UNIVERSITY OF
KWAZULU-NATAL

The sections carried out by each member are as follows:

Student Name	Student Number	Role
Hamzah Vahed	222003168	Coder/ Developer
Ismaeel Vawda	222003483	Coder/ Developer
Hakan Yaylaci	222084167	Coder/ Developer

Date: 29 / 04 / 2024

WEB VAULT

COMPUTER METHODS 3

PROJECT REPORT

1– Introduction

With technology being used in today's digital age, managing and protecting passwords is paramount. The group were tasked to develop a password manager application and as a result have produced an effective password manager, which provides the user with the ability to safeguard their passwords in one location.

The application also keeps the users password information safe by encrypting their information making it secure and effective for modern day needs.

2. Functional Requirements:

User Registration	Users should be able to register with a unique username, password, email, and phone number. Registration should validate input data, including username availability, password strength, email format, and phone number format. Upon successful registration, users should receive a confirmation message.
User Login	Registered users should be able to log in using their username and password. The system should verify the entered credentials against the stored user data. Upon successful login, users should be granted access to the password management functionality.
Password Management	Logged-in users should be able to add, view, edit, and delete passwords for various websites or applications. Passwords should be stored securely using encryption techniques. Users should be able to search for specific passwords based on website or username.
Data Protection	The system should implement measures to protect stored passwords from unauthorized access. Passwords should be encrypted.

Non-Functional Requirements:

Security	The system should use strong encryption algorithms to protect stored passwords. Access to sensitive user data should be restricted to authorized users only. The system should implement secure authentication mechanisms, such as username/password
Performance	The system should be responsive, even with a large number of stored passwords and users. Operations such as login, password retrieval, and data encryption/decryption should execute efficiently.
Usability	The user interface should easy to navigate, even for non-technical users. Error messages and prompts should be clear and informative, aiding users in understanding and resolving issues.
Scalability	The system should be designed to accommodate future growth in the number of users and stored passwords. Scalability considerations should be taken into account for database storage, application servers, and network infrastructure.

3- Technologies Used

- Programming Language: C++
- GUI Library: Qt for creating the graphical user interface
- Textfiles for data storage- userdata.txt and passwords.txt

4 – Extra Features

- Dynamic Search bar capabilities
- Changing the user password, adding a layer of professionalism
- Special GUI elements such as custom toggle button
- Multi-Platform interface – MacOS / Windows
- Multi user application

5 – Development Process

We chose to follow an iterative development process, resembling aspects of Agile methodology, for several reasons. Firstly, Agile methodologies emphasize collaboration and flexibility, which aligns well with our team's dynamic and collaborative approach to development.

Our project's user-centric nature made Agile particularly well-suited to our development process. By delivering incremental features and seeking feedback at each iteration, we ensured that the final product met then needs effectively.

Project timeline
Task 1 Version 1-Hard coded GUI
Task 2 Version 1.1Bug fixes and encryption
Task 3 Version 2-GUI designed
Task 4 Version 2.1-Coding fixing and additional features
Task 5 Testing 1-GUI updating and bugs
Task 6 Version 2.2-GUI update and debugging
Task 7 Version 2.3-Search Bar errors and debugging
Task 8 Version 2.4-Login page and validation debugging
Task 9 Testing 2-Revision of lifecycle and update
Task 10 Version 3-update

