PASSVAULT PASSWORD MANAGER

PROJECT DOCUMENTATION – OCR A-LEVEL COMPUTER SCIENCE

NAME – ROSHAAN MALIK

Candidate Number – 6681

Centre Number – 32157

Qualification Number – H446

Project Title – PassVault Password Manager

Date Finished – 05/05/23.

Contents

ANALYSIS

This section includes a breakdown of the requirements and planning of the application and how I will fulfil them.

PROBLEM IDENTIFICATION

Nowadays people often record their personal information because they are unable to remember so much. They often write it all in a diary or increasingly, on their smart devices. However, they do not make use of any dedicated applications that can record all their information.

The problem is that most free personal information managers for computers are unsecure. According to security researchers, they are not safe to input sensitive information, like passwords and bank info, because they have weak encryption or a lack of hashing, defeating one of the main reasons for using one. They are not as sophisticated on desktops as the passwords are manually copied and pasted onto login pages. Unlike mobile versions, they do not automatically fill in login screens when detected. The personal information managers that cost money require a monthly subscription after few passwords are saved to them and are too expensive for most people to use.

As a result, there are about two thousand cyber-attacks a year to access people's personal information. The UK had the highest number of cybercrime victims per million internet users at 4783 in 2022 – a 40% increase from 2020 figures, according to the NCSI protocol (Network Communications Service Interface) protocol.

PROGRAM SUITABILITY

Form Factor Suitability

Using a smart device is the most suitable way for storing passwords. It is the most seamless way of editing information in a digital notebook or database. For example, without tediously crossing and replacing information, the programmer can reduce that task into a few buttons or keyboard clicks.

Smart devices help centralise all a user's belongings into one unit. For example, mobile phones can act as a GPS, bank card, video camera and especially a note taker. A separate notepad or paper to store passwords will put them off having to carry around their phone and an additional item with them. So, it is better if passwords are recorded on their smart device.

Code Suitability

The program will store mostly strings (characters), which are fundamental data types, most found in personal information managers and note takers. They store lots of text, which can be represented with string. Most programs also require a particular condition for a certain outcome to occur, like checking if an inputted username and password match an existing one. If they do, the user will be granted access. This account system can be applied to the program, to group personal information to different accounts.

I will use a computational approach to this problem to create a desktop application. There are several ways of computational thinking that can be applied to developing a password manager: abstraction, for simplifying the password manager to only its most essential features; decomposition, for breaking down the functional requirements into smaller, more manageable problems and computation, because when encrypting or hashing passwords, the program needs to run a series of calculations to convert it to a string of characters.

QUESTIONNAIRE FINDINGS

I chose to do a questionnaire so I can further investigate the issues with personal information managers and how to solve them. Most questions will have either 'yes' or 'no' as answers to quickly process information, however, few will ask for an elaboration to further tailor the application to their needs.

They will be printed from a separate document and handed out to various age groups in my family, from tweens to the middle aged. They will be handed to students from all years in my school and a few teachers. This is to see what age range the app suits the most. I will approach them asking them to spare two minutes to answer each question for my A-Level Computer Science coursework.

Next to each answer is the number of times it was selected. Below them are influences they may have on my design.

Ages: 13, 17(x3), 18, 30, 44, 47

1) Do you usually use the same password for all your accounts?

   Always - 0
   Often - 5
   Sometimes - 3
   No – 0

   Since most users reuse the same password, the application will detect this occurrence and ask the user to re-name the password.

2) Would you pay a subscription for a password manager? Please state a reason for your answer.

   Yes – 2 – Mostly because they can afford it, as they have full time positions that make enough money.

No – 6 – Mostly because they were under eighteens who do not have enough money to afford one because they are in education.

Since most students could not afford to pay, I will make the application free, so it will not require a payment screen.

3) List any sensitive information you would store on a personal information manager.

Passwords – 8
Cards/ accounts – 6
Addresses – 5
Phone numbers – 1
General notes – 3

All age groups wrote similar answers; all items were identical, except one thirteen-year-old added phone numbers too. There is no need to create account levels for different age groups.
The most common answer was 'passwords' so it will be a password manager, storing for different types of accounts (Email passwords, bank accounts, Wi-Fi passwords, etc.).

4) Will you remember to record new passwords/ personal information to the application?

Yes – 5

No – 3

I will include a pop-up screen to remind them to do so whenever they create a new account. The app will run in the background and constantly check for a new log-in screen.

5) Do you feel comfortable saving passwords or personal information to web browsers like Google? Please state a reason for your answer.

Yes – 2
No – 6

People said no because they were concerned about Google's lack of security, so I will add a hash algorithm to it when storing it to prevent unauthorised access.

6) Do you already use your smartphone's personal information manager, such as iCloud, Google Passwords, etc.?

   Yes – 5
   No – 3

   A mobile version of PassVault isn't very demanding, so it is not required.

7) Do you use your PC or laptop's password storage services, such as Mac keychains, Chromebook password manager?

   Yes – 2
   No – 6

   A desktop version of PassVault is demanding for people with Windows computers, compared to MacBook's and Chromebooks, so the app will be able to operate on Windows.

8) Would you trust a password manager to store sensitive information on web browsers using cookies?

   Yes – 4
   No – 4

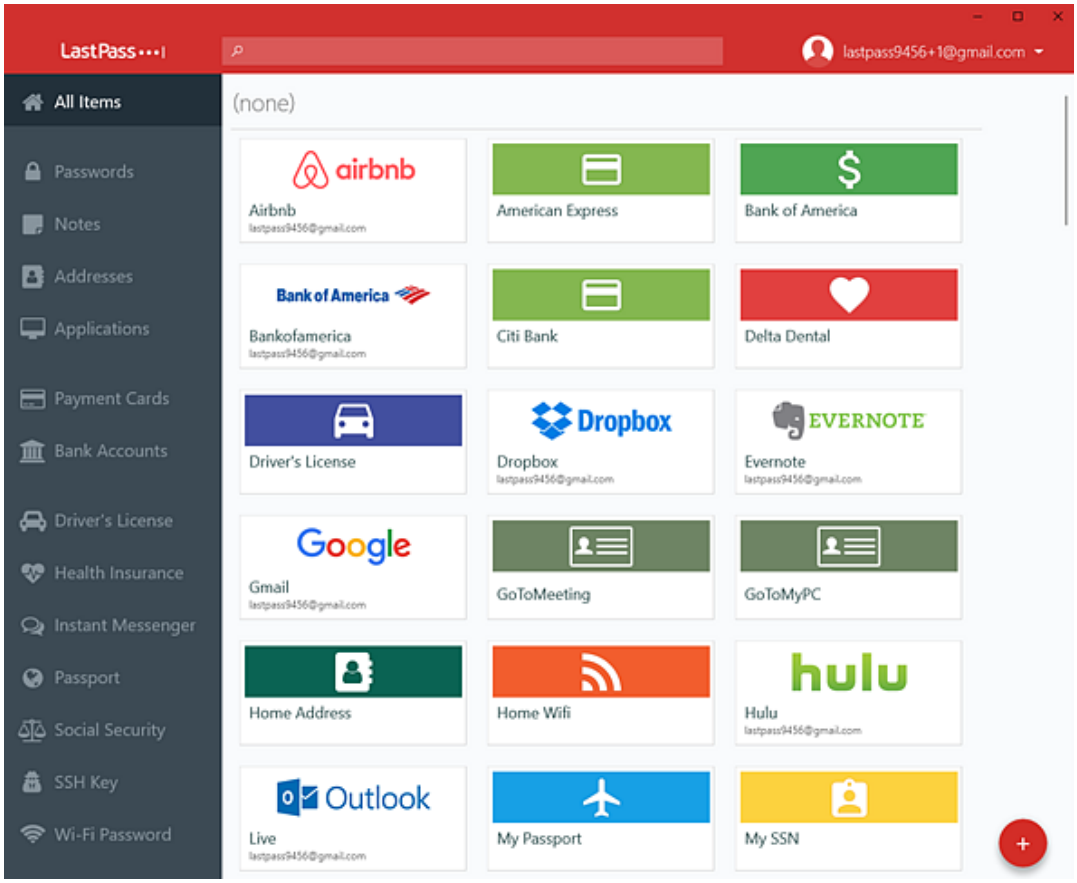
   The database will be stored locally on the user's device, rather than online.

Overall, I have a better understanding of the customer requirements. The app needs to be made for Windows 10 and 11 desktops, as they are the most used types of OS with the weakest security in their built-in personal information manager. The most common type of information people would record are passwords. Therefore, the personal information manager will be a password manager and store mostly passwords. It will however record cards and addresses too since it was frequently answered. It will also store secure notes if the user wants to store anything other than the main types of passwords. The type of passwords stored will be general passwords, bank account passwords, cards, addresses and secure notes. The passwords in each section will be hashed before they are stored.

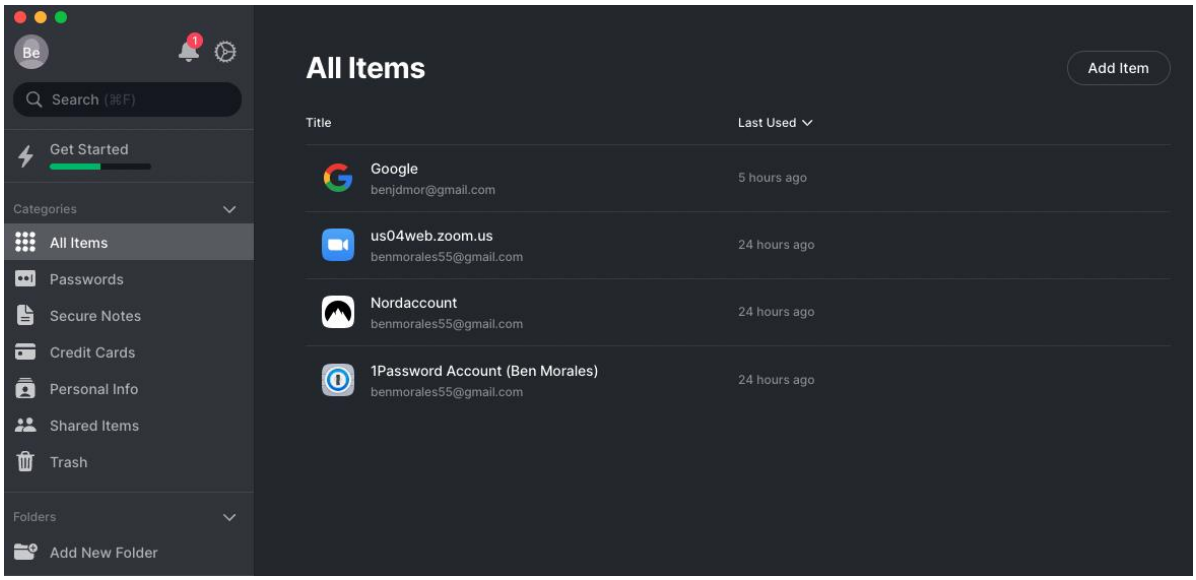
RESEARCH

As well as the questionnaire findings, the existing password manager features are going to be implemented in my application or have similar versions. They must be familiar to customers because of existing personal information managers, such as LastPass and NordPass, so the user can navigate it effortlessly.

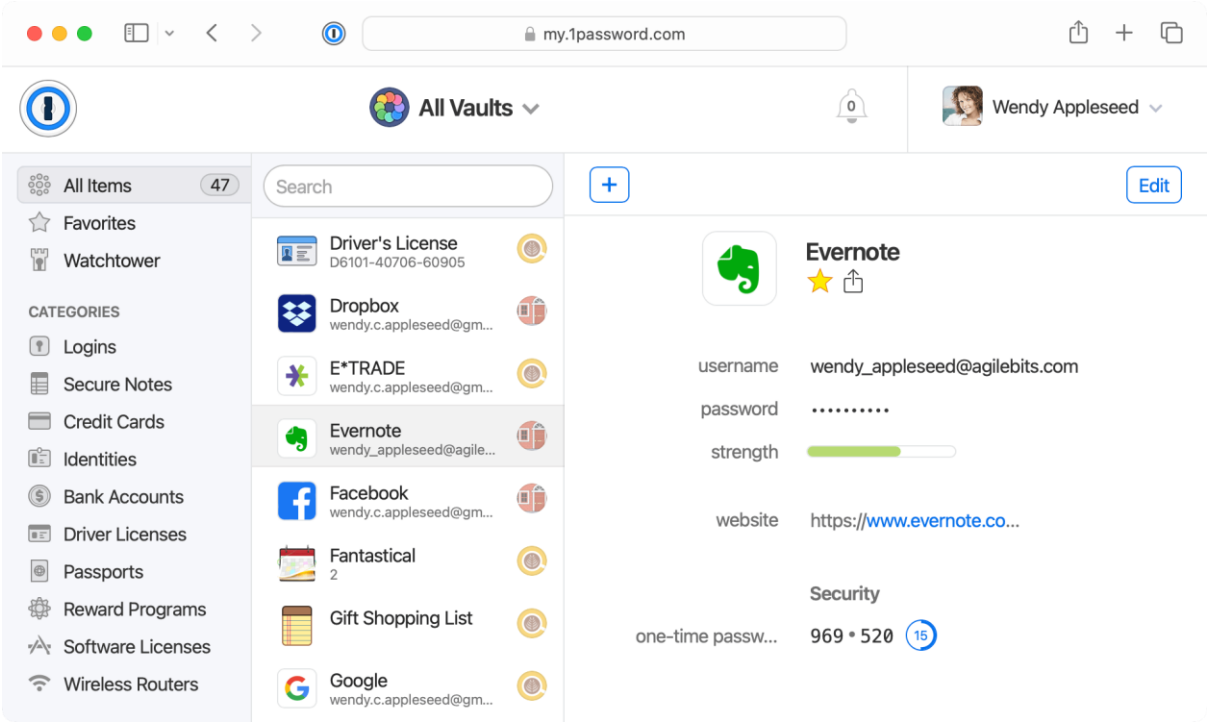
- This will be done by matching the login and vault page to the user's schema of a menu. They will resemble these menus from the most popular existing password managers (LastPass, NordPass and 1Password respectively):



https://support.lastpass.com

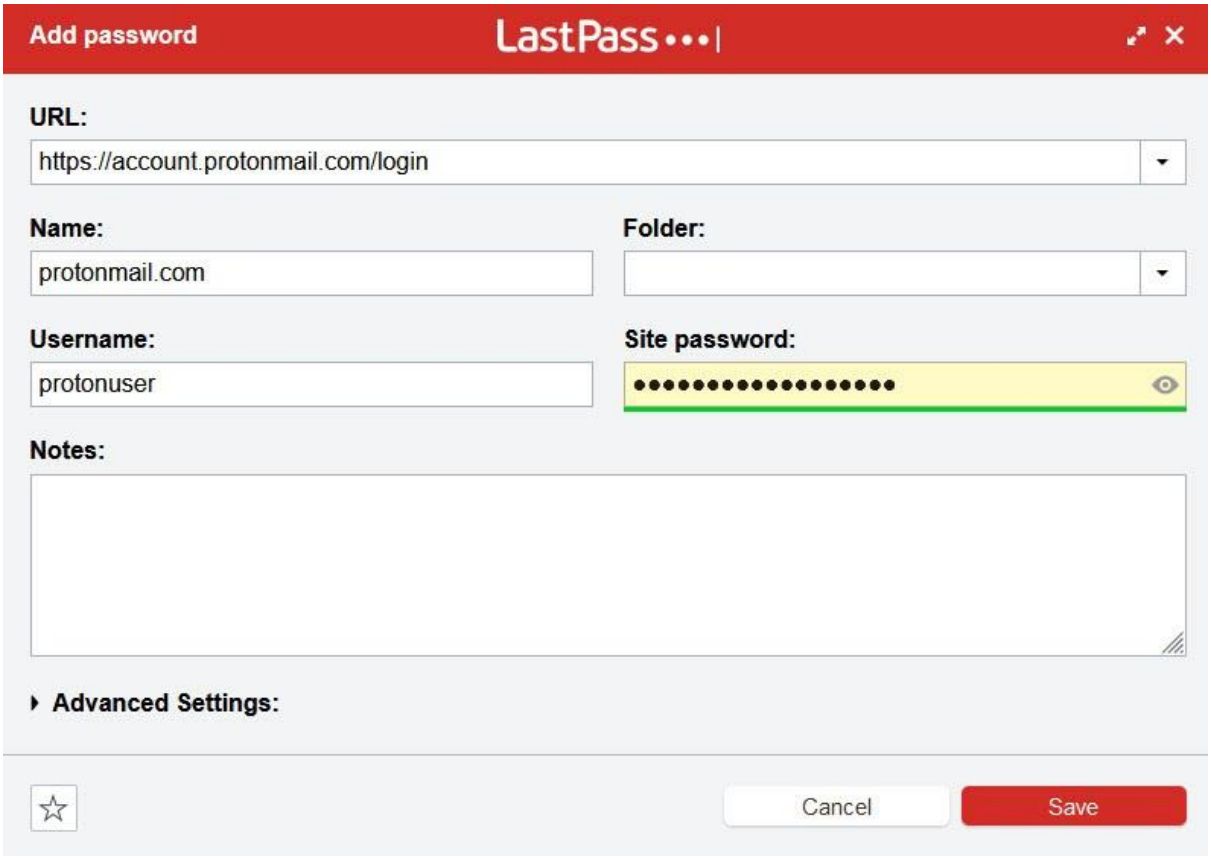


NordPass Reviews from Website



Google Images

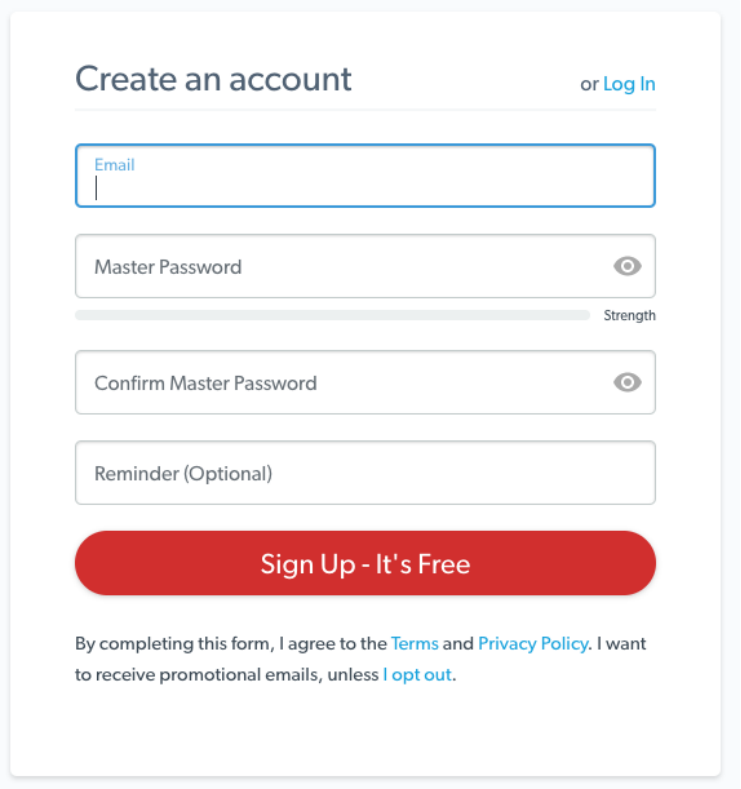
- These menus all include one left panel for the types of passwords, and the other one for the passwords saved there.



