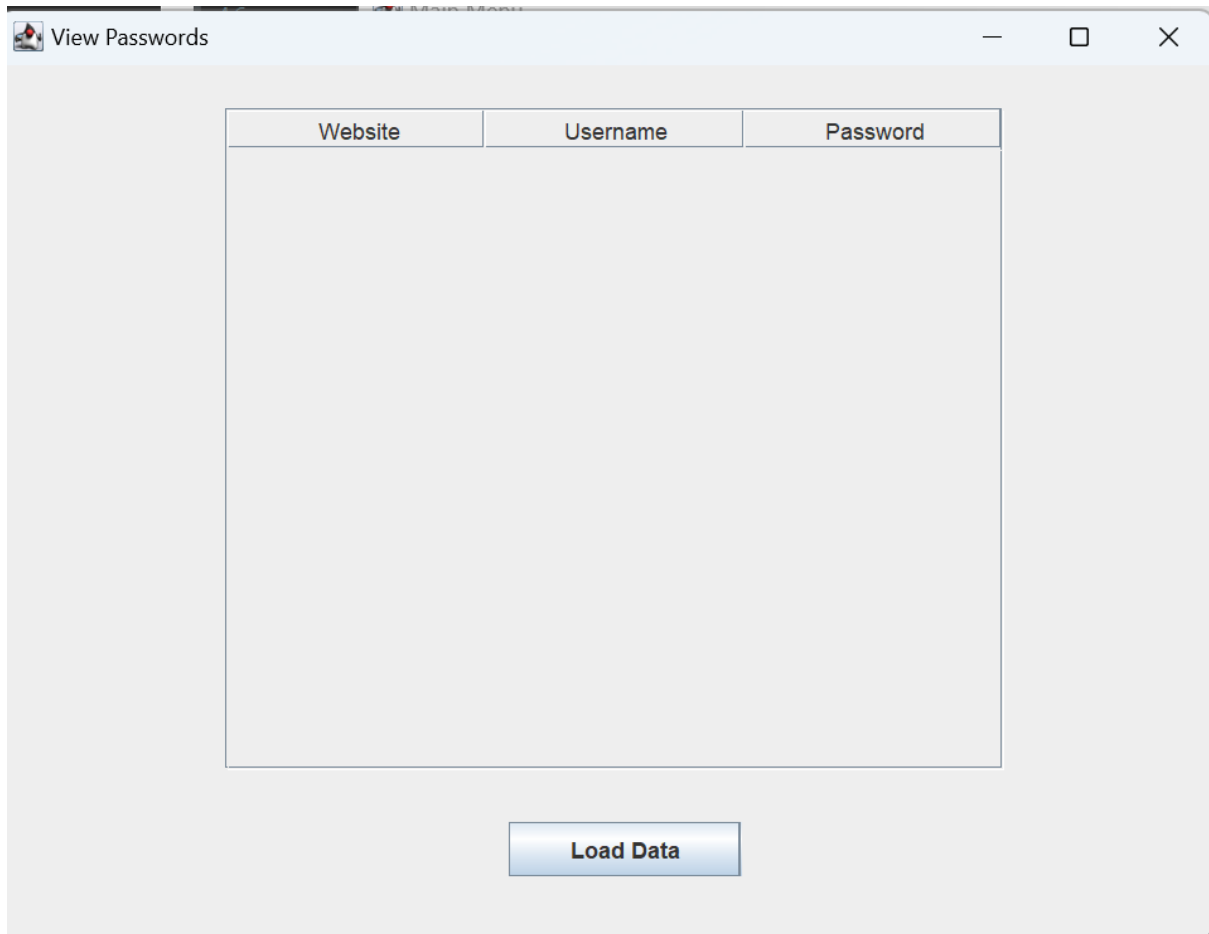
Store Passwords



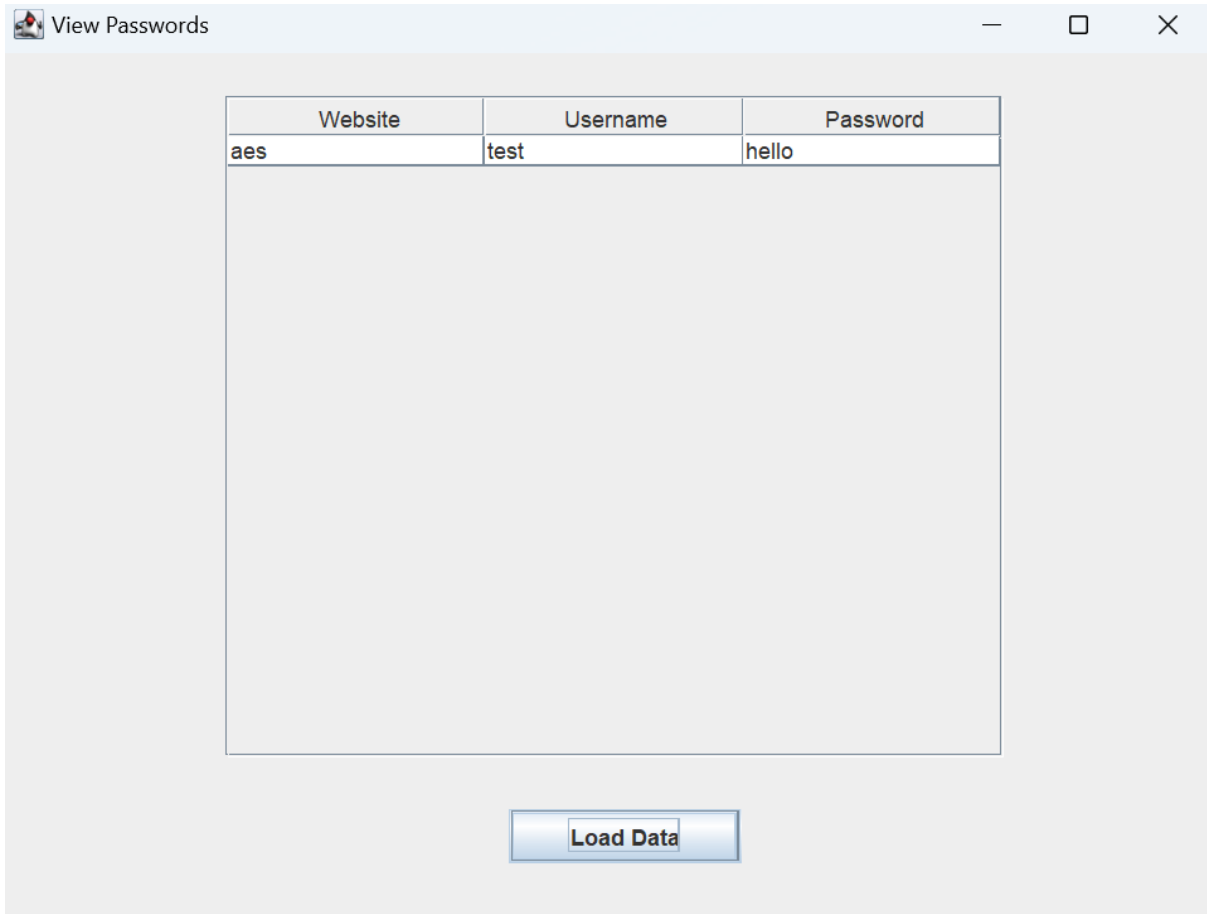
The image shows a web browser window titled "Store Passwords". The main content area has a light gray background and is titled "Store Password" in a large, dark serif font. Below the title, there are three input fields stacked vertically. The first field is labeled "Website", the second "Username", and the third "Password". Each label is positioned to the left of its corresponding white input box. At the bottom of the form, there is a blue button with a white border and the text "Store" centered on it.

Here we have the Store password frame. In this frame the user will enter the name of the website in which they want to store the details for then they will enter their username and password they chose to use for the account for the website, after the details have been filled in the user clicks the store button and then the 3 pieces of data will then be encrypted using aes. The code will check which user is logged in by retrieving the username from the logged in table and then using that name retrieved it will check the account table and retrieve the user_key that correlates to the username and then this key will be used to encrypt the data before storing it in the password table along with the username of the account trying to store the password in order for the passwords to only be viewed by that account and no other account.