Top-level design

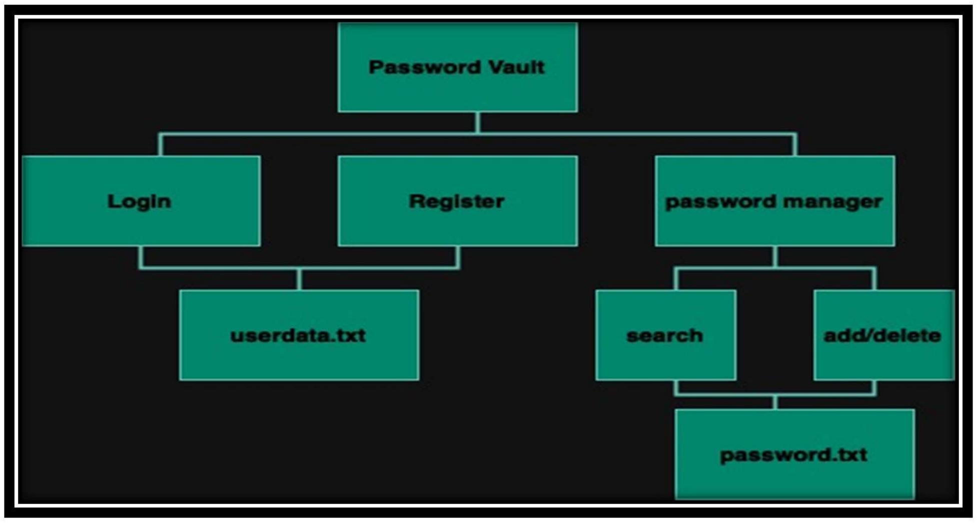


Figure 1 Functional Block Diagram of Application

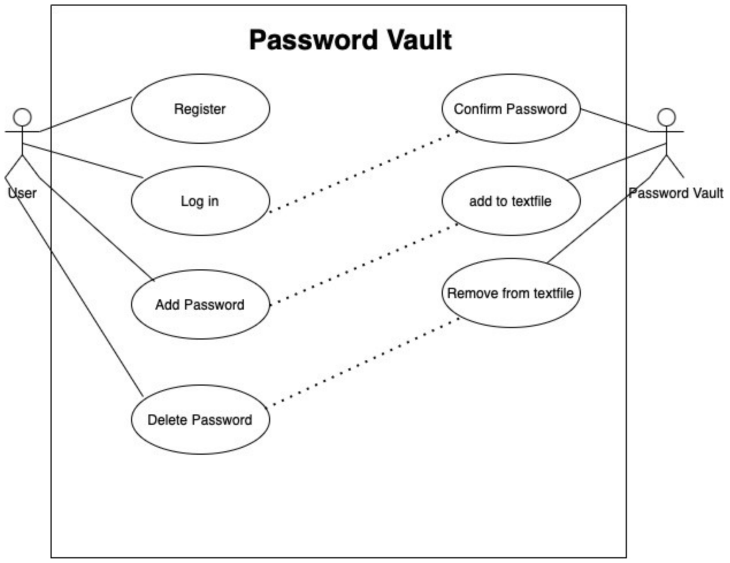


Figure 2 Use Case Diagram of Application

6- Coding development

GUI Design (Hamzah Vahed)

The GUI of the application features a simple yet clean and professional appeal to our application. Stacked Widgets were utilized to display different states of the application, enabling seamless transitions.

Toggle Button allows users to switch between login and registration forms, enhancing interactivity and convenience. Slight curves to the panels and buttons were used in the GUI, giving it a friendlier approach. The curved appeal makes it easier on the eye and thus brings a modern design.

Specially selected colours, such as blue and grey, to convey a sense of innovation. Certain colours strategically used to draw attention and enhance user engagement. Minimalist design was used to get rid of clutter and provide an overall polished user experience, with fewer tabs for easier navigation and accessibility. These design choices were made to create an interactive, enjoyable, and user-friendly experience for the application's users.

Login and Register (Hakan Yaylaci)

At the start of the program, the user is prompted to either register as a new user or login as a previous user with the credentials of which they registered before. For a new user registering, the user is prompted to enter 4 parts of values. Their username, password, email address and phone number of which are stored in a secure Text File. The username and password are to be used for the user’s login after they register or when they need to login into the application again.

After the user has correctly inputted their data, a message box is displayed informing the user that they have registered therefore they can proceed to login. The data entered by the user is stored in a text file named “userdata.txt”, the data is stored in a way that. All data is stored in 1 line of which each data member like username, password, email, and number are separated by a ":" allowing easier access of each data member when adding, deleting, or editing the values.