Support Section of Proton Website

- The most popular password, LastPass, use this format for saving passwords.
  - Every time the user clicks 'new password', this menu is brought to them.
  - The URL is used to launch the password's login page, saving the user time to load it them self.
  - There is a name box to title the password, so the user can distinguish it easier.
  - There is an additional text box: notes. This is meant for additional information about the account the user wants to save. However, since I have a separate section for general

notes, which most passwords managers do not include, it will only be included for that section.



Sarah Hension Blog

- LastPass's login details are the master username (the email) and the master account.
  - The master username and master password are the only details the user needs to enter, because their other passwords will be saved onto it, hence the term 'master'.

STAKEHOLDERS

Based off my questionnaire findings, I have identified who the stakeholders of my app should be.

The consumers of the PassVault app. The target demographic is young adults or teenagers who are graduates or still in education, because they are the demographic least willing to pay for password managers, so they need a free one the most.

These people will also be who I present the application to as a prototype for my evaluation. They will provide feedback by performing black box testing for the interactivity of the finished prototype.

ABSTRACTION

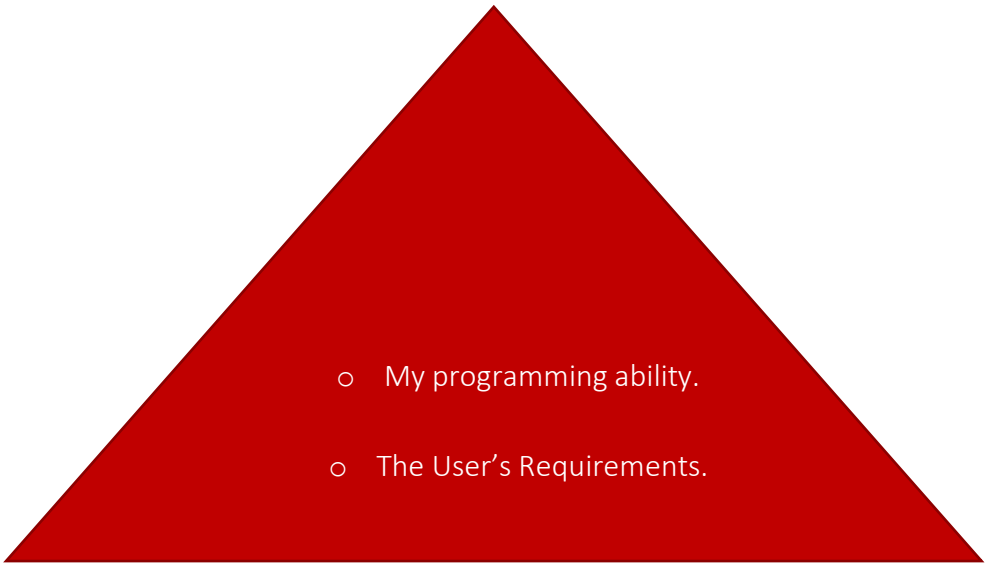
Below is table of all possible features any password manager will have sorted into relevant and irrelevant details. The relevant details will be included in the program as back-end processes or as proof of concept if the function is meant to change the aesthetics. The irrelevant details are not required for me to develop in this coursework; they are normally handled by an art team.

Relevant Details	Irrelevant Details
Login Menu	Menu Layout (Except for Layout Regarding Proof of Concept)
Sign Up Menu	Colours of Menu
Tabs for Password Types	Menu Animations
Data Grid	Loading Screen/ Bar
Input Text Boxes	User Profile Picture
Labels of Tabs and Textboxes	
Command Buttons	

REQUIREMENTS

Functional Requirements

These requirements are meant for the programming the app requirements will resemble what the customers expect the ideal password manager to be like to the best of my programming ability. My approach will use the following model:



The customer's requirements will be considered first if I am able to program them. If I am unable to program the solution, I will reduce the complexity of the function, so I am able to program.

The app will need to meet the following software features:

- Appropriate menus that the user would expect for the password manager to have:
  - The menus will be 16:9, for desktops.
  - One panel will have the types of information stored by the user: Passwords, Bank Accounts, Cards, Addresses, and Secure Notes. Each one of these types will record information in a grid, pre-built by Visual Studios.
  - The other panel will have the associated information in that type shown in a data grid.

- These are the following things the user can input for each category of information. Each one will have their own textbox where the user can input it and have their own column in the data grid:
  - Passwords: Title, Username, Password, Link
  - Bank Accounts: Title, Username, Password, Account Number, Sort Code, Link
  - Cards: Title, Name on Card, Card Number, Card Type, Security Code, Start Date, Finished Date, Link
  - Addresses: Title, Address 1, Address 2, Address 3, City/Town, County, Post Code, Link
  - Secure Notes: Title, Notes
  - The title is meant to be what the user chooses to call that password, address, or note for them to distinguish it better in the data grid.
  - The link is the hyperlink that will be launched by the user so that they can input the login information quicker. They may simply copy and paste it into the corresponding boxes.

- A login and sign-up screen to create a new PassVault account, or log into an existing one.
  - It needs a master username (which will be the user's email) .
  - It needs one master password. The user will only need to remember the master password to access all other passwords on their account.
  - The user creates a master account with the master username and the master password (this is how the password managers mentioned above work).

- Storage space on a database made by Microsoft Access, rather than on the internet.
  - This prevents wider access to hackers on the internet.

- A hashing algorithm for the user's passwords.
  - To prevent unauthorised access to the user's passwords in the database.
  - If the master password is inaccessible, a hacker cannot view the other passwords connected to that master username.

- A random password function if the user wants to produce a new strong password.

User Requirements

These are requirements the user must have for the finished application so it can be run by the user.

- A PC or laptop with the OS Windows 10 or 11.
- The app will run on Windows 10 and Windows 11, because they are the most used Operating systems today.
- The app is very basic, so it does not require much processing power to run. Therefore, virtually any PC or Laptop may run it.

Development Requirements

These are requirements for the development process, such as apps to install, and why computational methods are suitable for them.

- Visual Studio 2017, using the Window's Form App template.
  - This prevents me from programming a window from scratch, saving time.
- C# Programming Language.
  - It is an object-oriented language, so I am not limited to only functions and sequential logic. This means that I do not have to write code in the order it will be run.
- Microsoft Access
  - This is to create a database that the program will store the user's entered information.
- Microsoft Access Database Engine 2010 Redistributable
  - This is additional software so the code can read and write to the data base.

LIMITATIONS

These are things I am unable to meet in the requirements and the reasons for the changes I have made as a result, using the model mentioned before.

- Storage space on a database made by Microsoft Access locally, rather than on the internet.
  - This is also because I am unable to purchase server space to store the passwords on the internet.

- Inoperable on operating systems before Windows 10.
  - I am unable to ensure the app runs well on every version of Windows. The program would have to be altered to every version of Windows.
  - Most users use Windows 10 or 11, and older operating system usage is in decline anyway.

DESIGN

This part will contain the process of making the structure of the code and the menu and database layout.

DESIGN METHODOLOGY

The iterative-agile method will be utilised. Because I will be able to make constant amendments to the requirements and the plan if necessary. The requirements may change due to my programming ability, time constrains, faults in the code and design or user feedback.

DECOMPOSITION

This section will decompose any app requirements into smaller sub-problems, with a description for their solution, explaining the process.

Login screen:

- The user needs to enter text in the username and password boxes.
- The user needs to click login or press enter to sign in.
- The entered username (user's Email) and password need to be checked for in the database.
  - If they match, the account and all its saved passwords are opened.

- If they do not, the program will state that account access is denied.

Sign up screen:

- Needs a textbox to create the username (email), password and confirm the password.
- The user needs to click create account or press enter to create it.
- The account will be added to the database.

Password Generator:

- User needs to input a password length 8 – 16 characters long.
- The password then needs to be displayed once the number is selected.
- Then the user can select to copy the password or generate a new password instead.

Vault menu:

The menu will load two panels: one for password types, one for saved passwords.

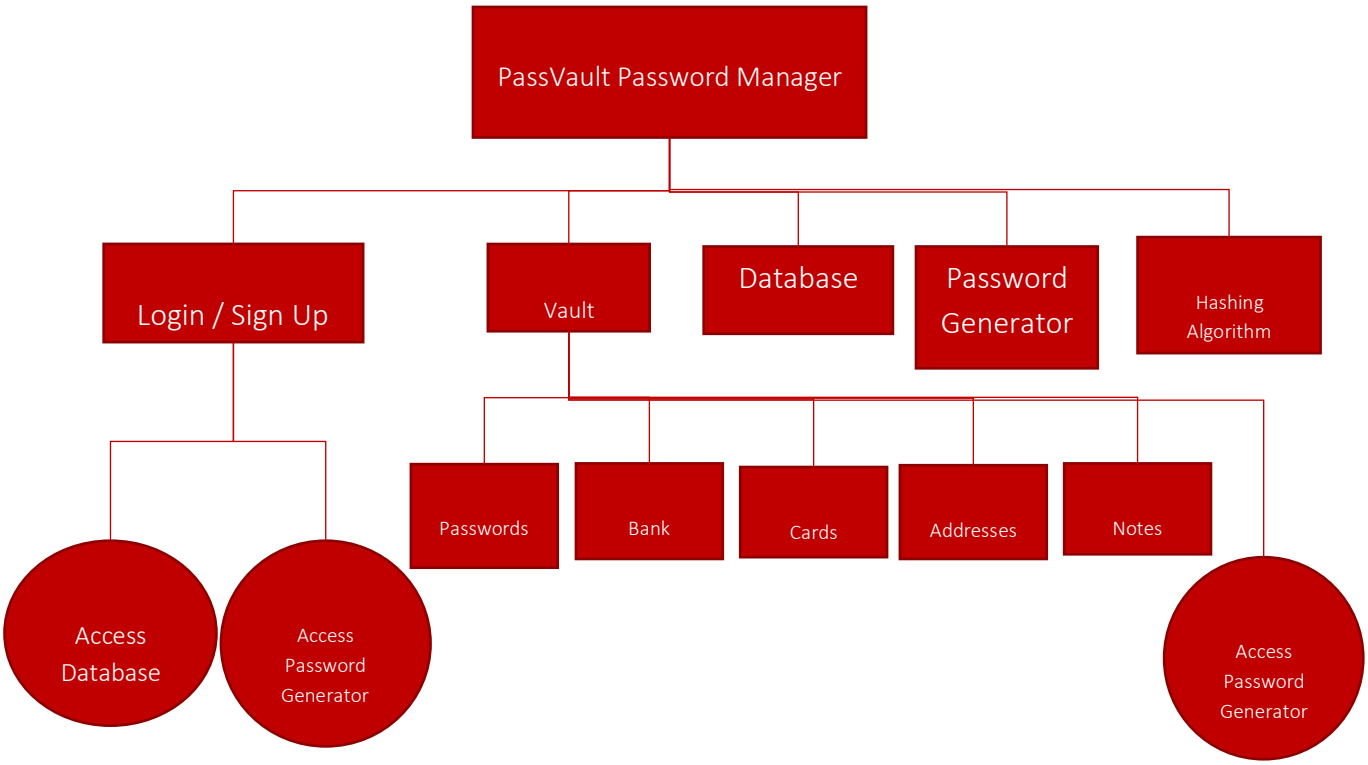
Password Types	Saved Passwords
○ Clicking on the type loads its saved passwords. ○ It must load them from the database to the data grid. ○ It must have an additional tab for generating a random password 8 – 16 characters long. ○ When the user clicks the single floppy disk, it saves the current data grid. ○ When the user clicks the double floppy disk, it saves all data grids.	○ It must display saved passwords in a data grid. ○ It must display textboxes for the user to input information to save new or edit existing passwords. ○ When the user enters information in the textbox, it must be updated in the data grid. ○ When the user enters the same password more than once, they are encouraged to change it.

- When either floppy disk is clicked, the password is hashed before being stored in the database.

- All the sub-problems involving buttons, tabs or textboxes are suitably sized. Visual Studios has them pre-built, so I do not need to program them from scratch; I simply need to drag and drop however many I need.

TOP-DOWN MODEL

Shown below is each menu of the PassVault app and each of their navigation paths from when the app is first launched. There may be more than one way of accessing each menu. For example, the password generator may be accessed when trying to create a master account or creating a new password to be stored.



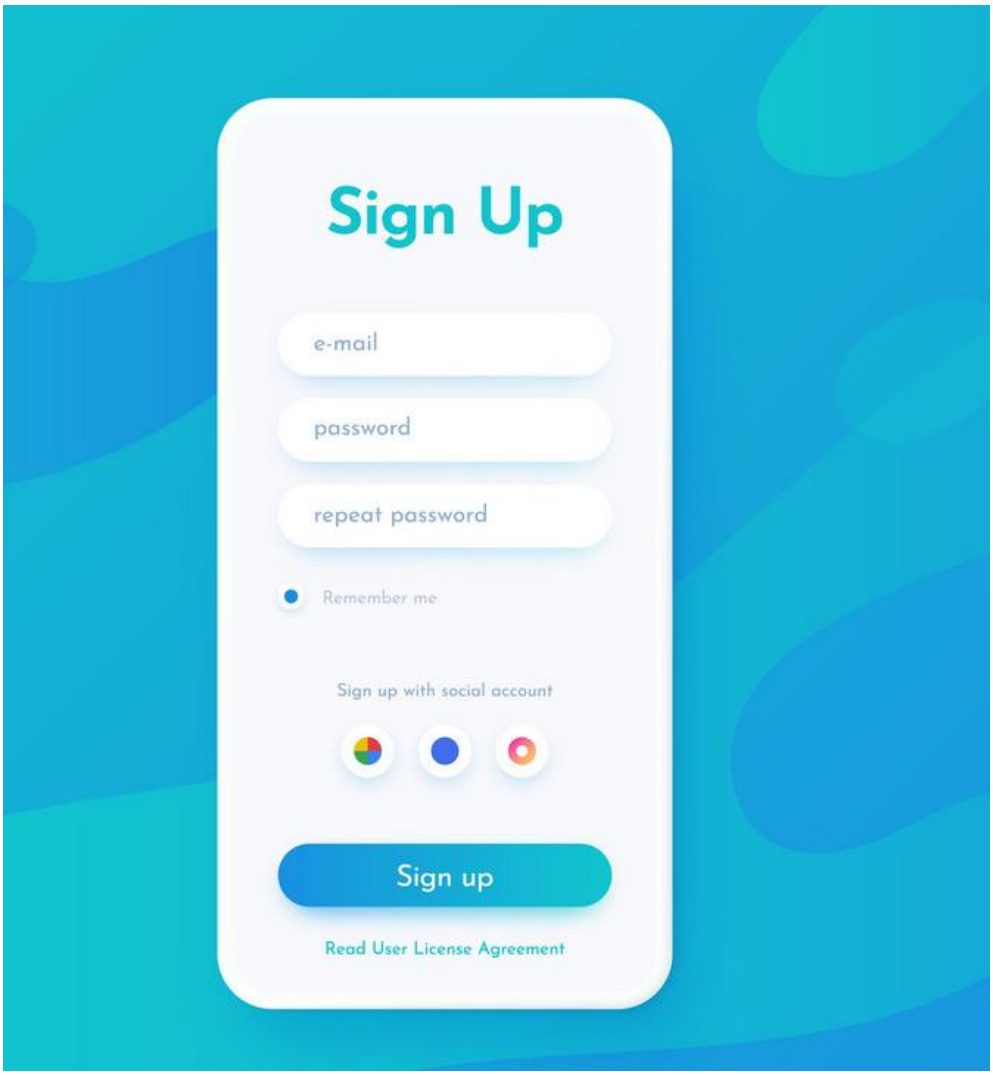
USABILITY

The app will be user-friendly by resembling the same log-in screens of most apps:

Vector Stock Website

- This is a basic log-in layout that every user has encountered, so PassVault will follow it too.

The sign-up menu will follow the same user schema:



Vector Stock Website

- There are two boxes for the new password to be created like most sign-up pages.

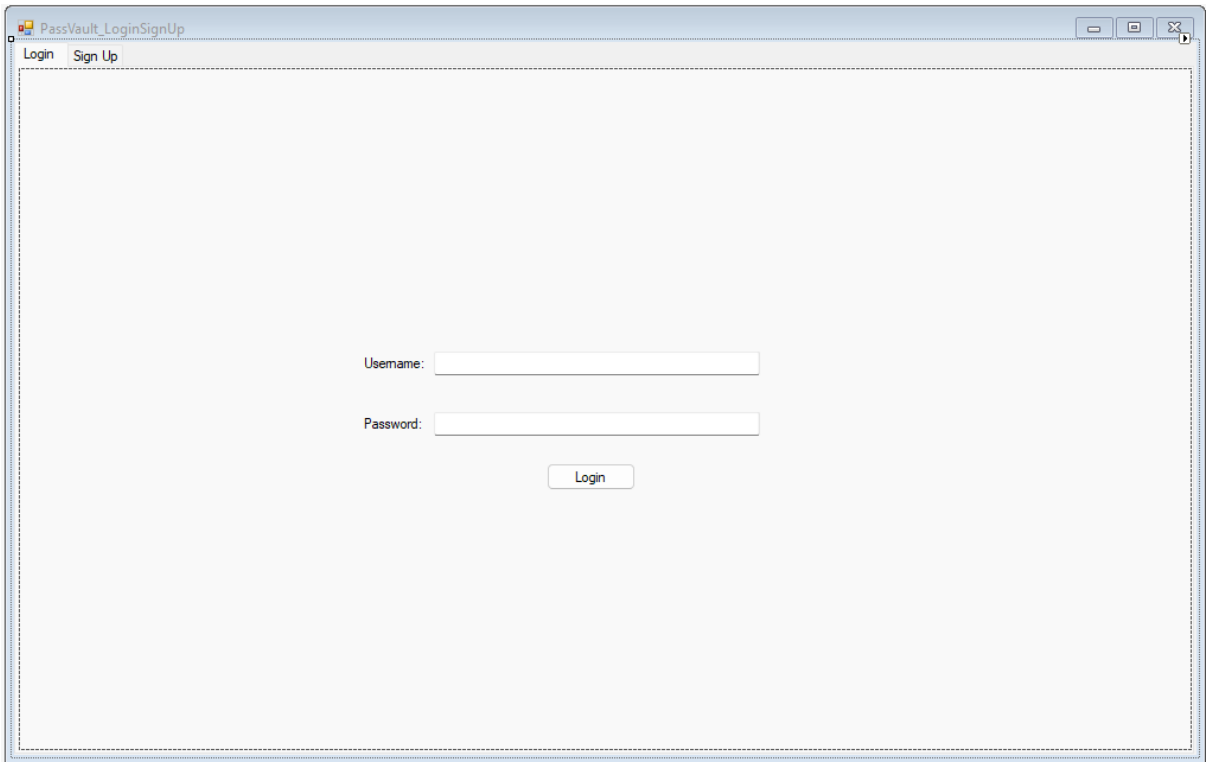
There must be consistency in the program's functionality and design.