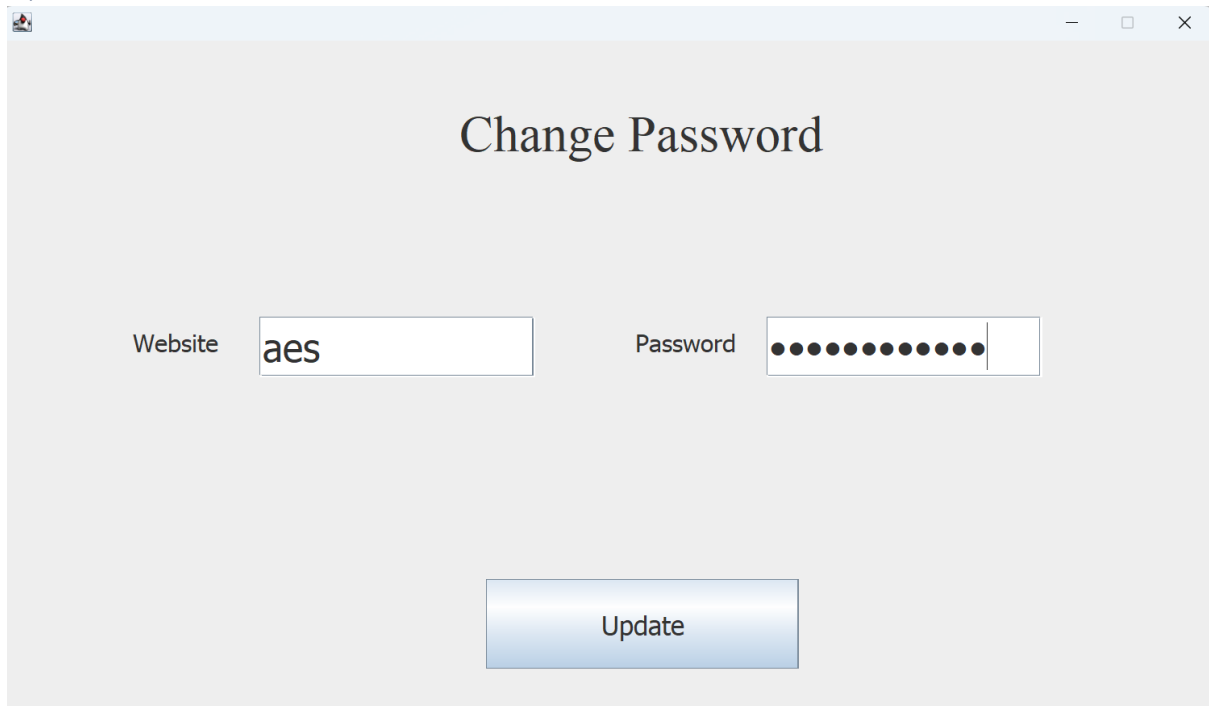
View Passwords



This is how the view passwords frame is designed and at first it wont display any information in case somebody is trying to shoulder surf so the user must click the load data button in order to view the passwords.



After the button is clicked the code will check the username from the logged in table and then retrieve the key from the account table that correlates with that username. After the key is retrieved it will then check the passwords table and find any instance where the username in the logged in table matches an entry in the passwords table under the same username. After finding all the entries it will use the users key to decrypt the website the details are for and then the username and password used for that website and then it will display each of these plaintext pieces of data in the table in the frame under the correct heading, so the decrypted websites will be displayed in the website column the decrypted username will be displayed in the username column and the decrypted passwords will be displayed under the password column.