A class called "EncryptionHelper" was used to efficiently encrypt and decrypt the password entered by the user during registration to ensure a level of security.

Once the user has finally registered and is now onto the login page, they have 2 fields to fill out. Their username and password, of which they previously created in the registration page. After the user enters their data, the application then searches through the text file and compares it to each username stored. Once it finds a match, it then encrypts the password entered by the user and compares it to the password inside the text file to check if it matches. If so, the user is granted access to their profile, else the user is prompted with an error message.

The technique used to read, compare, and mutate values regarding the username and password within the QT programming language was a QMap. It functions as a class template in QT that provides a dictionary-like collection of key-value pairs. This made it easier to verify if usernames and their respective passwords and the user's inputs were matching or not.

Errors faced by this technique were that during the earlier stages of the coding process, registering only required 2 fields, being password and username. This series of errors were solved using string handling by separating the data manually and storing only the password inside the QMap.

Password Manager (Ismaeel Vawda)

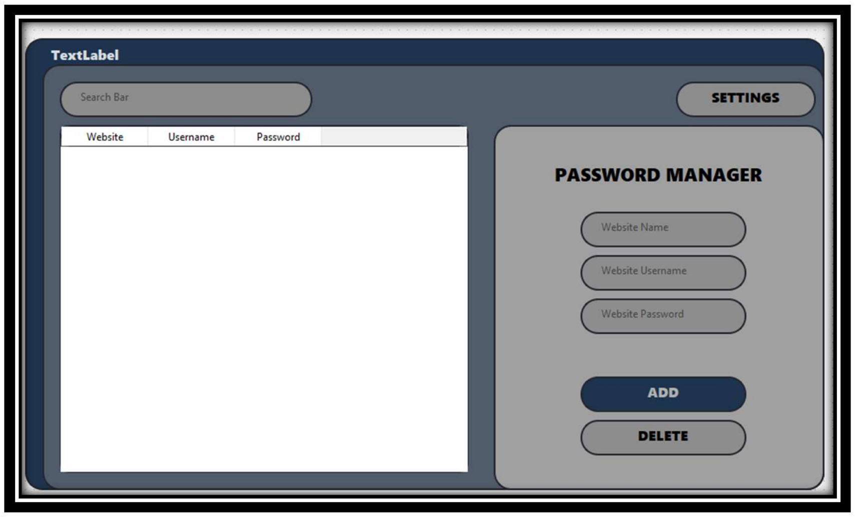
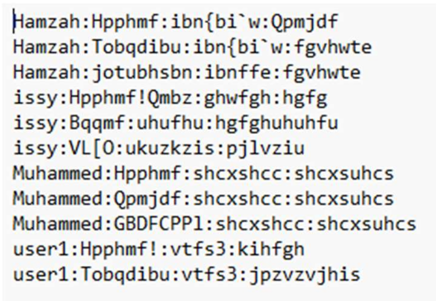


Figure 3 Password Manager interface



```
Hamzah:Hpphmf:ibn{bi`w:Qpmjdf
Hamzah:Tobqdibu:ibn{bi`w:fgvhwte
Hamzah:jotubhsbn:ibnffe:fgvhwte
issy:Hpphmf!Qmbz:ghwfgh:hgfg
issy:Bqqmf:uhufhu:hgfghuhuhfu
issy:VL[O:ukuzkzis:pjlvziu
Muhammed:Hpphmf:shcxshcc:shcxsuhcs
Muhammed:Qpmjdf:shcxshcc:shcxsuhcs
Muhammed:GBDFCPP1:shcxshcc:shcxsuhcs
user1:Hpphmf!:vtfs3:kihfgh
user1:Tobqdibu:vtfs3:jpzvzvjhis
```

Figure 2 Textfile with encrypted passwords

Storing of Data: The password manager UI allows the user to enter passwords, usernames and websites/ applications of their choice. Upon adding the necessary information, the information is stored in a text file. Thereafter the encryption function encrypts the website name, website user name and passwords placed in the file. This ensures that the user's information and privacy is kept safe at all times without exposing their information. Figure 4 shows the passwords being encrypted.