- The program will follow a basic CRUD (create, read, update, and delete) structure and design, so each section of passwords will have at least these buttons: save, load, delete.
- Each section will have a data grid to display the user's data in rows and columns, so the data is organised for them.
- There will be shortcuts to certain actions, like loading information to textboxes from the data grid can be done by double clicking on a row, instead of clicking it, then clicking the load button. The shortcut is less tedious as a result.

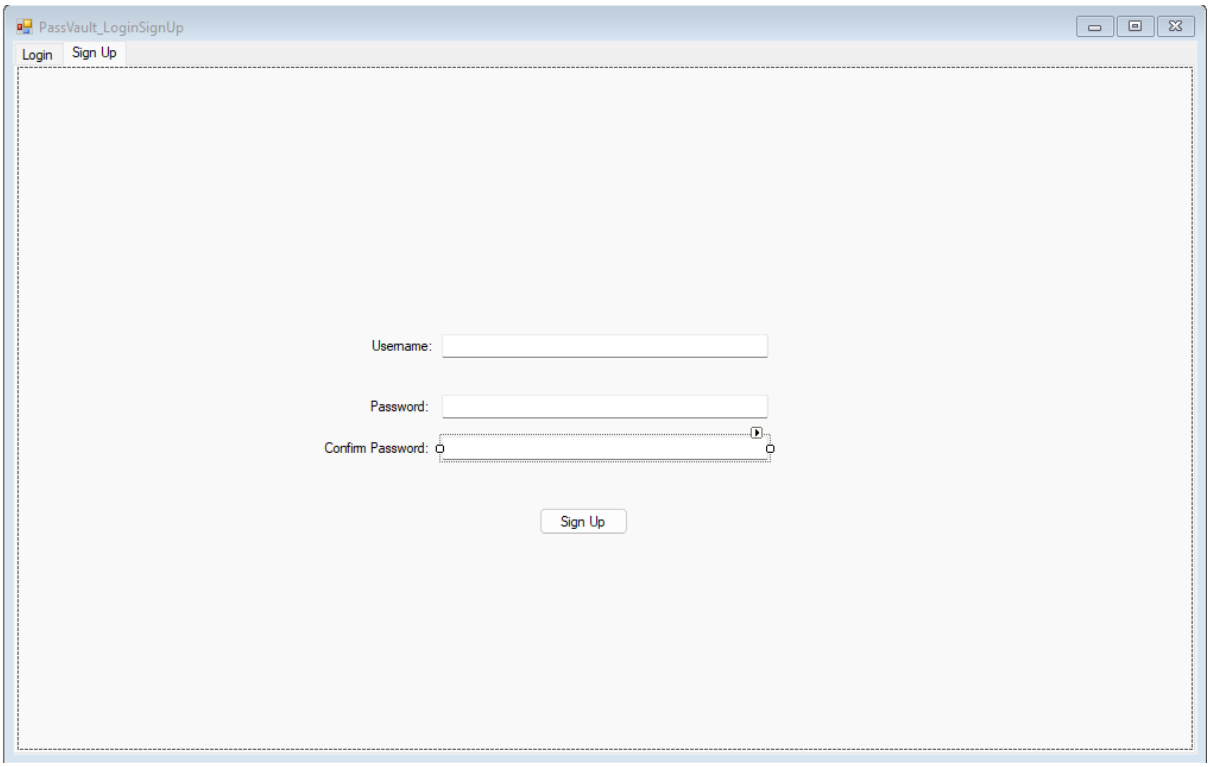
APP DESIGN

The images shown are not final menus, the layout and colour scheme of the features may differ slightly.

- Login/ Sign Up Menu



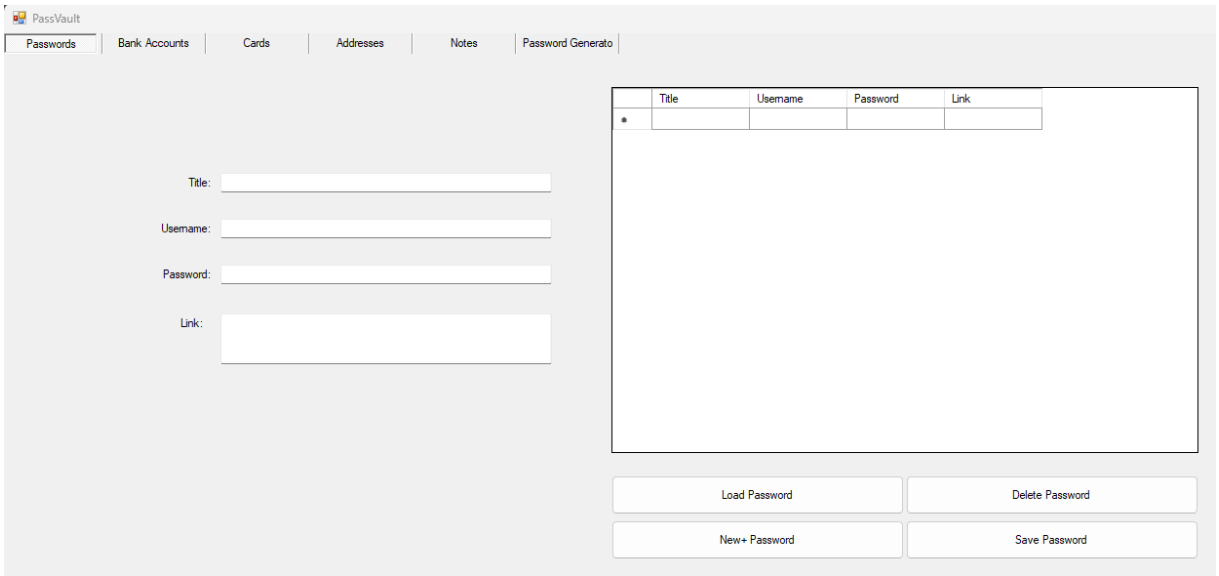
This will include a login screen with a username and password to fill, and a button that says sign up. The user can click sign up if they do not have an account, which will take them to another tab to create one:

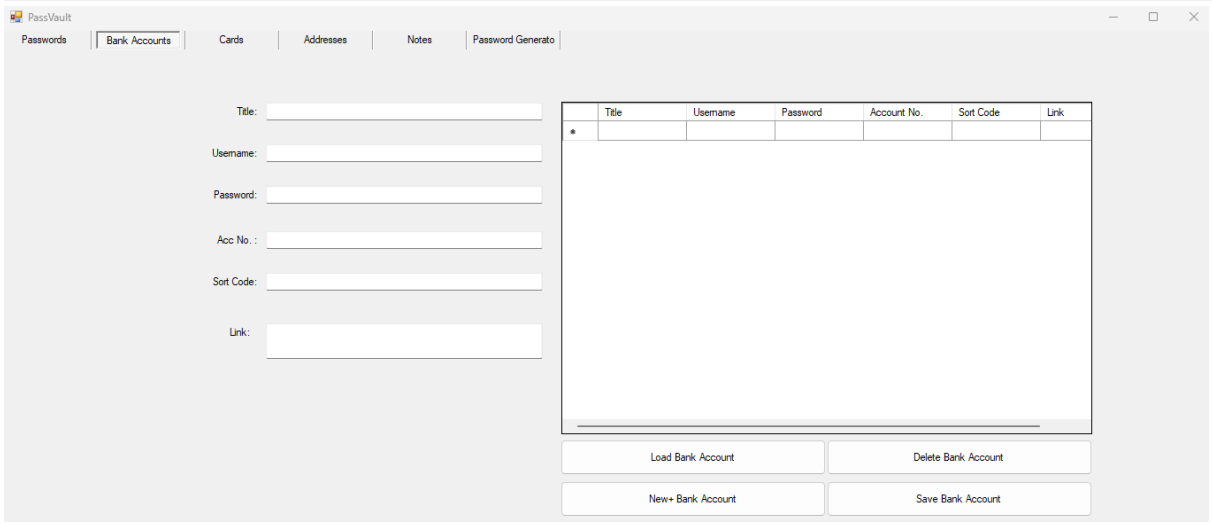


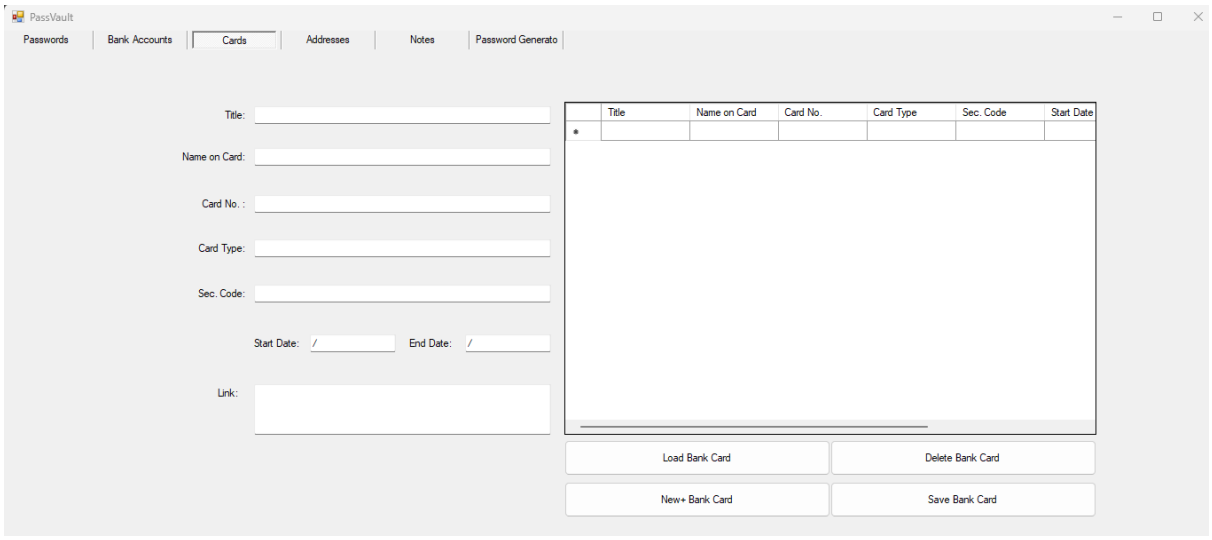
The account will be stored in a database once created.

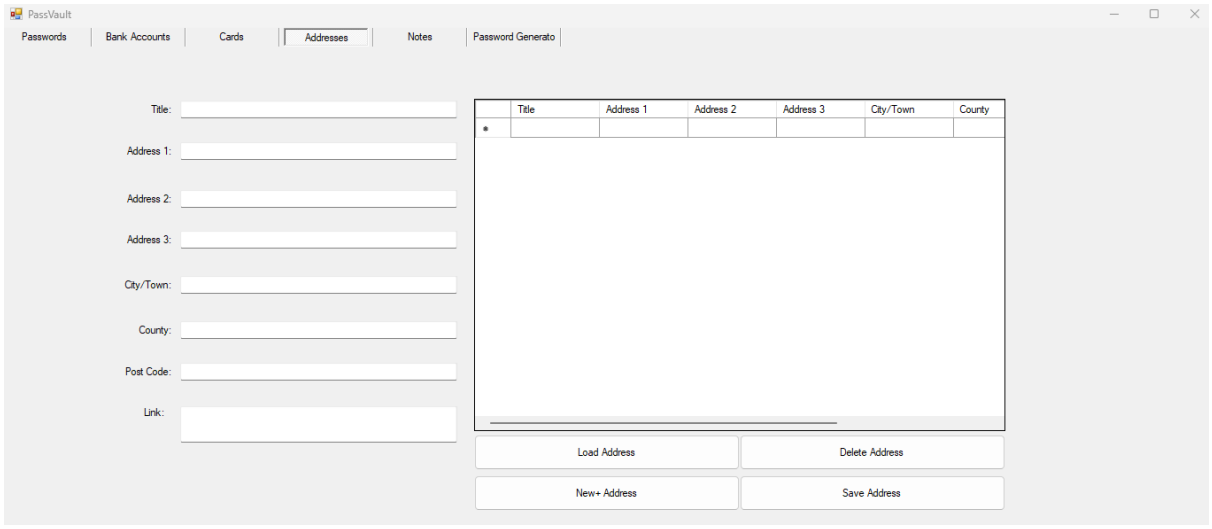
- Vault Menu:

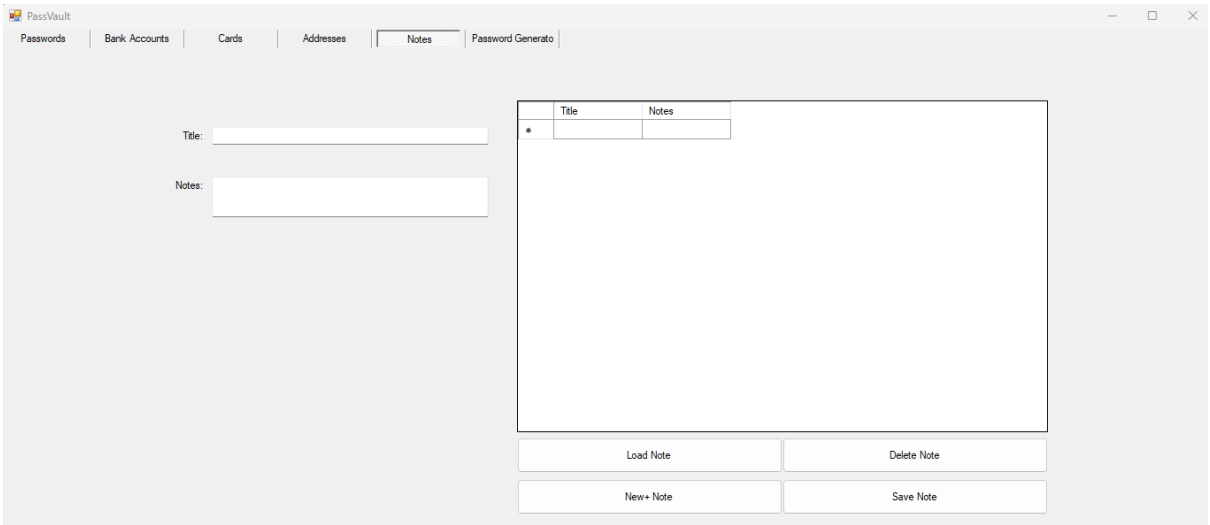
This is the page the user sees when logging in. It will include few tabs for the type of information a user may store, and their respective sub-menus:


PassVault
Passwords
Bank Accounts
Cards
Addresses
Notes
Password Generato
Title:
Username:
Password:
Link:
Title
Username
Password
Link
Load Password
Delete Password
New+ Password
Save Password


PassVault
Passwords
Bank Accounts
Cards
Addresses
Notes
Password Generato
Title:
Username:
Password:
Acc No. :
Sort Code:
Link:
Title
Username
Password
Account No.
Sort Code
Link
Load Bank Account
Delete Bank Account
New+ Bank Account
Save Bank Account


PassVault
Passwords
Bank Accounts
Cards
Addresses
Notes
Password Generato
Title:
Name on Card:
Card No. :
Card Type:
Sec. Code:
Start Date: /
End Date: /
Link:
Title
Name on Card
Card No.
Card Type
Sec. Code
Start Date
Load Bank Card
Delete Bank Card
New+ Bank Card
Save Bank Card


PassVault
Passwords
Bank Accounts
Cards
Addresses
Notes
Password Generato
Title:
Address 1:
Address 2:
Address 3:
City/Town:
County:
Post Code:
Link:
Title
Address 1
Address 2
Address 3
City/Town
County
Load Address
Delete Address
New+ Address
Save Address



The link boxes are large because links are normally several lines long. The note boxes have been made several lines long too in case the user needs to write extensive information.

Visual Studios has a pre-built tab control panel, so I implemented one and only change the number of tabs and the shown text, instead of creating panel controls from scratch.

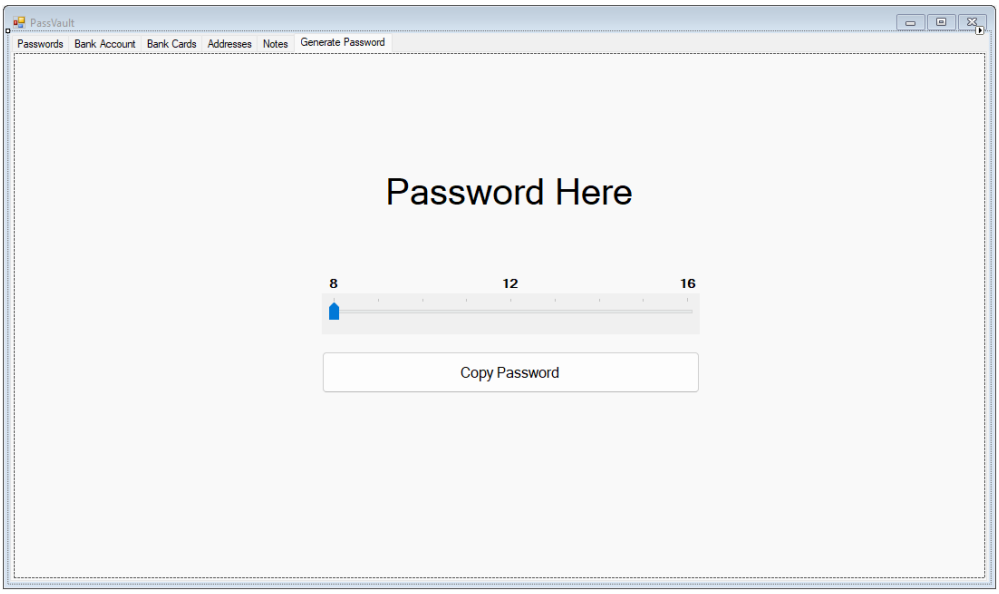
It also includes a pre-built data grid. The data grid in Visual Studio allows information to be displayed in organised rows and columns. Rows may be each account saved, and columns may be the title, username, etc., making it perfect for storing the different text box inputs for each item.

The user can press save, which inputs information from the text boxes into the data grid and clears the text in the note boxes, so they may type more information into the text boxes.

The user can select anything from the data grid by double clicking the information or by single clicking it and then clicking load.

They can delete items by clicking the row in the data grid and then clicking delete.

They may create a new item (a note in this case) by clicking new. The text boxes will be cleared, and all entered information will be saved, so the user can enter new information.



This is a password generator. The user simply needs to slide arrow to the desired character length that they wish, then that password is outputted onto the label. The password can be eight to sixteen characters long, so it is a suitable length required by all passwords.