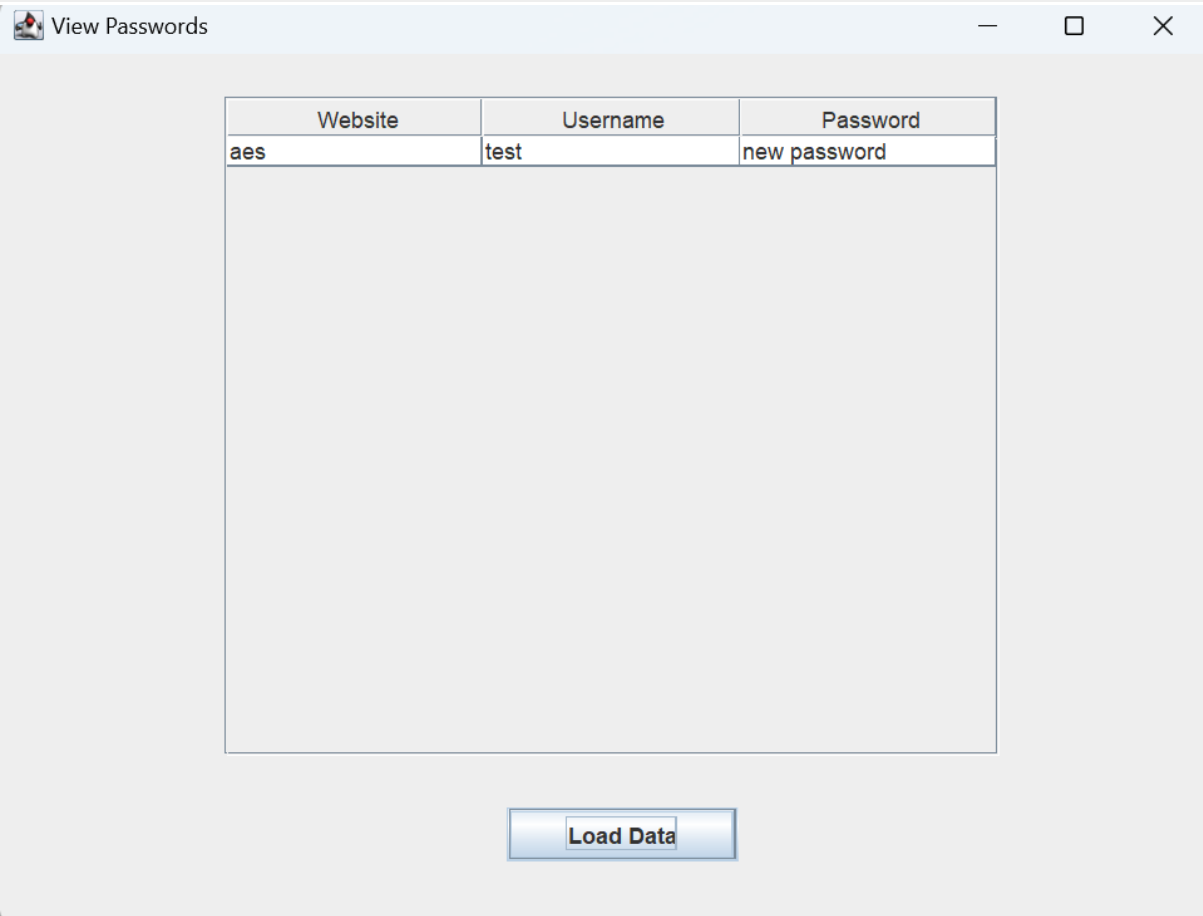
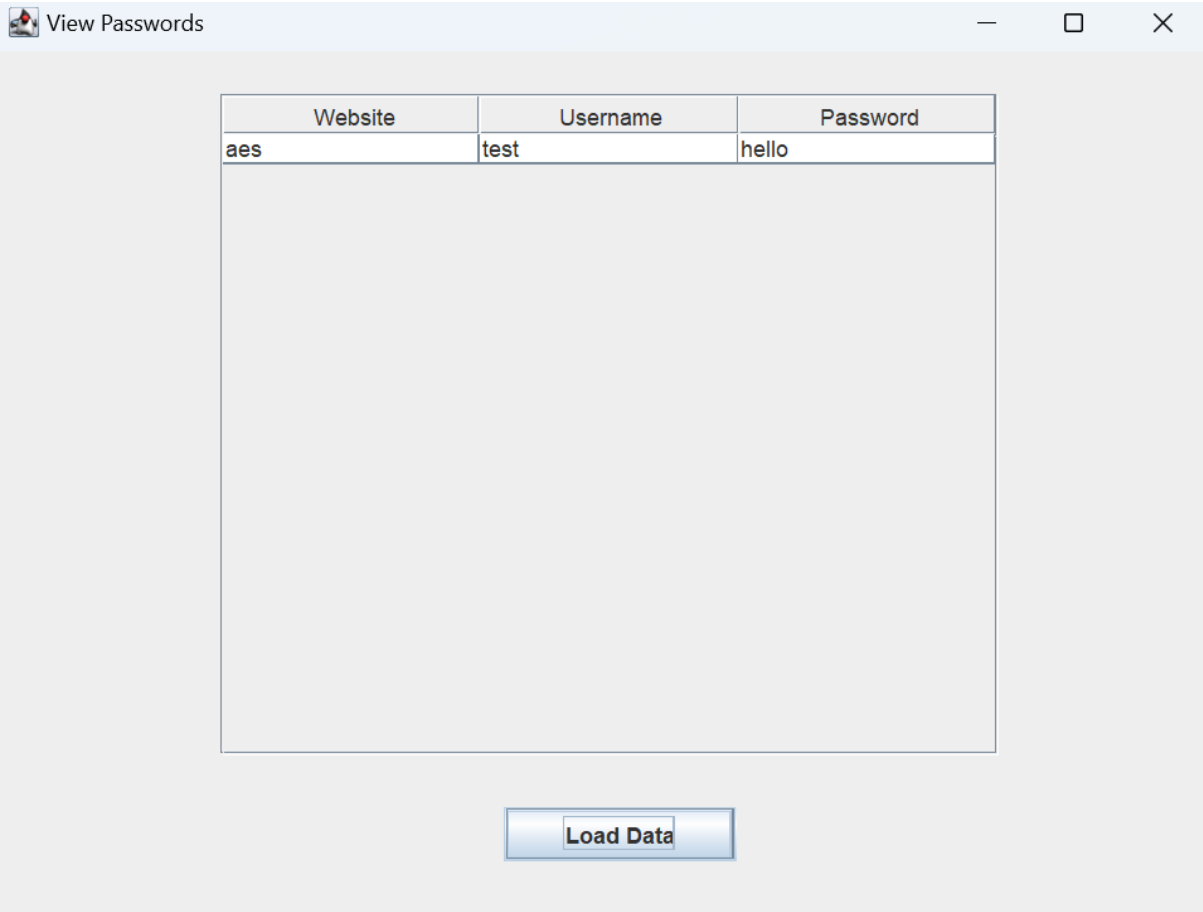
Change Password