Adding and deleting: The user is free to add and delete up to 1000 passwords for websites of their liking. If they no longer require a password or need to change it they can simply replace it by deleting their old password and adding a new password to be stored. The Inform

Searching: If the user has entered many passwords then scrolling through all the data is not ideal. Hence why a search bar was added as seen in figure 3 above to help the user navigate their information they require efficiently. As a result this makes the UI more user friendly.

The feature which we have added to the program is that if the user adds a duplicate password it will not allow them to and an error message will appear as a result of this.

Uservalidation.cpp and encryptionhelper.cpp (Hamzah Vahed)

UserValidation Class:

The “UserValidation” class is responsible for validating user input data during the registration process. It ensures that the data provided by users meets certain criteria before allowing them to create an account in the password vault system.

This function takes four parameters: username, password, email, and number, representing the user's input data.

- Username Validation: It checks if the username is empty. If so, it displays a warning message indicating that the username cannot be empty.
- Password Validation: It checks if the password is empty and if it contains spaces. If either condition is true, it displays a warning message.
- Email Validation: It checks if the email contains the "@" symbol, indicating a valid email format. If not, it displays a warning message.
- Number Validation: It checks if the phone number has a length of 10 digits and if it can be converted to an integer. If not, it displays a warning message.

If all validation checks pass, the function returns true, registration can proceed.

The “UserValidation” class is called during the registration process to validate user input data before allowing registration to proceed. Specifically, the validateUserData function of the “UserValidation” class is called when the user attempts to register a new account.

EncryptionHelper Class

The EncryptionHelper class provides basic encryption and decryption functionalities using a simple substitution cipher.

Encrypt Function: This function takes a text parameter representing the data to be encrypted. It loops through each character of the input text and applies a simple encryption algorithm. The encryption algorithm involves shifting each character's ASCII value by 1, effectively "encrypting" the text. The encrypted text is then returned as the output.

Decrypt Function: This function takes a text parameter representing the encrypted data to be decrypted. Similar to the encryption process, it goes through each character of the input text and applies the reverse of the encryption algorithm. The decryption algorithm involves shifting each character's ASCII value back by 1, effectively "decrypting" the text. The decrypted text is then returned as the output.

The “EncryptionHelper” class is called whenever encryption or decryption of data is required. Encryption is performed when storing sensitive data, such as passwords, in the password vault. Decryption is performed when retrieving encrypted data from storage.

Additional Feature: We have added a change password function that allows an existing user to change their password to a new one. The function goes through the textfile to the users details and edits the password. This gives the application a polished user experience. We have added validation to ensure that the user cannot use the current password as the new.