DATABASE DESIGN

Each tab will have their own data table, all related to a single account. The data will be organised like so:

MasterAccount		
UserID	MasterUsername	MasterPassword

- There is a user ID to uniquely identify the user's PassVault account, in case if there are duplicate passwords, and to link the user to respective information.
- Primary Key

Passwords					
PasswordID	Title	Username	Password	Link	UserID

- Each password has a unique ID, in case if the user includes duplicate passwords. The same applies to the tables below.

- Foreign Key

BankAccounts							
Bank AccountID	Title	Username	Password	Acc. No.	Sort Code	Link	UserID

Cards									
CardID	Title	Name on Card	Card No.	Card Type	Sec. Code	Start Date	End Date	Link	UserID

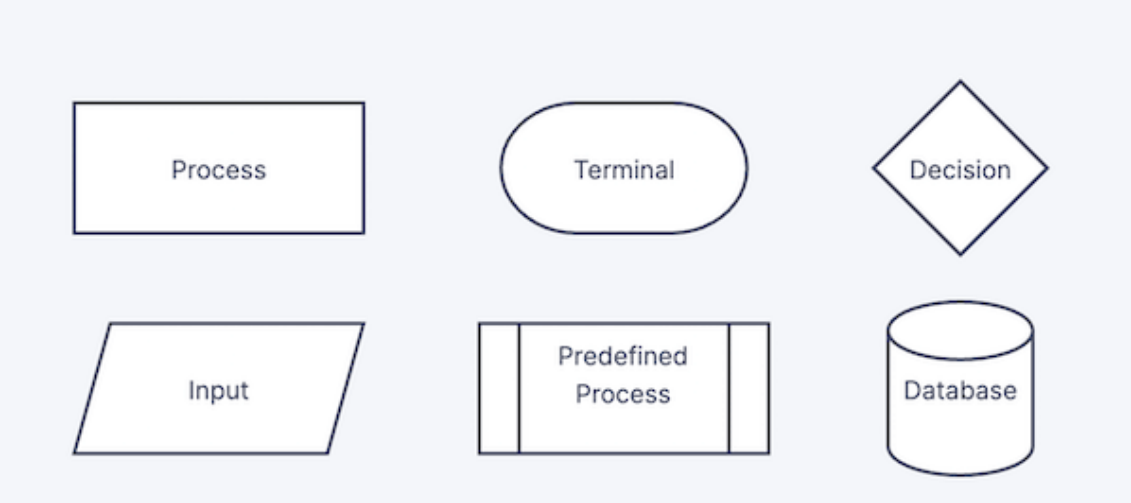
Addresses									
AddressID	Title	Line 1	Line 2	Line 3	City /Town	County /State	Postcode	Link	UserID

Notes			
NoteID	Title	Notes	UserID

Each database will save data IDs as auto numbers, links as hyperlinks, and everything else as texts.

PSEUDOCODE, FLOWCHARTS, DATA STRUCTURES & VARIABLES

Flow diagrams are a graphical way of representing a series of steps of a problem. These are the most abundant symbols in flowcharts that I will use in my project:



There are pre-built procedures for each button: 'Load', 'Delete', 'New+' and 'Save' in Visual Studio for when these buttons are clicked. There are also ones for sliders, tab controls and data grids, which will also be used in the code.

Each of the password types will have a data grid (referred to as dataGridView in code) and a data table (referred to as dataTable in code). The data grid in Visual Studio allows saved information to be displayed in organised rows and columns. This will display the information for the user's point of view in the Windows form. The data table is an internal table of the program that all the user's entered information will be stored in, and the data grid can access from it to display. The table will also store it in the database when the user wants to save the information in the data grid.

Each of the variables and data structures in passwords, bank accounts, cards, addresses, and notes either has the word of their password type or has a single-letter prefix to indicate they are for that section of the vault (e.g., the password section will have variables like editingPassword and p_textBoxTitle). This is to help distinguish what variables they are for with the shortest possible number of characters and to prevent variable names being repeated for each section. The letter 'x' will be replaced with the name of the password type and 'x_' is for their first letter.

- Variables:
  - editingX
    - If it is set to false, the next empty row is taken as the location to input the information in the data grid.

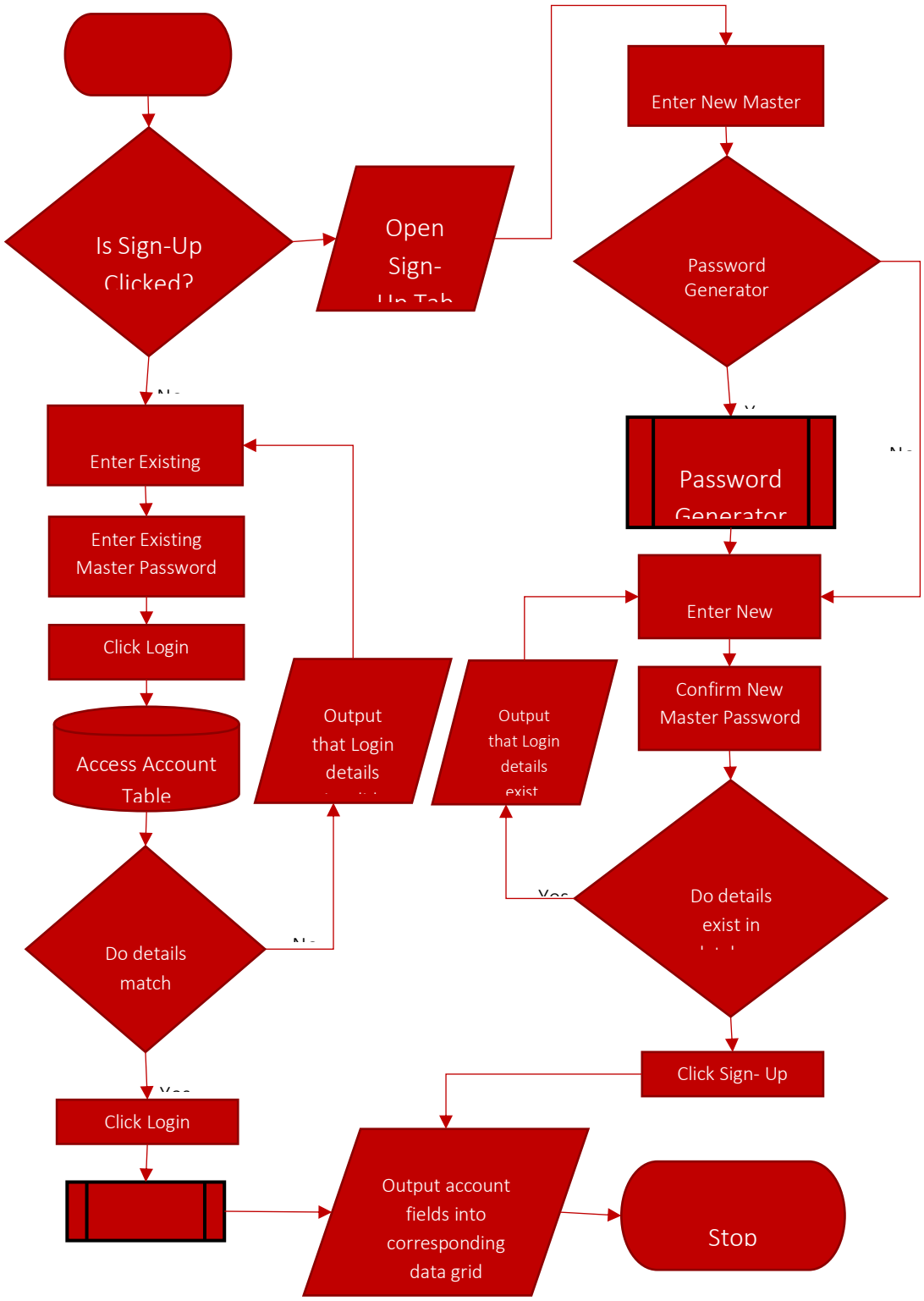
- If it is set to true, the selected row by the user which already has information, is overwritten when save is clicked.

- Procedures:
  - x_buttonNew_Click()
    - It inputs new information into the data grid when clicked.
    - The text boxes are cleared so that they are ready for inputs.
  - x_buttonSave_Click()
    - This saves text box inputs into the data table and displays it in the next row of the data grid when clicked.
    - Or if an existing item is being edited, it will overwrite it.
    - The program will use the hashing function to hash the saved password before storing it in the database.
  - x_buttonLoad_Click()
    - The procedure can be executed when a row is double clicked, or single clicked and then the load button is clicked.
    - If the load button is clicked first, the program will not know what needs to be loaded, so an error message outputs 'Select a row.'
    - The row's information is loaded into the text boxes.
    - The editingX variable is set to TRUE so the program will know it already exists in the code and it needs to be overwritten.
  - x_buttonDelete_Click()
    - The procedure can be executed when row is clicked and then the delete button is clicked.
    - If the delete button is clicked first, the program will not know what needs to be deleted, so an error message outputs 'select a row' in a message box.
    - When an item is deleted, the row it is in will disappear.

Login / Sign Up

FLOWCHART


Login / Sign Up


Is Sign-Up Clicked?
Open Sign-
Enter New Master
Password Generator
Enter Existing
Enter Existing Master Password
Click Login
Access Account Table
Output that Login details
Output that Login details exist
Password
Enter New
Confirm New Master Password
Do details match
Do details exist in
Click Login
Click Sign- Up
Output account fields into corresponding data grid
Stop

Description: The flowchart here checks whether the sign-up button is clicked. If so, it branches to another part of the program: the sign-up tab.

PSEUDOCODE

```
SET currentTab TO tabLogIn

PROCEDURE tabSignUp_Click()

      SET currentTab TO tabSignUp

END PROCEDURE


PROCEDURE su_buttonSignUp_Click()

      REQUEST DATA IN PassVault DATABASE

      IF su_textBoxMasterUsername.Text IN MasterAccount TABLE IN PassVault DATABASE DO

            IF su_textBoxMasterPassword.Text.Length < 8 OR su_textBoxMasterPassword.Text.Length > 16

                  ADD su_textBoxMasterPassword.Text AND su_textBoxMasterUsername.Text TO MasterAccount TABLE

            ELSE DO

                  RETURN "Password must be 8 - 16 characters." TO DISPLAY

            END IF

      ELSE DO

            RETURN "Username exists already." TO DISPLAY

      END IF

END PROCEDURE
```

- Procedures
  - tabSignUp_Click()

- Functions
  - su_buttonSignUp()

- ‘REQUEST DATA’ allows the program to access another application’s data (the Microsoft Access Database in this case, to add new login details)
- The program checks if the entered master username already exists in the database. If it does, the program immediately stops and outputs that the username is taken, saving time having to run the password length check.
- If the master username is not taken, the program moves onto the password length check.
- If the password is not in the 8-to-sixteen-character long range, the program stops and outputs that it needs to be.
- Otherwise, the new account is created and loaded into the database.

```
PROCEDURE tabLogIn_Click()
      SET currentTab TO tabLogIn
END PROCEDURE

PROCEDURE l_buttonLogIn_Click()
      REQUEST DATA IN PassVault DATABASE
      IF l_textBoxMasterUsername.Text IN MasterAccount TABLE IN PassVault DATABASE DO
            IF l_textBoxMasterPassword.Text FIELD IN l_textBoxMasterUsername.Text FIELD
                  LOAD l_textBoxMasterUsername.Text PRIMARY KEY TO Vault
            ELSE DO
                  RETURN "Invalid password." TO DISPLAY
            ENDIF
      ELSE DO
            RETURN "Invalid username." TO DISPLAY
      END IF
END PROCEDURE
```

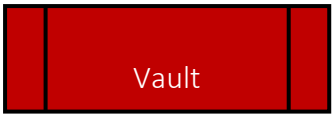
- Procedures
  - tabLogIn_Click()

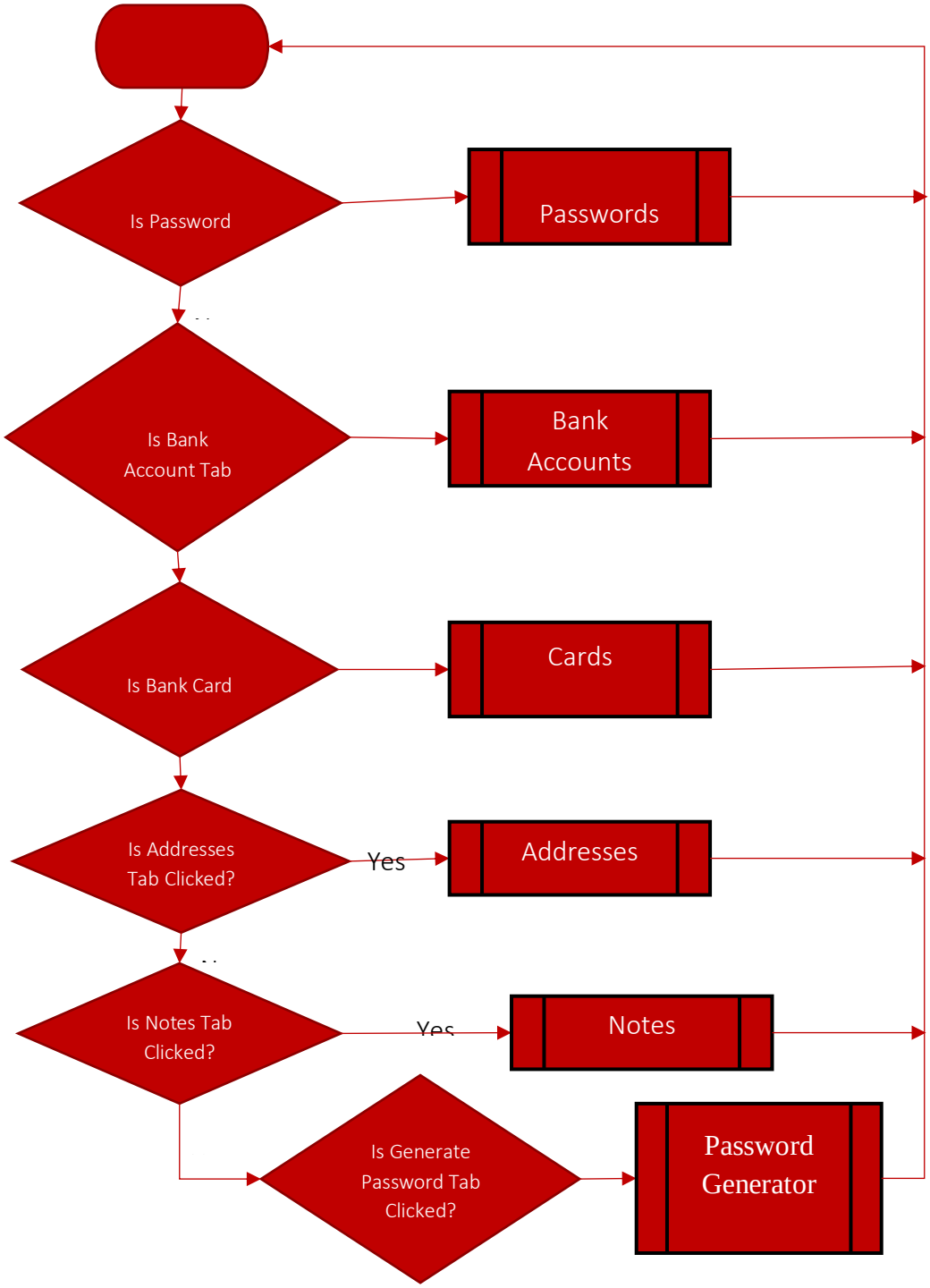
- Functions
  - l_buttonLogIn_Click()
    - ‘REQUEST DATA’ allows the program to access another application’s data (the Microsoft Access Database in this case, to check for login details)
    - The program checks if the entered master username exists in the database.
    - If it does, the password check is then carried out.

- If it does not, the program immediately stops and outputs that the username is invalid, saving time having to cycle the master password field to check the password matches.
- If the password matches, the Vault menu is loaded. The master username is a primary key to the account's saved items. The vault is loaded, and all the saved items linked to it will be sent to the Vault in their respective locations.
- If the password is invalid the program stops and outputs so.

Vault

FLOWCHART




Is Password
Passwords
Is Bank Account Tab
Bank Accounts
Is Bank Card
Cards
Is Addresses Tab Clicked?
Yes
Addresses
Is Notes Tab Clicked?
Yes
Notes
Is Generate Password Tab Clicked?
Password Generator

Description: This will be in the form of Visual Studio's built-in tab controls. It will constantly check if the tabs are selected.

PSEUDOCODE

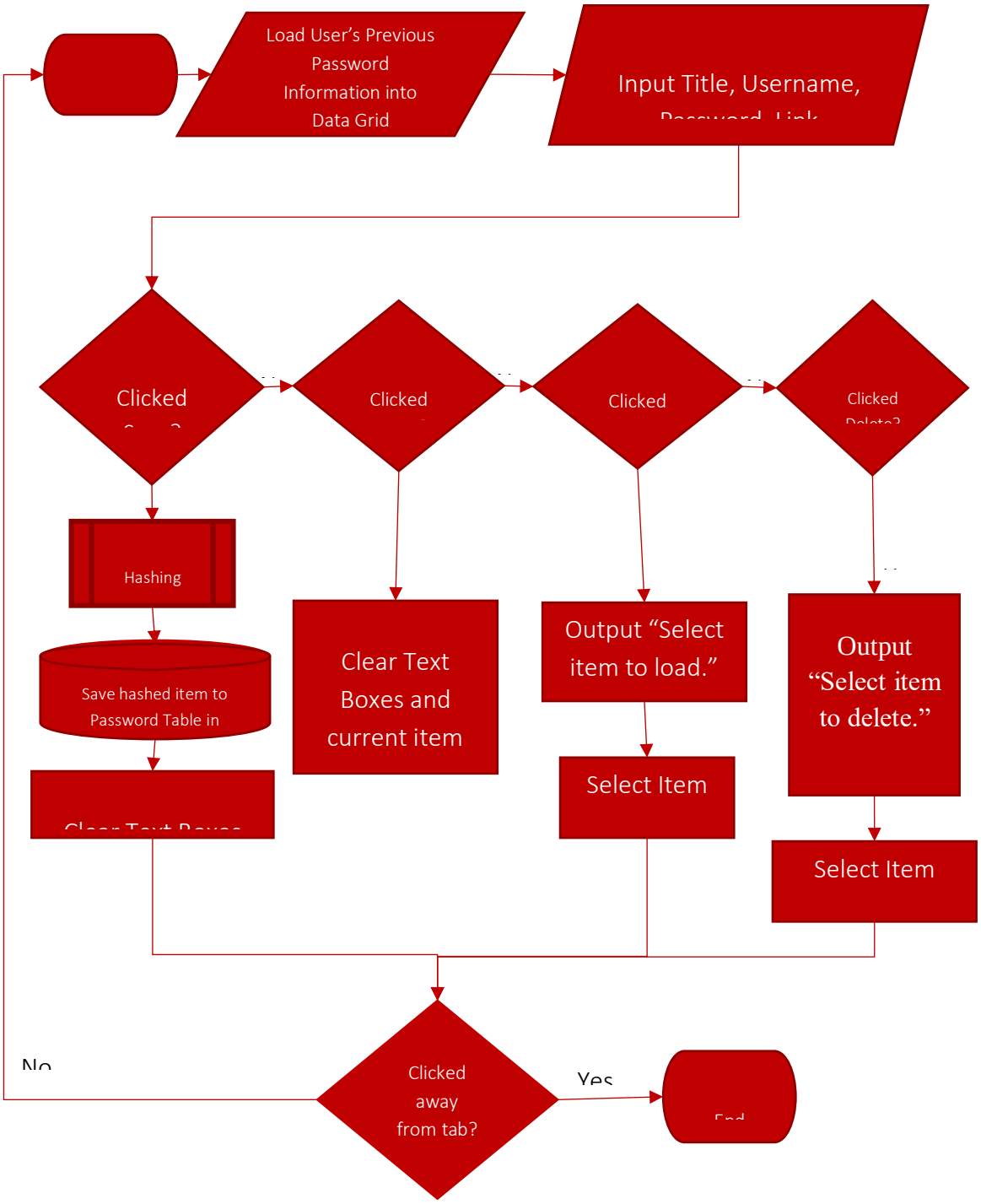
```
--- Vault ---


PROCEDURE tabPassword_Click()
        SET currentTab TO tabPassword
END PROCEDURE


PROCEDURE tabBankAccount_Click()
        SET currentTab TO tabBankAccount
END PROCEDURE


PROCEDURE tabCards_Click()
        SET currentTab TO tabCards
END PROCEDURE
```

```
PROCEDURE tabAddresses_Click()

        SET currentTab TO tabAddresses

END PROCEDURE



PROCEDURE tabNotes_Click()

        SET currentTab TO tabNotes

END PROCEDURE



PROCEDURE tabPasswordGenerator_Click()

        SET currentTab TO tabPasswordGenerator

END PROCEDURE
```

C# has built in tab control procedures, which will act as the user's navigation around the different types of passwords they want to save.