Website

Password

Here is the update password frame, in this code it will allow the user to enter the website they wish to change the password for and then they enter the new password and click update, when they click the update button it will again check which user is logged in by checking the logged in table and then using this username it will retrieve the key from the account table from that username, it will then encrypt these entries and update the password where the website is aes in the password table, if the website doesn't exist in the password table they will receive a message stating the website isn't stored in the table. After the password is successfully changed the user will be notified of the successful update of the password. When the user revisits the view passwords page it will then display the new password.

Note the following screenshots the first is before the password is updated and the second is after I have updated the password for the aes website:



Training

For the training sections each button will open a frame on the respective topic, within these frames there are 2 buttons to change the page as the frames contain card layouts which in java is used to have multiple pages per frame. On the first part of each frame, it provides information on what the chosen topic is and the most common types of attacks for each topic and then on the second page there is text describing how a person should go about defending themselves from this attack and what to do if they fall victim to the attack. The buttons at the bottom of each page are used to switch between each page allowing users to go forwards and backwards on the frame. The training consists of 5 topics phishing, business email compromise, password security, malware and ransomware. I chose these 5 topics as they are some of the most common types of cyber attacks but also because each of these allow attackers to gain access to users sensitive information including passwords so I wanted to teach the users of the application how to ensure their passwords stay safe from attackers instead of having all the security measures implemented in the app just for the user to leave themselves defenceless and allow an attacker to gain access to their passwords and therefore gain access to the sites they are registered with and have stored the information for within the password manager.

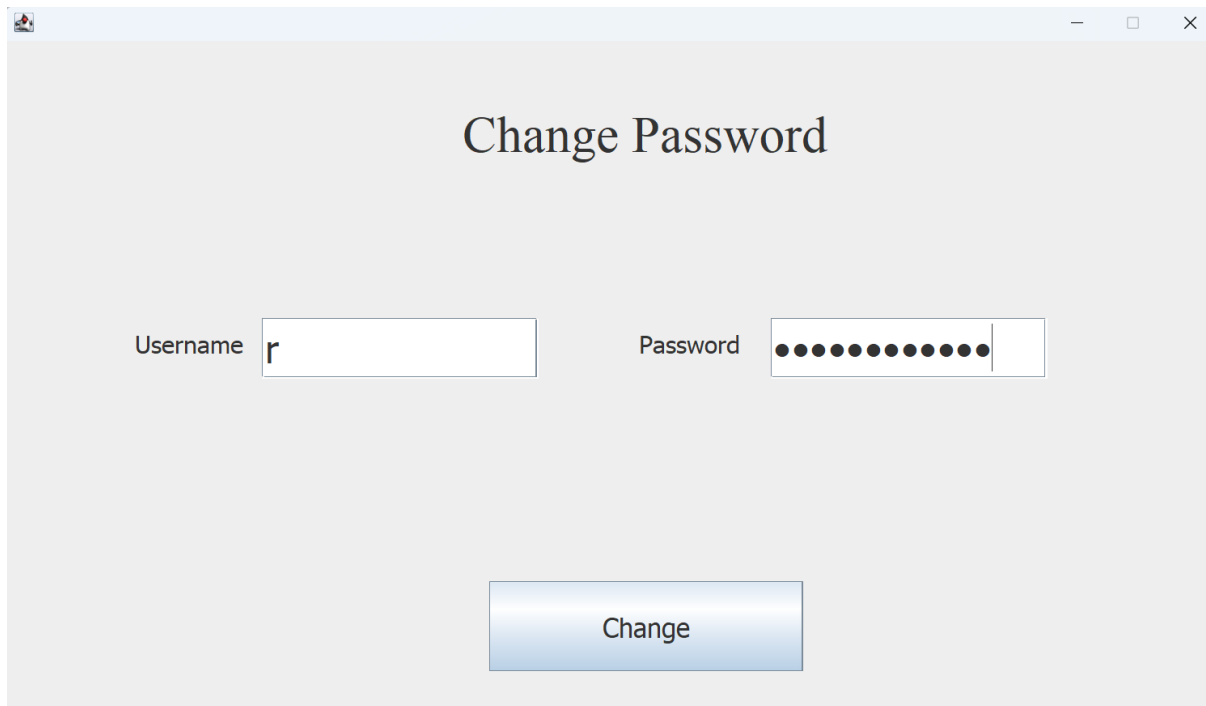
Quizzes

The quizzes are entirely based on what the user learns in the training sections and these quizzes were created using Microsoft forms so that it is easy to update the questions as new attack techniques and defensive techniques are being found constantly so users must constantly be educating themselves on these new pieces of information to stay safe. The quizzes are multiple choice questions, and each question is worth equal marks out of 100 total marks. The buttons for the quizzes will open the Microsoft forms through a google link and the quizzes can be attempted as many times as a user wishes.