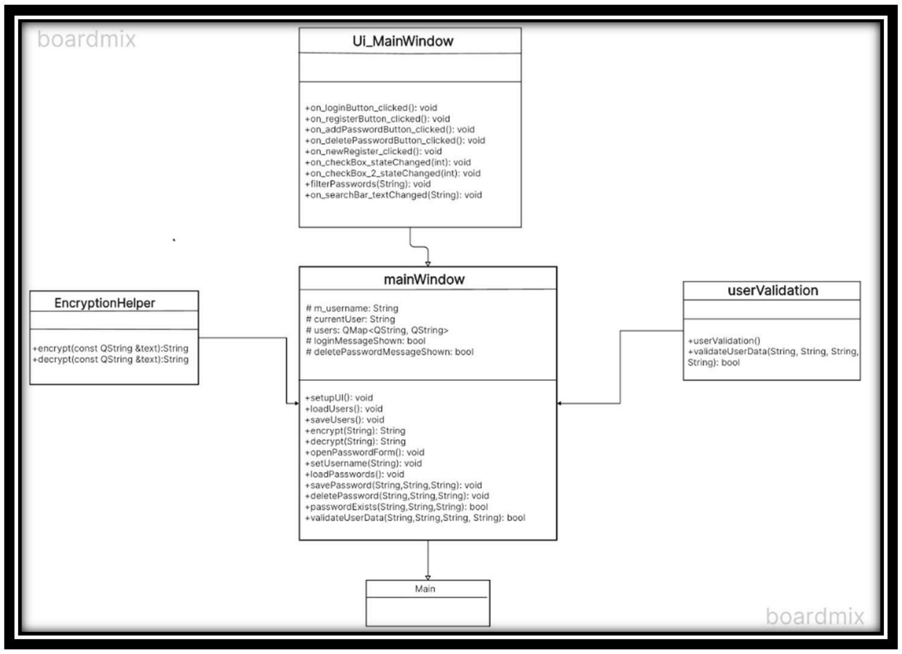

boardmix
Ui_MainWindow
+on_loginButton_clicked(): void
+on_registerButton_clicked(): void
+on_addPasswordButton_clicked(): void
+on_deletePasswordButton_clicked(): void
+on_newRegister_clicked(): void
+on_checkBox_stateChanged(int): void
+on_checkBox_2_stateChanged(int): void
+filterPasswords(String): void
+on_searchBar_textChanged(String): void
EncryptionHelper
+encrypt(const QString &text):String
+decrypt(const QString &text):String
mainWindow
# m_username: String
# currentUser: String
# users: QMap<QString, QString>
# loginMessageShown: bool
# deletePasswordMessageShown: bool
+setupUI(): void
+loadUsers(): void
+saveUsers(): void
+encrypt(String): String
+decrypt(String): String
+openPasswordForm(): void
+setUsername(String): void
+loadPasswords(): void
+savePassword(String,String,String): void
+deletePassword(String,String,String): void
+passwordExists(String,String,String): bool
+validateUserData(String,String,String, String): bool
userValidation
+userValidation()
+validateUserData(String, String, String, String): bool
Main
boardmix