Passwords

FLOWCHART


Load User's Previous Password Information into Data Grid
Input Title, Username,
Clicked
Clicked
Clicked
Clicked
Hashing
Save hashed item to Password Table in
Clear Text Boxes and current item
Output "Select item to load."
Output "Select item to delete."
Select Item
Select Item
Clicked away from tab?
No
Yes

Description: The password, bank account, cards, addresses, and notes sections of the program constantly checks for button clicks from the user.

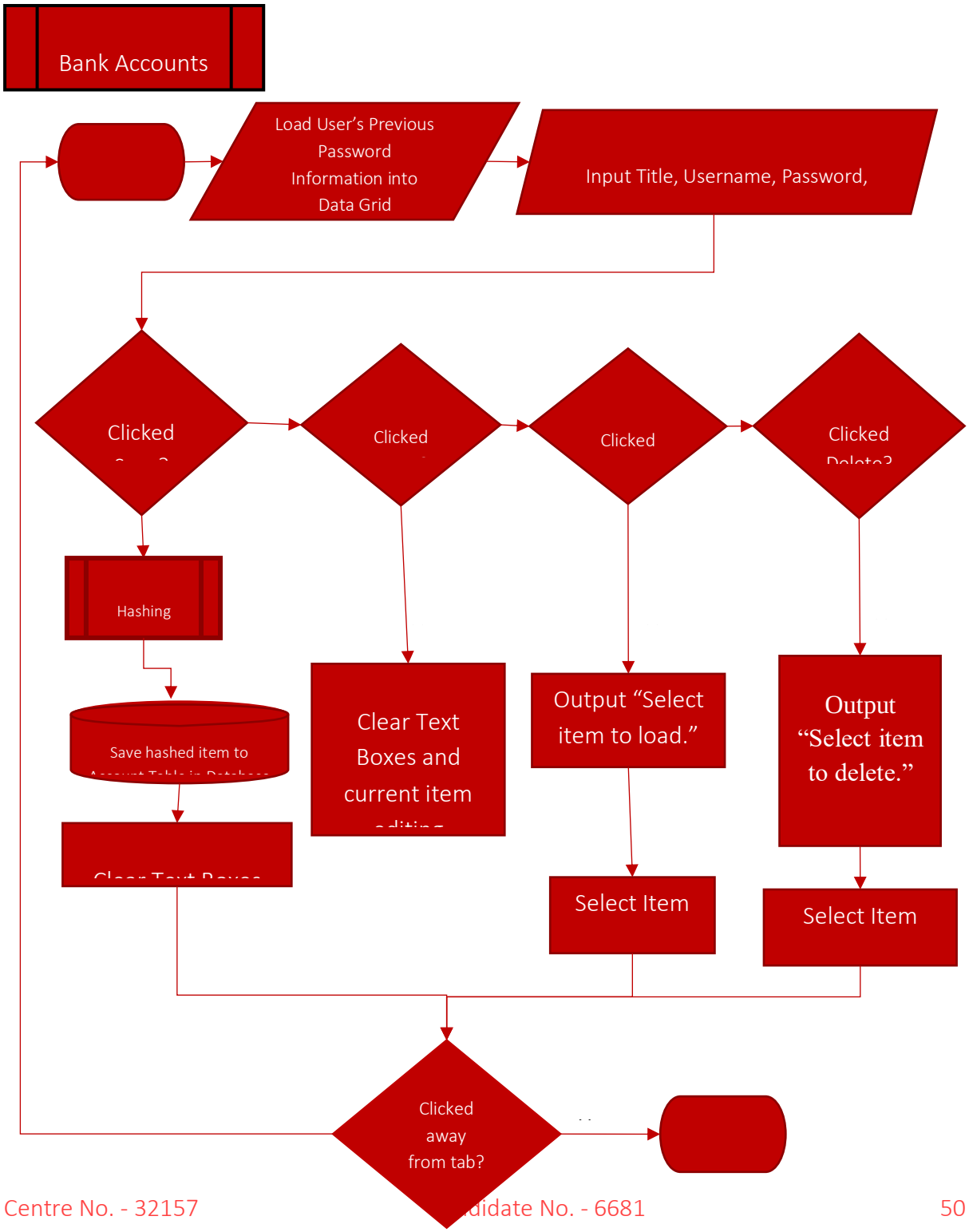
PSEUDOCODE

```
--- Passwords ---


SET p_dataTable TO NEW DATATABLE

SET editingPassword TO BOOL FALSE


PROCEDURE p_buttonDelete_Click()

        TRY

                DELETE p_dataTable.ROWS[p_dataGridView.CURRENTROW]

        CATCH Exception

                SEND "Select row to delete." TO DISPLAY

        END TRYCATCH

END PROCEDURE


PROCEDURE p_buttonLoad_Click OR p_dataGridView_DoubleClick()

        TRY

                SET p_textBoxTitle.Text TO p_dataTable.ROWS[p_dataGridView.CURRENTROW].ITEMARRAY[0]
                SET p_textBoxUsername.Text TO p_dataTable.ROWS[p_dataGridView.CURRENTROW].ITEMARRAY[1]
                SET p_textBoxPassword.Text TO p_dataTable.ROWS[p_dataGridView.CURRENTROW].ITEMARRAY[2]
                SET p_textBoxLink.Text TO p_dataTable.ROWS[p_dataGridView.CURRENTROW].ITEMARRAY[3]
                SET editingPassword TO BOOL TRUE

        CATCH Exception

                SEND "Selected empty row." TO DISPLAY

        END TRYCATCH

END PROCEDURE
```

```
PROCEDURE p_buttonNew_Click()

        SET p_textBoxTitle.Text TO ""
        SET p_textBoxUsername.Text TO ""
        SET p_textBoxPassword.Text TO ""
        SET p_textBoxLink.Text TO ""

END PROCEDURE


PROCEDURE p_buttonSave_Click()

        REQUEST DATA IN Passwords TABLE IN PassVault DATABASE

        IF editingPassword == TRUE DO

                SET p_dataTable.ROWS[p_dataGridView.CURRENTROW]["Title"] TO p_textBoxTitle.Text
                ADD toHash256(p_textBoxTitle.Text ) TO Passwords TABLE

                SET p_dataTable.ROWS[p_dataGridView.CURRENTROW]["Username"] TO p_textBoxUsername.Text
                ADD toHash256( p_textBoxUsername.Text) TO Passwords TABLE

                SET p_dataTable.ROWS[p_dataGridView.CURRENTROW]["Password"] TO p_textBoxPassword.Text
                ADD toHash256(p_textBoxPassword.Text ) TO Passwords TABLE

                SET p_dataTable.ROWS[p_dataGridView.CURRENTROW]["Link"] TO p_textBoxLink.Text
                ADD toHash256(p_textBoxLink.Text ) TO Passwords TABLE

        ELSE DO

                SET p_dataTable.ROWS("Title") TO p_textBoxTitle.Text
                OVERWRITE toHash256( p_textBoxTitle.Text) IN Passwords TABLE

                SET p_dataTable.ROWS("Username") TO p_textBoxUsername.Text
                OVERWRITE toHash256( p_textBoxUsername.Text) IN Passwords TABLE

                SET p_dataTable.ROWS("Password") TO p_textBoxPassword.Text
                OVERWRITE toHash256( p_textBoxPassword.Text) IN Passwords TABLE

                SET p_dataTable.ROWS("Link") TO p_textBoxLink.Text
                OVERWRITE toHash256( p_textBoxLink.Text) IN Passwords TABLE

        END IF

        SET p_textBoxTitle.Text TO ""
        SET p_textBoxUsername.Text TO ""
        SET p_textBoxPassword.Text TO ""
        SET p_textBoxLink.Text TO ""
        SET editingPassword TO False

END PROCEDURE
```

Bank Accounts

FLOWCHART


Bank Accounts
Load User's Previous Password Information into Data Grid
Input Title, Username, Password,
Clicked
Clicked
Clicked
Clicked
Hashing
Save hashed item to
Clear Text Boxes and current item
Output "Select item to load."
Output "Select item to delete."
Select Item
Select Item
Clicked away from tab?

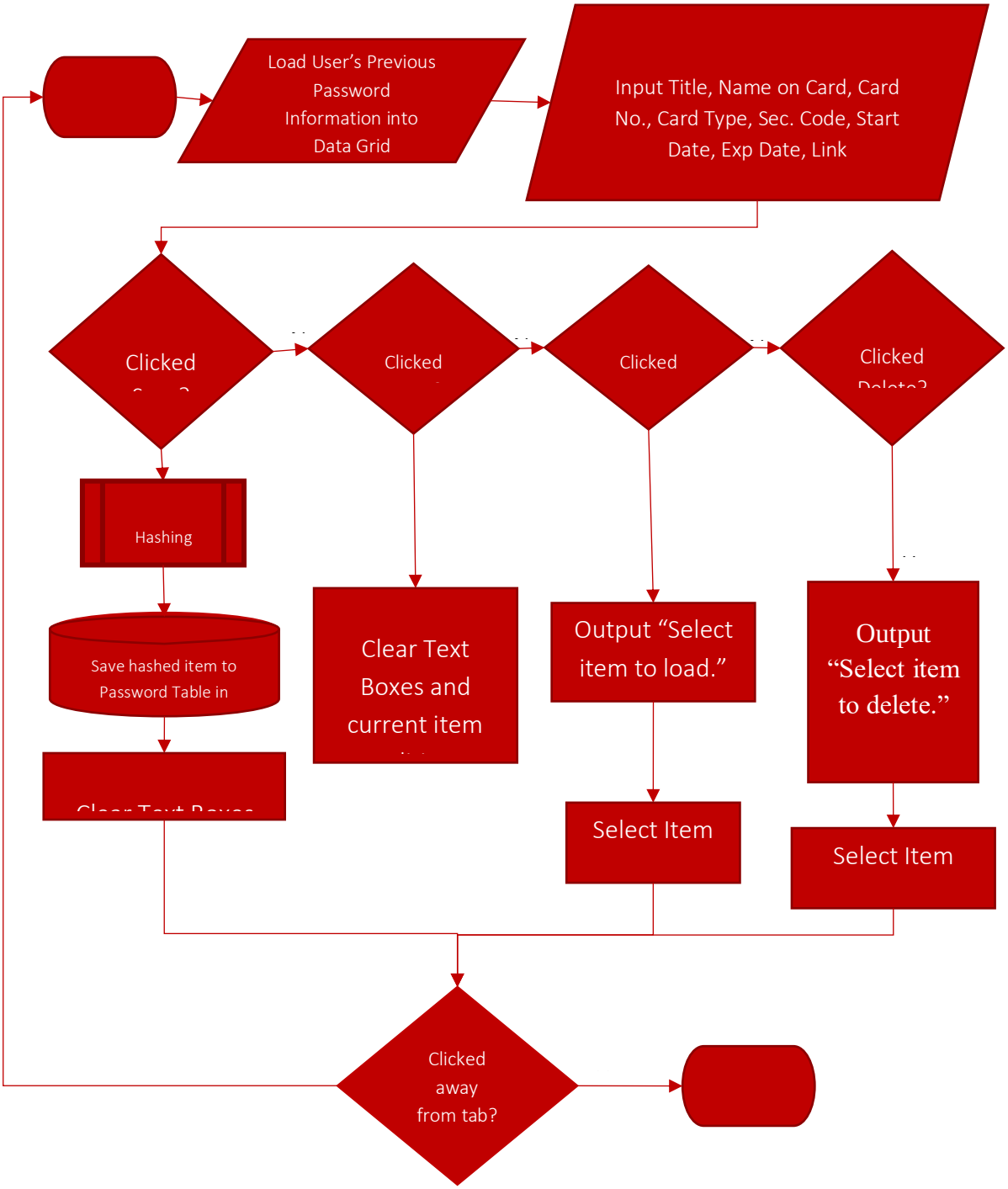
PSEUDOCODE

```
--- Bank Accounts ---


SET ba_dataTable TO NEW DATATABLE

SET editingBankAccount TO BOOL FALSE


PROCEDURE ba_buttonDelete_Click()

        TRY

                DELETE ba_dataTable.ROWS[ba_dataGridView.CURRENTROW]

        CATCH Exception

                SEND "Select row to delete." TO DISPLAY

        END TRYCATCH

END PROCEDURE


PROCEDURE ba_buttonLoad_Click OR ba_dataGridView_DoubleClick()

        TRY

                SET ba_textBoxTitle.Text TO ba_dataTable.ROWS[ba_dataGridView.CURRENTROW].ITEMARRAY[0]
                SET ba_textBoxUsername.Text TO ba_dataTable.ROWS[ba_dataGridView.CURRENTROW].ITEMARRAY[1]
                SET ba_textBoxPassword.Text TO ba_dataTable.ROWS[ba_dataGridView.CURRENTROW].ITEMARRAY[2]
                SET ba_textBoxAccNo.Text TO ba_dataTable.ROWS[ba_dataGridView.CURRENTROW].ITEMARRAY[3]
                SET ba_textBoxSortCode.Text TO ba_dataTable.ROWS[ba_dataGridView.CURRENTROW].ITEMARRAY[4]
                SET ba_textBoxLink.Text TO ba_dataTable.ROWS[ba_dataGridView.CURRENTROW].ITEMARRAY[5]
                SET editingBankAccount TO BOOL TRUE

        CATCH Exception

                SEND "Selected empty row." TO DISPLAY

        END TRYCATCH

END PROCEDURE
```

```
PROCEDURE ba_buttonNew_Click()

        SET ba_textBoxTitle.Text TO ""
        SET ba_textBoxUsername.Text TO ""
        SET ba_textBoxPassword.Text TO ""
        SET ba_textBoxAccNo.Text TO ""
        SET ba_textBoxSortCode.Text TO ""
        SET ba_textBoxLink.Text TO ""

END PROCEDURE


PROCEDURE ba_buttonSave_Click()

        REQUEST DATA IN BankAccounts TABLE IN PassVault DATABASE

        IF editingBankAccount == TRUE DO

                SET ba_dataTable.ROWS[ba_dataGridView.CURRENTROW]["Title"] TO ba_textBoxTitle.Text
                ADD toHash256(ba_textBoxTitle.Text ) TO BankAccounts TABLE

                SET ba_dataTable.ROWS[ba_dataGridView.CURRENTROW]["Username"] TO ba_textBoxUsername.Text
                ADD toHash256( ba_textBoxUsername.Text) TO BankAccounts TABLE

                SET ba_dataTable.ROWS[ba_dataGridView.CURRENTROW]["Password"] TO ba_textBoxPassword.Text
                ADD toHash256( ba_textBoxPassword.Text) TO BankAccounts TABLE

                SET ba_dataTable.ROWS[ba_dataGridView.CURRENTROW]["Account No."] TO ba_textBoxAccNo.Text
                ADD toHash256(  ba_textBoxAccNo.Text) TO BankAccounts TABLE

                SET ba_dataTable.ROWS[ba_dataGridView.CURRENTROW]["Sort Code"] TO ba_textBoxSortCode.Text
                ADD toHash256(ba_textBoxSortCode.Text ) TO BankAccounts TABLE

                SET ba_dataTable.ROWS[ba_dataGridView.CURRENTROW]["Link"] TO ba_textBoxLink.Text
                ADD toHash256(ba_textBoxLink.Text ) TO BankAccounts TABLE


        ELSE DO

                SET ba_dataTable.ROWS("Title") TO ba_textBoxTitle.Text
                OVERWRITE toHash256( ba_textBoxTitle.Text) IN BankAccounts TABLE

                SET ba_dataTable.ROWS("Username") TO ba_textBoxUsername.Text
                OVERWRITE toHash256( ba_textBoxUsername.Text) IN BankAccounts TABLE

                SET ba_dataTable.ROWS("Password") TO ba_textBoxPassword.Text
                OVERWRITE toHash256( ba_textBoxPassword.Text) IN BankAccounts TABLE

                SET ba_dataTable.ROWS("Account No.") TO ba_textBoxAccNo.Text
                OVERWRITE toHash256(ba_textBoxAccNo.Text ) IN BankAccounts TABLE

                SET ba_dataTable.ROWS("Sort Code") TO ba_textBoxSortCode.Text
                OVERWRITE toHash256(ba_textBoxSortCode.Text ) IN BankAccounts TABLE

                SET ba_dataTable.ROWS("Link") TO ba_textBoxLink.Text
                OVERWRITE toHash256( ba_textBoxLink.Text) IN BankAccounts TABLE

        END IF

        SET ba_textBoxTitle.Text TO ""
        SET ba_textBoxUsername.Text TO ""
        SET ba_textBoxPassword.Text TO ""
        SET ba_textBoxAccNo.Text TO ""
        SET ba_textBoxSortCode.Text TO ""
        SET ba_textBoxLink.Text TO ""
        SET editingBankAccount TO False

END PROCEDURE
```

Cards

FLOWCHART


Load User's Previous Password Information into Data Grid
Input Title, Name on Card, Card No., Card Type, Sec. Code, Start Date, Exp Date, Link
Clicked
Clicked
Clicked
Clicked
Hashing
Save hashed item to Password Table in
Clear Text Boxes and current item
Output "Select item to load."
Output "Select item to delete."
Select Item
Select Item
Clicked away from tab?