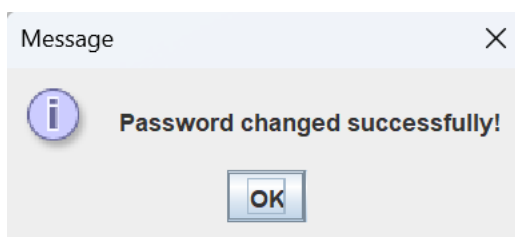
Account

Change password

Like the login and forgot password you must authenticate using mfa in order to be able to change your password. It uses the same mfa technique sending a 6 digit code to the user and comparing what the user enters to the stored code in the mfa table and if they match are they are granted access to the change password page. Also like the register and forgot password pages there are complexity rules implemented into the code so that the password must meet the requirements otherwise the password cannot be changed.

A screenshot of a web browser window displaying a 'Change Password' form. The title 'Change Password' is centered at the top. Below the title, there are two input fields: 'Username' with the letter 'r' entered, and 'Password' which is masked with 12 black dots. A blue 'Change' button is positioned below the input fields. The browser window has standard minimize, maximize, and close buttons in the top right corner.

Here is the frame where you change the password. After authenticating using mfa as for all sensitive information access you must re authenticate so mfa was implemented, when authenticated the user enters their username and their new password and their account will be updated with this new password.



The user will be presented this message upon successful password change and then will be redirected back to the login page to ensure security. After the password change is successful the next time the user tries to login they must use the new password as the old password wont work anymore since its no longer stored in the account table as it was updated with the new password.

Logout

As stated earlier the logout button on the main menu will instantly bring the user to the login page. The user also has no way of getting back to the pages where you are required to have authentication like the main menu without reauthenticating by logging in and verifying the user's identity using the mfa implementations, by having this feature it ensures correct session management as if a user could go back to the change password page straight away after a changed password happened then an attacker could revisit the change password page and then change the users password to whatever they wish and therefore preventing the real user from being able to gain access to the application and the attacker will then have full access to the users stored passwords so that is why they must be logged out instantly with no way back to the application unless they log in and authenticate again.

Description of Conformance to Specification and Design

For the most part my project matches the intended but there are a few things I had to change. Originally I planned on making the quizzes myself using java swing to create the tests but this would've required me to create an admin page in order to change each question and update it in a table on MySQL as I was told by my supervisor not to hardcode the questions in my program and this would've taken a lot more time than I had to finish the rest of the project so I decided to use Microsoft forms as another student was informed to do their quizzes on that and it would have the same outcome with no extra unrequired work as they would be the exact same thing so it would let me focus on more important things like letting each user have their own key for aes and to get the mfa working.

I also had the generator making some passwords that weren't as strong as they could be originally but I changed this to ensure the user would only receive strong passwords from the generator as it would be a security issue if they were using passwords that weren't as strong and it would be tedious as well for the user to keep having to regenerate the password until they received a strong password to use.

For the view passwords and update passwords I originally wanted them to be in the same program so I wanted the user to view the table and then make the content of the table editable so they would change their passwords their and then update the table but I couldn't get this to work as the program couldn't retrieve what was stored in the table on the frame so it would compare empty strings to what was stored in the MySQL table. I decided to try a different approach where I separated the 2 programs because there could be a case of human error where they accidentally change information in the jtable and then would lose the originally password if they can't tell what part got changed accidentally and the new way of having them separated makes it so that this error cannot happen as if they do accidentally mistype the password they want updated they can go back and change it as they are intending to change the password in the first place so they will know what the password needs to be and can fix a misspelling of the password.

I also unfortunately ran out of time to implement salting and peppering into the program. I decided to use what time I had remaining to focus on getting mfa and the random key for each user to use for aes working instead as I felt that it would add more to the security than the salt and hash because with the implementation of the password complexity rules it will already be extremely hard for an attacker to find the original version of the user data and even if they do find the original text the database is stored locally so not only will the attacker need to access the original text through a lot of trial and error that will take a lot of resources to crack they will also need the users physical device, their devices password and the location of the app to access it along with access to the users email to receive the verification mfa code to be granted access and they will also need the users key in order to

decrypt the users stored passwords so I felt that the security would be a lot stronger having mfa and the unique random key for each user than adding a salt and a pepper to the password. If I had more time, I would implement this as it would still make the security as strong as it can be, but the program is still extremely secure without it.