Figure 5 Class diagram

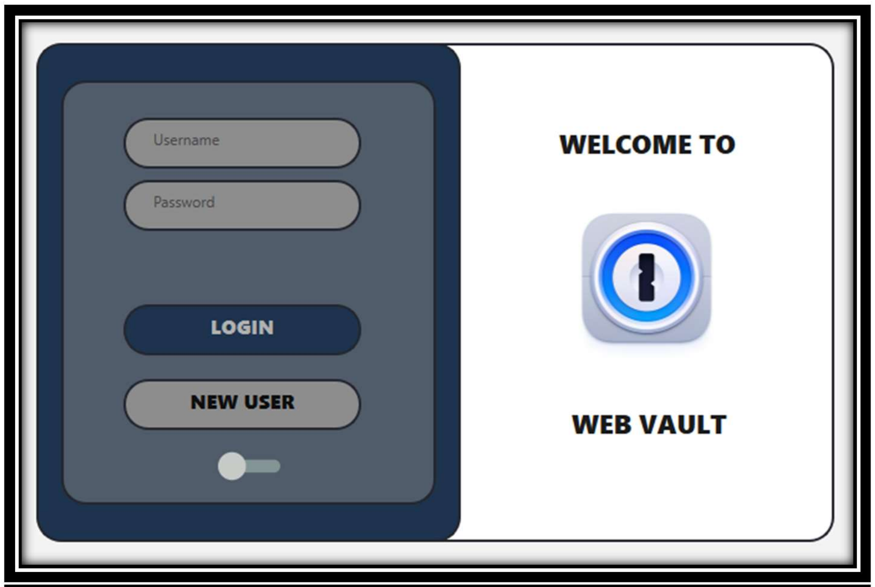
6- Future enhancements

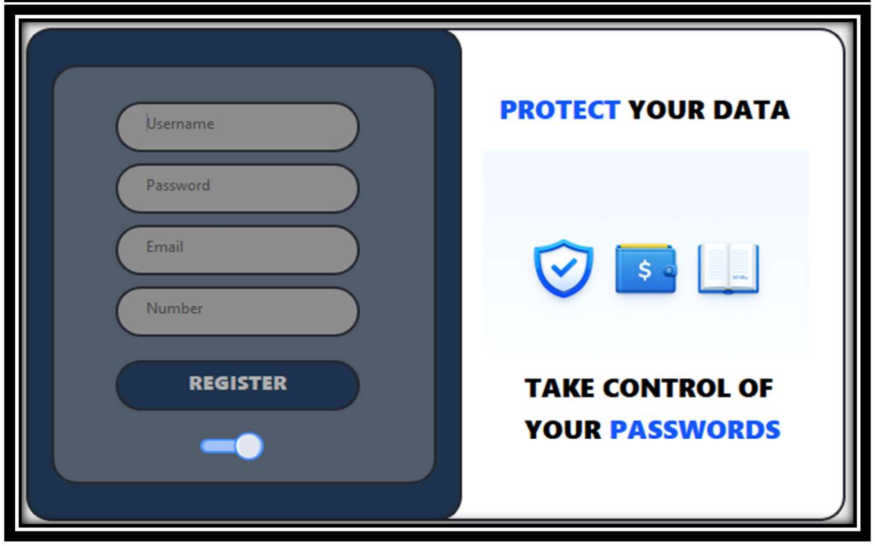
Integration with cloud storage services for backup and synchronization. Enhanced security features such as multi-factor authentication. Improved user interface for a more intuitive user experience.

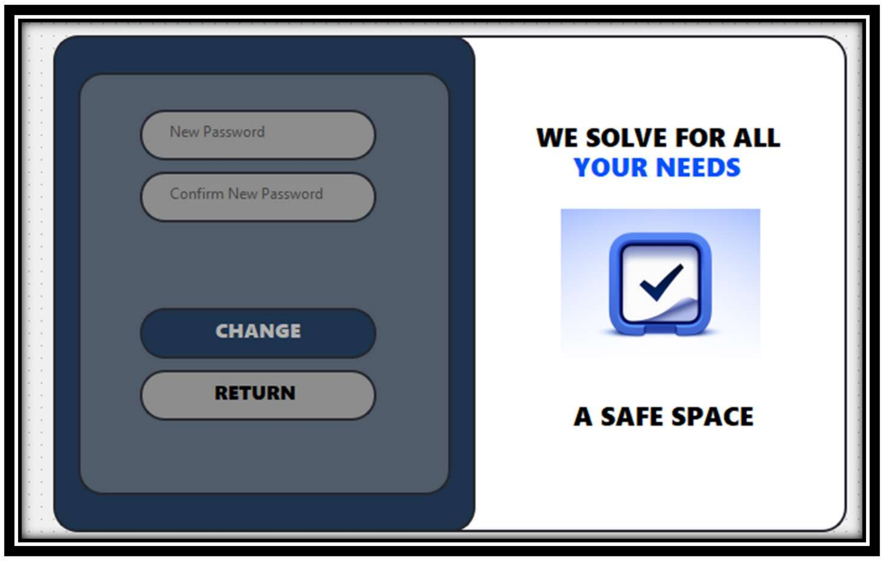
7-Conclusion

In conclusion, the password vault created has allowed the group to become familiar with OOP principles in application development. The group has also learned about GUI’s in C++ which were used to demonstrate the working code. At the end of development the group successfully created the Password Vault application and can take away important lessons from this task.

APPENDIX


Username
Password
LOGIN
NEW USER
WELCOME TO
WEB VAULT


Username
Password
Email
Number
REGISTER
PROTECT YOUR DATA
TAKE CONTROL OF
YOUR PASSWORDS


New Password
Confirm New Password
CHANGE
RETURN
WE SOLVE FOR ALL
YOUR NEEDS
A SAFE SPACE