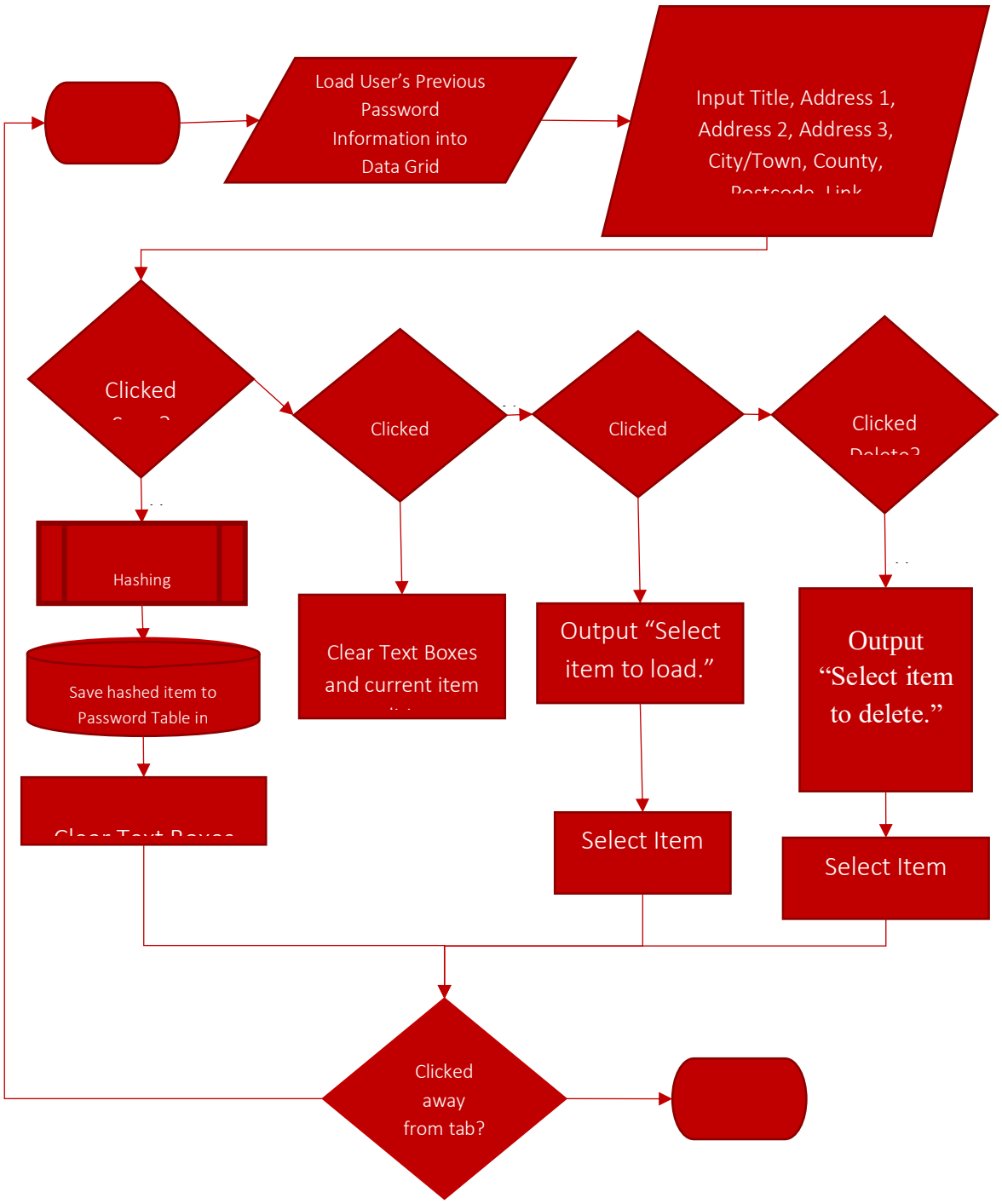
PSEUDOCODE

```
--- Cards ---


SET c_dataTable TO NEW DATATABLE

SET editingCards TO BOOL FALSE


PROCEDURE c_buttonDelete_Click()

        TRY

                DELETE c_dataTable.ROWS[c_dataGridView.CURRENTROW]

        CATCH Exception

                SEND "Select row to delete." TO DISPLAY

        END TRYCATCH

END PROCEDURE


PROCEDURE c_buttonLoad_Click OR c_dataGridView_DoubleClick()

        TRY

                SET c_textBoxTitle.Text TO c_dataTable.ROWS[c_dataGridView.CURRENTROW].ITEMARRAY[0]
                SET c_textBoxNameOnCard.Text TO c_dataTable.ROWS[c_dataGridView.CURRENTROW].ITEMARRAY[1]
                SET c_textBoxCardNo.Text TO c_dataTable.ROWS[c_dataGridView.CURRENTROW].ITEMARRAY[2]
                SET c_textBoxCardType.Text TO c_dataTable.ROWS[c_dataGridView.CURRENTROW].ITEMARRAY[3]
                SET c_textBoxSecCode.Text TO c_dataTable.ROWS[c_dataGridView.CURRENTROW].ITEMARRAY[4]
                SET c_textBoxStartDate.Text TO c_dataTable.ROWS[c_dataGridView.CURRENTROW].ITEMARRAY[5]
                SET c_textBoxEndDate.Text TO c_dataTable.ROWS[c_dataGridView.CURRENTROW].ITEMARRAY[6]
                SET c_textBoxLink.Text TO c_dataTable.ROWS[c_dataGridView.CURRENTROW].ITEMARRAY[7]
                SET editingCards TO BOOL TRUE

        CATCH Exception

                SEND "Selected empty row." TO DISPLAY

        END TRYCATCH

END PROCEDURE


PROCEDURE c_buttonNew_Click()

        SET c_textBoxTitle.Text TO ""
        SET c_textBoxNameOnCard.Text TO ""
        SET c_textBoxCardNo.Text TO ""
        SET c_textBoxCardType.Text TO ""
        SET c_textBoxSecCode.Text TO ""
        SET c_textBoxStartDate.Text TO ""
        SET c_textBoxEndDate.Text TO ""
        SET c_textBoxLink.Text TO ""

END PROCEDURE
```

```
PROCEDURE c_buttonSave_Click()

        REQUEST DATA IN Cards TABLE IN PassVault DATABASE

        IF editingCards == TRUE DO

                SET c_dataTable.ROWS[c_dataGridView.CURRENTROW]["Title"] TO c_textBoxTitle.Text
                ADD toHash256(c_textBoxTitle.Text ) TO Cards TABLE

                SET c_dataTable.ROWS[c_dataGridView.CURRENTROW]["Name on Card"] TO c_textBoxNameOnCard.Text
                ADD toHash256(c_textBoxNameOnCard.Text) TO Cards TABLE

                SET c_dataTable.ROWS[c_dataGridView.CURRENTROW]["Card No."] TO c_textBoxCardNo.Text
                ADD toHash256( c_textBoxCardNo.Text) TO Cards TABLE

                SET c_dataTable.ROWS[c_dataGridView.CURRENTROW]["Card Type"] TO c_textBoxCardType.Text
                ADD toHash256(c_textBoxCardType.Text ) TO Cards TABLE

                SET c_dataTable.ROWS[c_dataGridView.CURRENTROW]["Sec. Code"] TO c_textBoxSecCode.Text
                ADD toHash256(c_textBoxSecCode.Text ) TO Cards TABLE

                SET c_dataTable.ROWS[c_dataGridView.CURRENTROW]["Start Date"] TO c_textBoxStartDate.Text
                ADD toHash256(c_textBoxStartDate.Text ) TO Cards TABLE

                SET c_dataTable.ROWS[c_dataGridView.CURRENTROW]["End Dade"] TO c_textBoxEndDate.Text
                ADD toHash256(c_textBoxEndDate.Text ) TO Cards TABLE

                SET c_dataTable.ROWS[c_dataGridView.CURRENTROW]["Link"] TO c_textBoxLink.Text
                ADD toHash256(c_textBoxLink.Text.Text ) TO Cards TABLE


        ELSE DO

                SET c_dataTable.ROWS("Title") TO c_textBoxTitle.Text
                OVERWRITE toHash256( c_textBoxTitle.Text) IN Cards TABLE

                SET c_dataTable.ROWS("Name on Card") TO c_textBoxNameOnCard.Text
                OVERWRITE toHash256(c_textBoxNameOnCard.Text ) IN Cards TABLE

                SET c_dataTable.ROWS("Card No.") TO c_textBoxCardNo.Text
                OVERWRITE toHash256( c_textBoxCardNo.Text) IN Cards TABLE

                SET c_dataTable.ROWS("Card Type") TO c_textBoxCardType.Text
                OVERWRITE toHash256(c_textBoxCardType.Text ) IN Cards TABLE

                SET c_dataTable.ROWS("Sec. Code") TO c_textBoxSecCode.Text
                OVERWRITE toHash256(c_textBoxSecCode.Text ) IN Cards TABLE

                SET c_dataTable.ROWS("Start Date") TO c_textBoxStartDate.Text
                OVERWRITE toHash256(c_textBoxStartDate.Text ) IN Cards TABLE

                SET c_dataTable.ROWS("End Date") TO c_textBoxEndDate.Text
                OVERWRITE toHash256(c_textBoxEndDate.Text ) IN Cards TABLE

                SET c_dataTable.ROWS("Link") TO c_textBoxLink.Text
                OVERWRITE toHash256(c_textBoxLink.Text ) IN Cards TABLE

        END IF

        SET a_textBoxTitle.Text TO ""
        SET c_textBoxNameOnCard.Text TO ""
        SET c_textBoxCardNo.Text TO ""
        SET c_textBoxCardType.Text TO ""
        SET c_textBoxSecCode.Text TO ""
        SET c_textBoxStartDate.Text TO ""
        SET c_textBoxEndDate.Text TO ""
        SET c_textBoxLink.Text TO ""
        SET editingCards TO False

END PROCEDURE
```

Addresses

FLOWCHART


Load User's Previous Password Information into Data Grid
Input Title, Address 1, Address 2, Address 3, City/Town, County,
Clicked
Clicked
Clicked
Clicked
Hashing
Save hashed item to Password Table in
Clear Text Boxes and current item
Output "Select item to load."
Output "Select item to delete."
Select Item
Select Item
Clicked away from tab?

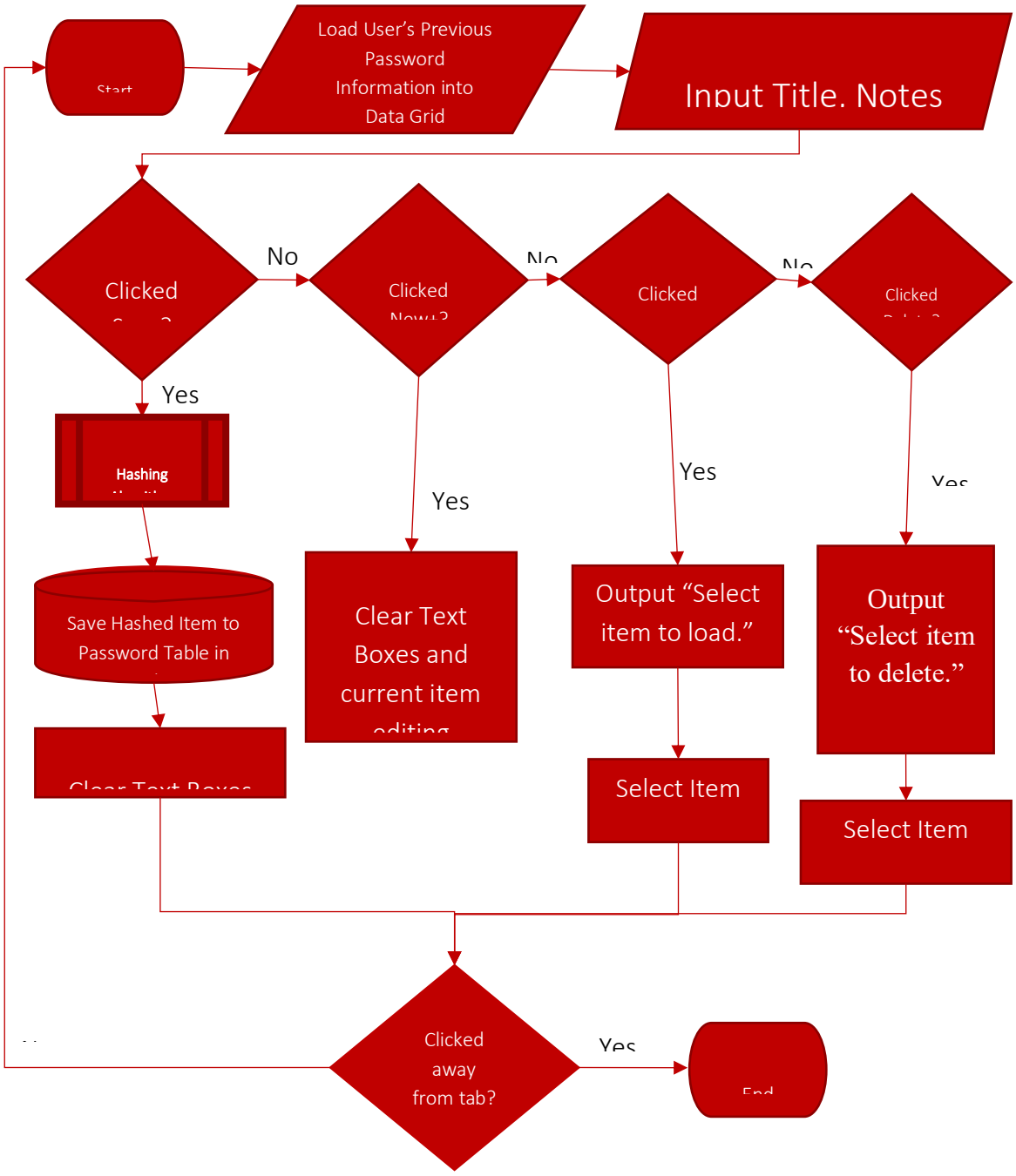
PSEUDOCODE

```
--- Addresses ---


SET a_dataTable TO NEW DATATABLE

SET editingAddress TO BOOL FALSE


PROCEDURE a_buttonDelete_Click()

        TRY

                DELETE a_dataTable.ROWS[a_dataGridView.CURRENTROW]

        CATCH Exception

                SEND "Select row to delete." TO DISPLAY

        END TRYCATCH

END PROCEDURE


PROCEDURE a_buttonLoad_Click OR a_dataGridView_DoubleClick()

        TRY

                SET a_textBoxTitle.Text TO a_dataTable.ROWS[a_dataGridView.CURRENTROW].ITEMARRAY[0]
                SET a_textBoxAddress1.Text TO a_dataTable.ROWS[a_dataGridView.CURRENTROW].ITEMARRAY[1]
                SET a_textBoxAddress2.Text TO a_dataTable.ROWS[a_dataGridView.CURRENTROW].ITEMARRAY[2]
                SET a_textBoxAddress3.Text TO a_dataTable.ROWS[a_dataGridView.CURRENTROW].ITEMARRAY[3]
                SET a_textBoxCity.Text TO a_dataTable.ROWS[a_dataGridView.CURRENTROW].ITEMARRAY[4]
                SET a_textBoxCounty.Text TO a_dataTable.ROWS[a_dataGridView.CURRENTROW].ITEMARRAY[5]
                SET a_textBoxPostCode.Text TO a_dataTable.ROWS[a_dataGridView.CURRENTROW].ITEMARRAY[6]
                SET a_textBoxLink.Text TO a_dataTable.ROWS[a_dataGridView.CURRENTROW].ITEMARRAY[7]
                SET editingAddress TO BOOL TRUE

        CATCH Exception

                SEND "Selected empty row." TO DISPLAY

        END TRYCATCH

END PROCEDURE


PROCEDURE a_buttonNew_Click()

        SET a_textBoxTitle.Text TO ""
        SET a_textBoxAddress1.Text TO ""
        SET a_textBoxAddress2.Text TO ""
        SET a_textBoxAddress3.Text TO ""
        SET a_textBoxCity.Text TO ""
        SET a_textBoxCounty.Text TO ""
        SET a_textBoxPostCode.Text TO ""
        SET a_textBoxLink.Text TO ""

END PROCEDURE
```

```
PROCEDURE a_buttonSave_Click()

        REQUEST DATA IN Addresses TABLE IN PassVault DATABASE

        IF editingAddress == TRUE DO

                SET a_dataTable.ROWS[a_dataGridView.CURRENTROW]["Title"] TO a_textBoxTitle.Text
                ADD toHash256( a_textBoxTitle.Text) TO Addresses TABLE

                SET a_dataTable.ROWS[a_dataGridView.CURRENTROW]["Address1"] TO a_textBoxAddress1.Text
                ADD toHash256(a_textBoxAddress1.Text ) TO Addresses TABLE

                SET a_dataTable.ROWS[a_dataGridView.CURRENTROW]["Address2"] TO a_textBoxAddress2.Text
                ADD toHash256( a_textBoxAddress2.Text) TO Addresses TABLE

                SET a_dataTable.ROWS[a_dataGridView.CURRENTROW]["Address3"] TO a_textBoxAddress3.Text
                ADD toHash256(  a_textBoxAddress3.Text) TO Addresses TABLE

                SET a_dataTable.ROWS[a_dataGridView.CURRENTROW]["City/Town"] TO a_textBoxCity.Text
                ADD toHash256(a_textBoxCity.Text ) TO Addresses TABLE

                SET a_dataTable.ROWS[a_dataGridView.CURRENTROW]["County/State"] TO a_textBoxCounty.Text
                ADD toHash256( a_textBoxCounty.Text) TO Addresses TABLE

                SET a_dataTable.ROWS[a_dataGridView.CURRENTROW]["Postal Code"] TO a_textBoxPostCode.Text
                ADD toHash256( a_textBoxPostCode.Text) TO Addresses TABLE

                SET a_dataTable.ROWS[a_dataGridView.CURRENTROW]["Link"] TO a_textBoxLink.Text
                ADD toHash256( a_textBoxLink.Text) TO Addresses TABLE

        ELSE DO

                SET a_dataTable.ROWS("Title") TO a_textBoxTitle.Text
                OVERWRITE toHash256( a_textBoxTitle.Text) IN Addresses TABLE

                SET a_dataTable.ROWS("Address1") TO a_textBoxAddress1.Text
                OVERWRITE toHash256( a_textBoxAddress1.Text) IN Addresses TABLE

                SET a_dataTable.ROWS("Address2") TO a_textBoxAddress2.Text
                OVERWRITE toHash256( a_textBoxAddress2.Text) IN Addresses TABLE

                SET a_dataTable.ROWS("Address3") TO a_textBoxAddress3.Text
                OVERWRITE toHash256( a_textBoxAddress3.Text) IN Addresses TABLE

                SET a_dataTable.ROWS("City/Town") TO a_textBoxCity.Text
                OVERWRITE toHash256( a_textBoxCity.Text ) IN Addresses TABLE

                SET a_dataTable.ROWS("County/State") TO a_textBoxCounty.Text
                OVERWRITE toHash256( a_textBoxCounty.Text) IN Addresses TABLE

                SET a_dataTable.ROWS("Postal Code") TO a_textBoxPostCode.Text
                OVERWRITE toHash256( a_textBoxPostCode.Text) IN Addresses TABLE

                SET a_dataTable.ROWS("Link") TO a_textBoxLink.Text
                OVERWRITE toHash256( a_textBoxLink.Text) IN Addresses TABLE

        END IF

        SET a_textBoxTitle.Text TO ""
        SET a_textBoxAddress1.Text TO ""
        SET a_textBoxAddress2.Text TO ""
        SET a_textBoxAddress3.Text TO ""
        SET a_textBoxCity.Text TO ""
        SET a_textBoxCounty.Text TO ""
        SET a_textBoxPostCode.Text TO ""
        SET a_textBoxLink.Text TO ""
        SET editingAddress TO False

END PROCEDURE
```