I also had to use coloured buttons for my strength meters as java has no way that I could find to do a meter like in a web dev language where it would be a progress meter so at the start it would be red then the further you get on the meter it turns orange and then finally at the end it turns green instead I had to use coloured buttons which are by default set visible as false and based on the strength the buttons would then be set visible to true to display them to the user as this was the best work around I could think of that would perform the same action as what I needed.

Description of Learning

Technical Learning

Throughout my time working on this project, I have learned a lot of new things both technically and personally. For Technical learning I have become better at understanding and writing java code which I struggled with throughout the years and am now confident in my abilities. In terms of what I've learned for java from this project I have learned how to create and implement slide bars, Multi Factor Authentication by using MySQL and sending emails through java and using card layouts all of which I hadn't seen before in java from college classes throughout the year. I also greatly improved my knowledge and abilities of using MySQL which I have also struggled with throughout the years for projects and classes, I feel like I have full knowledge know on how to use MySQL to its full extent.

Personal Learning

This project has helped me learn a vast amount of new things like how peppering is used in order to not just add security to a database and data but it also allows you to instantly figure out which database or table has been breached as the pepper must be the same for everyone in the database and only be unique for each database as a whole so if there's a breach you can check which pepper was found in the breach and instantly know what database was breached. Then after identifying this incident handling can be performed and when the reason for the breach has been found it can be fixed on all databases before another breach occurs. I have also learned more on topics that have been included in the training and quizzes sections of the program due to researching the topics to find enough thorough information to add to the program and create quizzes out of.

Review of Project

What Went Right and What Went Wrong

Again, for my project almost everything went right. I ran into a few issues throughout the project like for multi factor authentication originally, I couldn't get the emails to send because Gmail used to have an option in the settings called "no third party apps" which you would have to turn off in order to allow third party apps to send emails using your Gmail account but for security reasons they removed this option making it impossible to use Gmail to send emails in applications. I then tried to use my I.T. Carlow student email but found out I would need the port number that the college uses for their emails, and I would also have to display my username and real password in the code in order to use the program and to let it connect to the account. In order to fix this, I had to use a brand new outlook account and find the outlook port number for the smtp which was easy to find I just had to google it and then it worked. Another issue with mfa was that I couldn't get the code to work by retrieving the email from the account that was logged in automatically as the email was stored as a hashed variable to ensure it was secure so I had to change the code so that the user would enter their email and then the code would know which recipient to send the email to. And finally the last problem I encountered with mfa was that if you send too many mfa codes to 1 recipient I would get an error saying Gmail was considering the email sender to be a spam sender and prevent the emails from being received for a certain period of time unless they were paid to use the services so it took a cooldown period before I could send another email to that recipient, This took about 40 emails before I got this error but it was during testing of the sending of emails so I had sent around 15 emails within 20-30 minutes at a guess which unless the program is being tested will never occur in regular use of the application for a user as they would have to login then logout and log back in over and over in quick succession in order to have this problem which again should never happen unless somebody is testing the program which I was. When this problem occurred, I made another account for the emails to be sent from so I could keep testing while I let my original email complete the sending timeout it was given.

Another problem I had was trying to get the aes code to use a different key based on which user was attempting to encrypt/decrypt their information. Something I didn't realise until I faced it in this project is that in java you can create web pages using java but I didn't use web pages I used java swing Gui, now there was no problem with this as I was able to create all the Gui no problem but what was a problem is that java has no session variables unless you use the code to create a web page. Since I had no access to session variables for this project, I couldn't just track which user was logged into the session that way and therefore I had no way to retrieve the key for which user was logged in as without the session variables this was a lot harder to do. In order to fix this problem, the solution I came up with was to create a new table in the MySql database called loggedIn which would work by storing the username of whichever account logged in on the app into the table. Since the app works locally only 1 person can access the app per device it would only allow 1 entry and just constantly update the table with the newest username of the account logged in, I then had

the code retrieve this username from the logged in table and then find the matching username in the account table, when it found the username in the account table it then retrieved the key and stored it as a variable to be used for either encryption or decryption whichever was required. Originally for testing I had the key hardcoded which is a huge security risk but I designed my store, view, and update password code to use this aes code with the hardcoded key and when I got the aes code to use the unique key for whichever user was logged in to work it caused errors in these other pieces of code, it took me a day or 2 to fix but it required me to change a small part of the code by adding a connection onto the code for the new variables to encrypt or decrypt the data. Here is a picture of this code.

```
website = encrypt(websites, connection);
site_username = encrypt(site_userNames, connection);
password = encrypt(passwords, connection);
```

To fix the error I had to add the connection to the encrypt method call.

Missing Aspects of the Project

The only aspect of the project that is missing that I consider major is that I haven't implemented a salt or pepper to the password as I ran out of time and felt I needed to focus on getting mfa and the aes code to work properly as it would provide a lot more security as the complexity rules on the password and the fact everything is locally stored makes the application incredibly secure already. If the application was remote based, it would need to have the salt and pepper.

I also couldn't get the application to be hosted remotely as I sent an email trying to get a blacknight account from I.T. services but by the time I found out I could try and get this account it was the day before the easter break so when I heard nothing back, I assumed that I.T. services had already gone on break.

What I Would Do Differently

The main thing I would do differently in this project if I had to start it again would be:

I would do the Project in a different language preferably a web development language like JavaScript. If I had done this project in a different language, it would have been easier as the use of session variables, xampp for MySQL and even PHPMyAdmin would have helped save a lot of time and allowed me to focus on different parts of the project more. Also, with JavaScript I was able to find a lot more guides and help for problems I was stuck on like the slide bar as this is a lot easier to find out how to do it in web based languages and it looks nicer as well, I could've also made an extension for the program for google as I found this is actually a lot easier to do than I thought. The main reason I chose java was because I wanted to get better at java, and it was a lot easier to fully secure the program than web based languages as you don't have to worry about attacks like cross site scripting or cross site

request forgery and session management wouldn't be as big of a problem with java as there's no back buttons like on a website but throughout 4th year in secure app development from the labs and project in that class you learn how to secure these vulnerabilities so its nothing to worry about but at the start of the year I was worried I would make an insanely insecure program so I decided to use java to ensure security but I would use a web based language either php or JavaScript if I had to restart this project.

Other than a language change for the project I felt everything worked out nicely I had achieved almost everything I wanted to do. The only other thing I would do is at the start of the project find out what services you can acquire from I.T. services to aid the project as I was only informed of blacknight from a fellow student who learned about it at roughly the same time so it would have been very useful to have known about it from the start.

Advice for Next Years Students

If your good at coding maybe try python for this projects because you can add a lot of nice features like buttons that will copy the password and also open whatever website it is for and then it pastes the users details into the site, If you aren't that good at programming then try JavaScript as there's a lot of guides online that I would've liked to have had for java but unfortunately they weren't available for java and made things a lot harder or do it in php as with secure app development you will learn a lot that can be implemented into this project and for the project for that class will be a lot easier if you get more practice with php and you can add to the documents how you used stuff you learned in 4th year as this is seen as a big pro for projects.

Review of Technology Choices

As far as my technology choices I feel as though I made the correct decisions as they weren't big decisions the only technology I really used was MySQL to store my data and then there was bouncy castle to use for my encryption algorithms and hashing, I felt like these were the right decisions as they are the same technologies we have used in previous classes like secure app development and cryptography and they provide full functionality to the project. The only technology change I might make is to use a different server than MySQL as I would like to have the project be able to work remotely but it would require changes to parts of the project like language as it would be hard to perform proper session management techniques using java for a remotely accessible application.

In terms of aes and sha so far today sha 256 hasn't been broken so I believe that is the best algorithm to use until that is cracked, also aes is still considered to be one of the top

encryption algorithms and CBC mode is one of the strongest modes of aes if not the strongest so I believe for those I chose the correct algorithms to ensure security.

Was my Project a Success?

I feel as though my project was a success as it does its intended purpose of storing passwords securely using aes and incredibly strong algorithm and each user has a unique key ensuring higher security standards as hardcoding data is considered to be one the top 25 biggest vulnerabilities and on top of that the application is secure from attacks like brute forcing, SQL injection through the use of prepared statements, and it uses mfa to ensure the user is who they say they are. The program also implements good session management techniques and requires re-authentication for sensitive information access or changes. Also, through the use of mfa this enables users to be able to securely change their password if they forgot their password which is another reason as to why I feel this project is a success.

Acknowledgements

I'd like to thank Christopher Staff for supervising this project and allowing me to do the password manager as I was originally assigned a different project. I would also like to thank the other supervisors for giving me advice and insight on the project during demos and presentations during the year which helped improve the overall design of the application.

References

<https://businessplus.ie/tech/cyber-security-irish-businesses/#:~:text=Some%2028%25%20of%20companies%20have,down%20from%2040%25%20in%202021.>