Notes

FLOWCHART


Start
Load User's Previous Password Information into Data Grid
Input Title. Notes
Clicked
No
Clicked
No
Clicked
No
Clicked
Yes
Yes
Yes
Yes
Hashing
Save Hashed Item to Password Table in
Clear Text Boxes and current item
Output "Select item to load."
Output "Select item to delete."
Select Item
Select Item
Clicked away from tab?
Yes

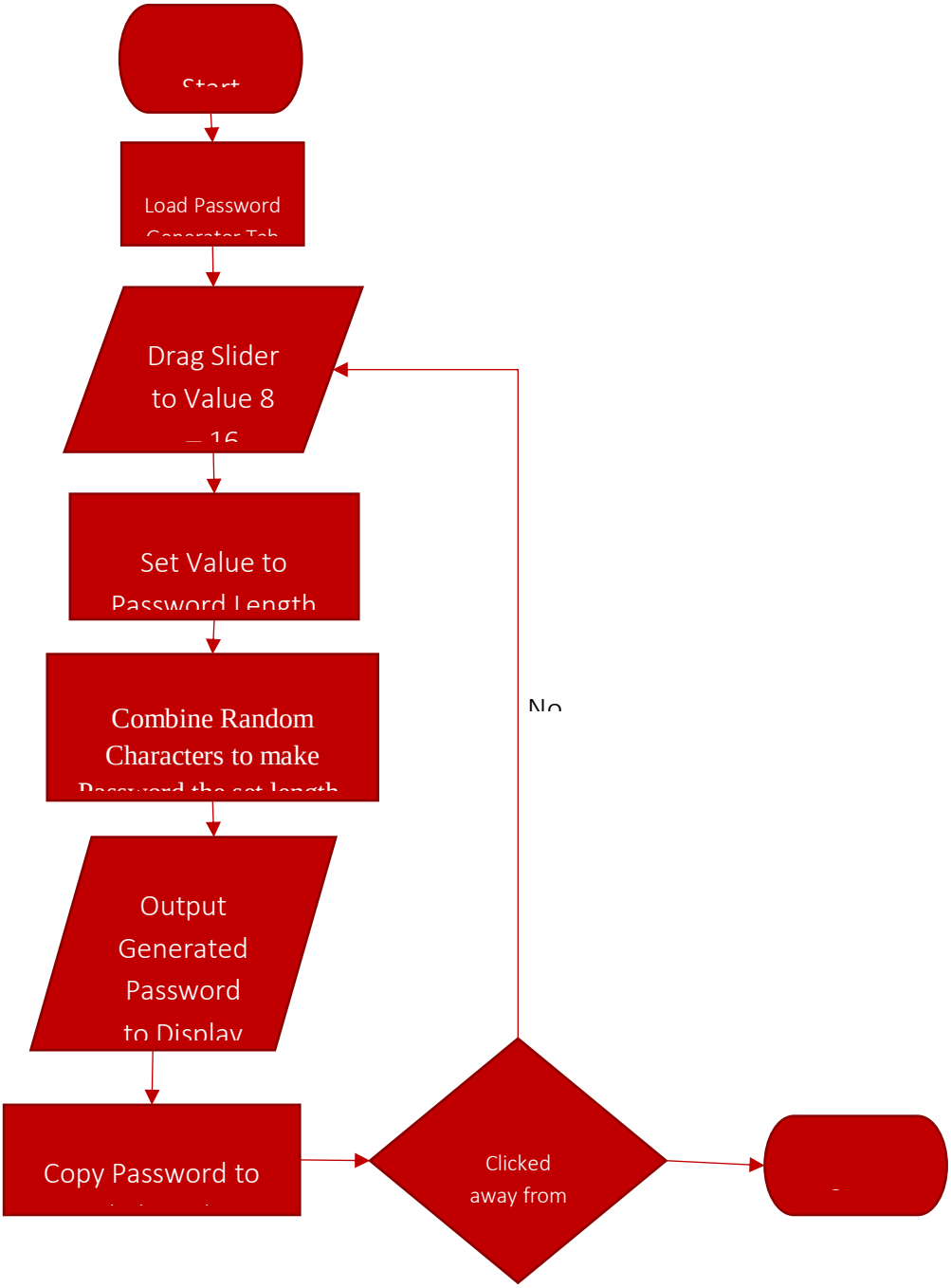
PSEUDOCODE

```
--- Notes ---


SET n_dataTable TO NEW DATATABLE

SET editingNote TO BOOL FALSE


PROCEDURE n_buttonDelete_Click()

        TRY

                DELETE n_dataTable.ROWS[n_dataGridView.CURRENTROW]

        CATCH Exception

                SEND "Select row to delete." TO DISPLAY

        END TRYCATCH

END PROCEDURE


PROCEDURE n_buttonLoad_Click OR n_dataGridView_DoubleClick()

        TRY

                SET n_textBoxTitle.Text TO n_dataTable.ROWS[n_dataGridView.CURRENTROW].ITEMARRAY[0]
                SET n_textBoxNotes.Text TO n_dataTable.ROWS[n_dataGridView.CURRENTROW].ITEMARRAY[1]
                SET editingNote TO BOOL TRUE

        CATCH Exception

                SEND "Selected empty row." TO DISPLAY

        END TRYCATCH

END PROCEDURE


PROCEDURE n_buttonNew_Click()

        SET n_textBoxTitle.Text TO ""
        SET n_textBoxNotes.Text TO ""

END PROCEDURE
```

```
PROCEDURE n_buttonSave_Click()

        REQUEST DATA IN Notes TABLE IN PassVault DATABASE

        IF editingNote == TRUE DO

                SET n_dataTable.ROWS[n_dataGridView.CURRENTROW]["Title"] TO n_textBoxTitle.Text
                ADD toHash256(n_textBoxTitle.Text) TO Notes TABLE

                SET n_dataTable.ROWS[n_dataGridView.CURRENTROW]["Notes"] TO n_textBoxNotes.Text
                ADD toHash256(n_textBoxNotes.Text) TO Notes TABLE

        ELSE DO

                SET n_dataTable.ROWS("Title") TO n_textBoxTitle.Text
                OVERWRITE toHash256(n_textBoxTitle.Text) IN Notes TABLE

                SET n_dataTable.ROWS("Notes") TO n_textBoxNotes.Text
                OVERWRITE toHash256(n_textBoxNotes.Text) IN Notes TABLE

        END IF

        SET n_textBoxTitle.Text TO ""
        SET n_textBoxNotes.Text TO ""
        SET editingNote TO False

END PROCEDURE
```

Password Generator

FLOWCHART


Start
Load Password
Drag Slider to Value 8
Set Value to
Combine Random Characters to make
Output Generated Password to Display
Copy Password to
Clicked away from
No

The slider constantly checks if it has been dragged, so it can change the generated password length, until the password generator is exited.

PSEUDOCODE

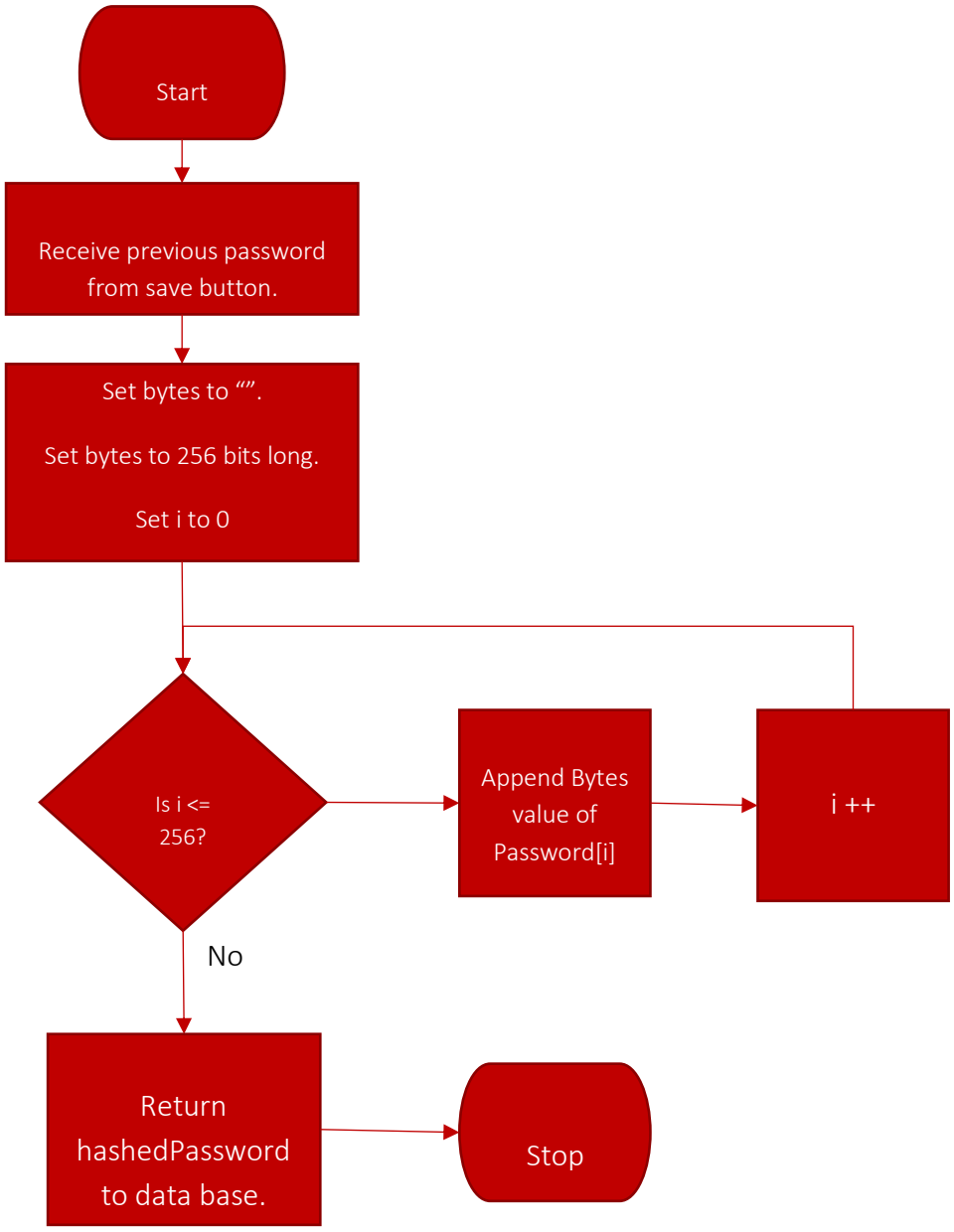
```
--- Password Generator ---


SET RandomCharacter TO NEW RANDOM

SET currentPasswordLength TO INT 0

PROCEDURE trackBarPasswordLength_Scroll()

        SET currentPasswordLength TO trackBarPasswordLength

        LOAD PROCEDURE randomPasswordGenerator(currentPasswordLength)

END PROCEDURE


PROCEDURE randomPasswordGenerator(int passwordLength)

        SET Characters TO CHAR[]

                Q','W','E','R','T','Y','U','I','O','P','A','S','D','F','G','H','J','K','L','Z',
        'X','C','V','B','N','M','q','w','e','r','t','y','u','i','o','p','a','s','d','f',
                'g','h','j','k','l','z','x','c','v','b','n','m','1','2','3','4','5','6','7','8',
        '9','0','!','£','$','%','^','&','*','(',')','_','-','+','=','{','}','[',']',':',
        ':','@','#','~',',','<','>','.','/','?'

        SET randomPassword TO STRING ""

        SET i TO INT 1

        for i < passwordLength DO

                SET randomPassword TO CHAR Characters[RandomCharacter.Next(0,90)]

                SET i TO INT i + 1

        SET labelPassword TO STRING randomPassword;

END PROCEDURE


PROCEDURE buttonCopyPassword_Click

        SET CLIPBOARD TO STRING labelPassword

END PROCEDURE
```

- Key Variables:
  - labelPassword – the label that stores the final password generated, randomPassword (labels are text that are appear on the windows form app)
  - passwordLength – is assigned the value that currentPasswordLength is set to.
  - currentPasswordLength – stores the current length of the password determined by the track bar's current position.
  - randomPassword – the password generated by the random characters is stored here.
  - randomCharacter – the random character that is selected from the array is stored here.
  - trackBarPasswordLength – stores the value attribute of the trackBar class.
  - Characters [] – stores most characters that may appear on a typical desktop keyboard. The program will randomly select one character at a time to store in 'randomCharacter' and then add randomCharacter to the randomPassword.

- Subroutines:
  - trackBarPasswordLength_Scroll()
    - This is a built-in procedure for a built-in function in Visual Studios: the trackBar, for when its 'Scroll' event occurs.
    - I can set it to be dragged from left to right.
    - I can also set it to be dragged up to 7 units across (first unit = 8 characters, last unit = 16 characters).
    - The procedure sets the dragged value to be the currentPasswordLength.
  - randomPasswordGenerator(passwordLength)
    - This procedure randomly generates the password by passing the currentPasswordLength variable through it as passwordLength.
    - Each iteration of the for loop chooses a random character from the list and adds it to randomPassword.
    - This is repeated 'passwordLength' number of times.

Hashing Algorithm

FLOW CHART



The iteration represents a for loop in the code, which loops until the password has finished hashing.

PSEUDOCODE

```
--- Hasher ---


FUNCTION toHash256(string passwordInput)

        SET sha256 TO SHA256.Create();
        SET byte[] bytes TO sha256.GetBytes(passwordInput)

        SET hashedPassword TO NEW StringBuilder();

        SET i TO 0

        FOR i < bytes.Length DO

                hashedPassword.Append(bytes[i])

        RETURN hashedPassword.ToString()

        END FOR

END FUNCTION
```

- toHash256(STRING passwordInput)
    - passwordInput is the password that the user saves when the save button is pressed.
    - The procedure is run in between the moment save is clicked and saved into the database.
    - It is meant to convert the string into hashed form with Visual Studio's built-in hashing function: SHA 256.Create and assembles the individual characters together with StringBuilder() (called hashedPassword in code) and GetBytes() (called bytes).

- passwordInput – the last password entered before the save button is clicked will be stored in this variable to be hashed before entering the database.

TEST PLAN

The code will be tested with a variety of different inputs to check if the desired output will occur. The program needs to handle different types of inputs, such as: inputted characters, button clicks, and single/double clicks for the data grid. Below are the types of inputs and their test data for the testing phase:

Login/ Sign Up

- Loading the Vault

Test Feature	Test Data
Clicking login should declare each data table in their respective tabs.	Load program

Password Types

- Entered Characters

Test Feature	Test Data
Anything entered in the text box in the vault or sign in page must be saved as string to the data grid	Passwords: - a* 40 (This is to test how long string is displayed in cells and text boxes) - Netflix - name@gmail.com

or overwritten over existing data in it.	- Pa22w0rd - https://www.youtube.com/ - https://www.outlook.office.com/login/ - https://www.netflix.com/login/ Bank Accounts: - Lloyd's Account - Lloyd Smith - Ayaan's Account - Roboy532 - annI2525 - 36641248 - 02-01-69 - https://online.lloydsbank.co.uk/personal/logon/login.jsp Cards: - Lloyd's Card - Rohan Majid - 4201 9818 3687 3309 - Debit - 111 - 752 - 10/20 - 10/25 - https://online.lloydsbank.co.uk/personal/logon/login.jsp Addresses: - School Address

	- RBHS - 6th Form - 100 Lever Park Avenue - Blackrod - Horwich - Bolton - Lancashire - BL6 7RU - https://www.google.com/maps Notes: - iPhone PIN - Watch PIN - 6962 - Wi-Fi Password - 6963 - nm4Fmrsrtvwd – for MaliksWiFi ☒Success – to indicate the test data gave the desired outcome.
The login page will need to check if the user's input matches any account in the database.	- bob@gmail.com - Pa22w0rd

- Empty String

Test Feature	Test Data

If a text box is empty in the log in page, the program needs to output an error message box telling the user to enter information.	“”
If a text box is empty in the vault, the program will enter the empty string in its corresponding cell in the data grid.	“”

- Button Clicks

Test Feature	Test Input
New – when this is clicked, the program needs to clear all text boxes, ready for input by the user.	Click one of these buttons in their corresponding tab: new password, new address, new card, new bank account and new note.
Load –	
▪ When this is clicked, the information for the selected row will be transferred from the data grid into their corresponding text boxes.	○ First, click row from one of these data grids from their corresponding tab: passwords grid, bank accounts grid, cards grid, addresses grid, notes grid. ○ Next, click one of these buttons in their corresponding tab: load password, load address,

	load card, load bank account and load note.
▪ If a row is not selected first, an error message box informs the user to do so.	Click one of these buttons in each corresponding tab: load password, load address, load card, load bank account and load note.
Save -	
○ When this is clicked, all entered information will be transferred from the text boxes to be overwritten into their corresponding data grid rows.	○ From their corresponding tab, click a row to overwrite from one of the data grids: passwords grid, bank accounts grid, cards grid, addresses grid, notes grid. ○ Click one of these buttons in their corresponding tab: save password, save address, save card, save bank account, and save note.
○ If a row isn't selected first, the next empty row is where the information is saved.	Click one of these buttons in their corresponding tab: save password, save address, save card, save bank account, and save note.
Delete –	

▪ When this is clicked, the data grid will remove that row permanently.	○ From their corresponding tab, click a row from one of the data grids: passwords grid, bank accounts grid, cards grid, addresses grid, notes grid. ○ Click one of these buttons in the corresponding tab: delete password, delete address, delete card, delete bank account, and delete note.
▪ If a row is not selected first, an error message box informs the user to do so.	Click one of these buttons in their corresponding tab: delete password, delete address, delete card, delete bank account, and delete note.

- Cell Clicks

Test Feature	Test Input
Double click - the information for the double-clicked cell will transfer the whole row's information into their corresponding text boxes.	Click grid cell for each data grid.

Single click – the program will set that as the row to be loaded or deleted, depending on the user's next input.	Click grid cell for each data grid.

Password Generator

- Sliding

Test Feature	Test Input
Drag Slider – The slider's index must declare the length of the randomly generated password.	Drag slider to every position.
Copy Password – when clicked, the generated password is copied to the clipboard.	Click button and paste password.

Hashing Algorithm

- Hashing

Test Feature	Test Input
Hash some string in the program.	Any random string from the program will be selected and the hashing function will be run for it.

Database Linkage

This will be carried out after all the other tests have finished.

- Test Data

Test data for vault.	Passwords: - a* 40 (This is to test how long string is displayed in Access cells) - Netflix - name@gmail.com - Pa22w0rd - https://www.netflix.com/login/ - YouTube - name@gmail.com - @nnI2525 - https://www.youtube.com/ - Outlook - 16krosahm@rbhs.co.uk - Bb161616 - https://www.outlook.office.com/login/ - Gmail - name@gmail.com - Tt151515 - https://mail.google.com/mail/ Bank Accounts: - Lloyd's Account - Loid532 - annI2525

	- 36641248 - 02-01-69 - https://online.lloydsbank.co.uk/personal/logon/login.jsp - Halifax Account - Hal123 - Safia1000 - 84218496 - 05-06-54 - https://www.halifax-online.co.uk/personal/logon/login.jsp - HSBC Account - Heichesbeesee456 - Ayaan123 - 33216954 - 85-66-99 - https://www.hsbc.co.uk/security/ Cards: - Lloyd's Card - Rohan Majid - 4201 9818 3687 3309 - Debit - 111 - 10/20 - 10/25 - https://online.lloydsbank.co.uk/personal/logon/login.jsp

	- Halifax Card - Hal Jordan - 8421 4916 8189 7899 - Debit - 732 - 07/21 - 07/26 - https://www.halifax-online.co.uk/personal/logon/login.jsp Addresses: - School Address (or 6th Form or RBHS) - 100 Lever Park Avenue - Blackrod - Horwich - Bolton - Lancashire - BL6 7RU - https://www.google.com/maps Notes: - iPhone PIN - 2211 - 2211

	- Wi-Fi Password - 6963 - nm4Fmrsrtvwd - for MaliksWiFi
Test data for login/ sign up page.	- bob@gmail.com - Pa22w0rd

- Button Clicks

Save -	
○ If a row isn't selected, information is saved in the next empty space. ○ The ID column must also update automatically.	○ From their corresponding tab, click a row to overwrite from one of the data grids: passwords grid, bank accounts grid, cards grid, addresses grid, notes grid. ○ Next click one of these buttons in their corresponding tab: save password, save address, save card, save bank account, and save note.
○ If a row is selected first, that is where	○ Click one of these buttons in their corresponding tab: save password, save

information is overwritten.	address, save card, save bank account, and save note.
Delete –	
When this is clicked, the data grid will remove that row permanently, in the data grid and in Access.	○ From their corresponding tab, click a row from one of the data grids: passwords grid, bank accounts grid, cards grid, addresses grid, notes grid. ○ Click one of these buttons in the corresponding tab: delete password, delete address, delete card, delete bank account, and delete note.

DEVELOPMENT & TESTING

This section documents the coding process for each section of the program and any amendments made to them. Then it will record their testing phase.

First, I will code the vault and login screen in the app development phase. With the provided test data from the previous section, the part of the code will be tested as soon as I am finished with the test data from the design section. If the desired outcome does not occur, I will terminate the testing process and repeat the coding process for that section of code. Then I will continue testing from where I left off to check it works correctly. I shall focus on the app phase before the database phase; I will program the app as if there was not meant to be a database connection to Access and it does not save data.

Once the app phase has finished, I will then do the database development phase and link it to the vault. Linking the data tables of the program to the ones in Microsoft Access is challenging because I have little experience using Access.

Each section of the code will be separated by three lines to distinguish them from each other, and each subroutine will be separated by one line.

VAULT PROTOTYPE & TESTING

Coded & Design Solution

This is a prototype of the Vault containing only the password section of the vault and a random password generator. It has no save features or login system or Access connection.

The code will be copied, and the variable names will be changed for different sections of the vault (cards, addresses, etc.). Text boxes, labels or data grid columns will be added or removed for each section depending on how many things the user can add for that section:

```
DataTable dataGridItems = new DataTable();

bool EditingState = false;

1 reference
public PassVault()
{

    InitializeComponent();

}

1 reference
private void PassVault_Load(object sender, EventArgs e) // Load App
{

    dataGridItems.Columns.Add("Title");
    dataGridItems.Columns.Add("Username");
    dataGridItems.Columns.Add("Password");
    dataGridItems.Columns.Add("Notes");

    dataGridViewItems.DataSource = dataGridItems;

}



1 reference
private void buttonNew_Click(object sender, EventArgs e) // click New +
{

    textBoxTitle.Text = ""; textBoxTwo.Text = ""; textBoxThree.Text = ""; textBoxNotes.Text = "";

}


1 reference
private void buttonLoad_Click(object sender, EventArgs e) // click Load
{

    textBoxTitle.Text = dataGridItems.Rows[dataGridViewItems.CurrentCell.RowIndex].ItemArray[0].ToString();
    textBoxTwo.Text = dataGridItems.Rows[dataGridViewItems.CurrentCell.RowIndex].ItemArray[1].ToString();
    textBoxThree.Text = dataGridItems.Rows[dataGridViewItems.CurrentCell.RowIndex].ItemArray[2].ToString();
    textBoxNotes.Text = dataGridItems.Rows[dataGridViewItems.CurrentCell.RowIndex].ItemArray[3].ToString();
    EditingState = true;

}
```

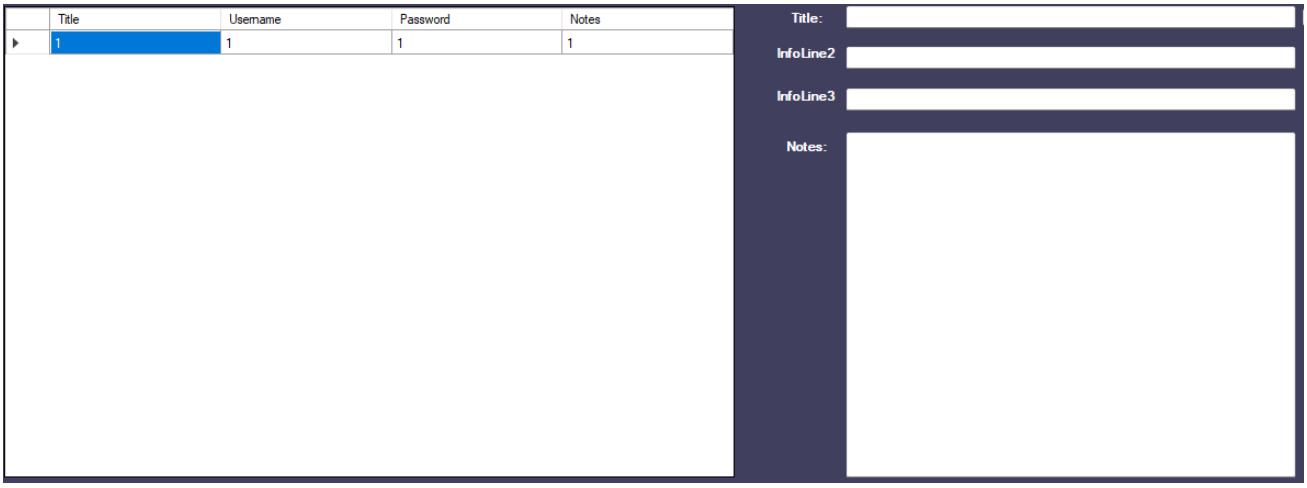
```
1 reference
private void datagridviewInfo_CellDoubleClick(object sender, DataGridViewCellEventArgs e) // Load Shortcut
{

    textBoxTitle.Text = dataGridItems.Rows[dataGridViewItems.CurrentCell.RowIndex].ItemArray[0].ToString();
    textBoxTwo.Text = dataGridItems.Rows[dataGridViewItems.CurrentCell.RowIndex].ItemArray[1].ToString();
    textBoxThree.Text = dataGridItems.Rows[dataGridViewItems.CurrentCell.RowIndex].ItemArray[2].ToString();
    textBoxNotes.Text = dataGridItems.Rows[dataGridViewItems.CurrentCell.RowIndex].ItemArray[3].ToString();
    EditingState = true;

}
```

```
1 reference
private void buttonSave_Click(object sender, EventArgs e) // Click Save
{

    if (EditingState)
    {

        dataGridItems.Rows[dataGridViewItems.CurrentCell.RowIndex]["Title"] = textBoxTitle.Text;
        dataGridItems.Rows[dataGridViewItems.CurrentCell.RowIndex]["Username"] = textBoxTwo.Text;
        dataGridItems.Rows[dataGridViewItems.CurrentCell.RowIndex]["Password"] = textBoxThree.Text;
        dataGridItems.Rows[dataGridViewItems.CurrentCell.RowIndex]["Notes"] = textBoxNotes.Text;

    }

    else
    {

        dataGridItems.Rows.Add(textBoxTitle.Text, textBoxTwo.Text, textBoxThree.Text, textBoxNotes.Text);

    }

    textBoxTitle.Text = "";
    textBoxTwo.Text = "";
    textBoxThree.Text = "";
    textBoxNotes.Text = "";
    EditingState = false;

}
```

```
1 reference
private void buttonDelete_Click(object sender, EventArgs e) // Delete
{

    try
    {

        dataGridItems.Rows[dataGridViewItems.CurrentCell.RowIndex].Delete();

    }

    catch (Exception) { Console.WriteLine("Invalid"); }

}
```

Quick Test

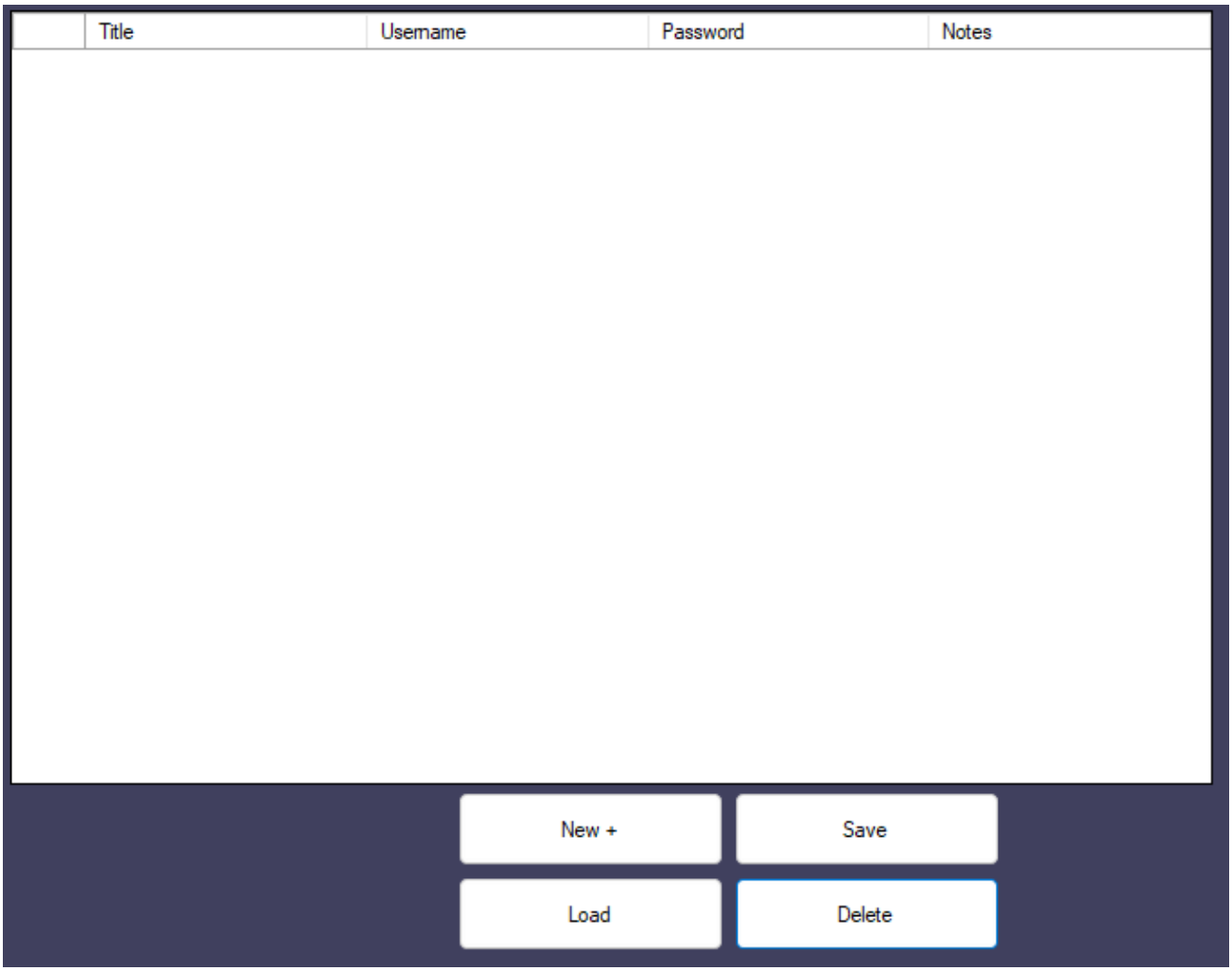
This is meant to test the most crucial features of the password manager work (saving, deleting, and loading information to the user's data grid) before extensive testing of the complete version of PassVault begins.



- The save feature appears to work here because when I entered '1' in each text box and clicked 'save', the numbers appeared in the correct rows.

- The text appears when load is clicked, or when the row is double clicked.

- The ‘New+’ button clears the textboxes as intended.



- Clicking the delete button deletes the entire row.

APP DEVELOPMENT & TESTING

Some recurring lines of code will be in the following formats converted from pseudocode to C#:

- X_dataTable.ROWS[X_dataGridView.CURRENTROW] = X_dataTable.Rows[X_dataGridView.CurrentCell.RowIndex]

- CurrentCell.RowIndex replaces CURRENTROW for identifying the row index because in C# the CurrentCell function is used first, then its row index is retrieved.

- SEND "Select X." TO DISPLAY = MessageBox.Show("Select X.")
  - In C# there are pre-built message boxes to display messages in a pop-up window onto the user's display.

- SET X_textBoxX.Text TO "" = X_dataTable.Rows.Add(X_textBoxX.Text)
  - The use of brackets allows for more text box texts to be added to the data table without writing more lines of code.
  - Each password type will have their own types of text boxes in the brackets, for example, the cards section will have these texts transferred from the text boxes to the data table.

```
c_dataTable.Rows.Add(c_textBoxTitle.Text, c_textBoxNameOnCard.Text,
c_textBoxCardNo.Text, c_textBoxCardType.Text, c_textBoxSecCode.Text,
c_textBoxStartDate.Text, c_textBoxEndDate.Text, c_textBoxLink.Text);
```

Login / Sign Up

CODED SOLUTION

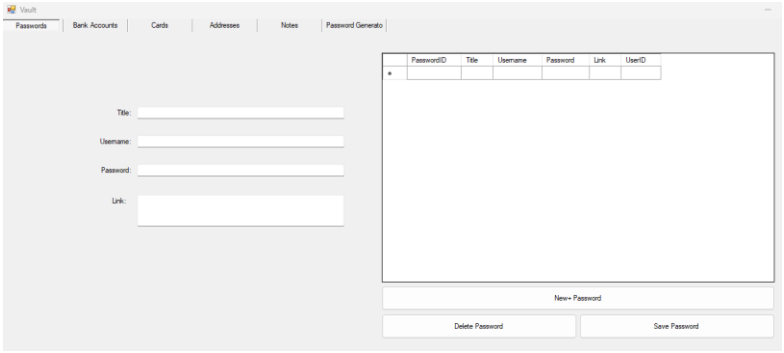
```
1 reference
private void su_buttonSignUp_Click(object sender, EventArgs e)
{
    if (su_textBoxMasterPassword.Text == "" || su_textBoxConfirmMasterPassword.Text == "" || su_textBoxMasterUsername.Text == "")
    {
        MessageBox.Show("Fill all boxes");
    }

    else if (su_textBoxMasterPassword.Text == su_textBoxConfirmMasterPassword.Text)
    {

    }

    else
    {
        MessageBox.Show("Passwords don't match.");
    }
}

1 reference
private void l_buttonLogin_Click(object sender, EventArgs e)
{
    Vault Vault = new Vault();
    Vault.Show();
}
```

The if statements check if any text boxes are empty, or if the passwords in the 'master password' box and 'confirm master password' box. It will output message boxes telling the user to input the correct data in them.

The login button loads the vault when clicked. In a later phase of coding it, I will make the program check if the login details match the user's inputs. If they do, the respective account's information will be loaded in the vault. If it is not, the program will deny access to the vault.

CODE TEST

- Loading the Vault

Test Feature	Test Data
Clicking login should declare each data table in their respective tabs.	 The vault menu loads. ☑Success

Vault

CODED SOLUTION

```
using System;
using System.Data;
using System.Windows.Forms;
using System.Security.Cryptography;
using System.Text;

namespace PassVault_Multi_Menu_Version
{
    3 references
    public partial class PassVault : Form
    {


        1 reference
        public PassVault()
        {
            InitializeComponent();
        }
```

- ‘Using System.Windows.Form’ imports Windows form functionality so the features, such as buttons, tabs and data grids can be displayed in front the user.
- ‘Public partial class ‘PassVault: Form’ declares the form for the vault, which is called ‘PassVault’.
- The ‘InitializeComponent()’ code loads the form when the program is run.

```
private void PassVault_Load(object sender, EventArgs e)
{
    p_dataTable.Columns.Add("Title"); p_dataTable.Columns.Add("Username"); p_dataTable.Columns.Add("Password"); p_dataTable.Columns.Add("Link");
    p_dataGridView.DataSource = p_dataTable;

    ba_dataTable.Columns.Add("Title"); ba_dataTable.Columns.Add("Username"); ba_dataTable.Columns.Add("Password");
    ba_dataTable.Columns.Add("Account No."); ba_dataTable.Columns.Add("Sort Code"); ba_dataTable.Columns.Add("Link");
    ba_dataGridView.DataSource = ba_dataTable;

    c_dataTable.Columns.Add("Title"); c_dataTable.Columns.Add("Name on Card"); c_dataTable.Columns.Add("Card No."); c_dataTable.Columns.Add("Card Type")
    c_dataTable.Columns.Add("Sec. Code"); c_dataTable.Columns.Add("Start Date"); c_dataTable.Columns.Add("End Date"); c_dataTable.Columns.Add("Link");
    c_dataGridView.DataSource = c_dataTable;

    a_dataTable.Columns.Add("Title"); a_dataTable.Columns.Add("Address 1"); a_dataTable.Columns.Add("Address 2"); a_dataTable.Columns.Add("Address 3");
    a_dataTable.Columns.Add("City/Town"); a_dataTable.Columns.Add("County"); a_dataTable.Columns.Add("Post Code"); a_dataTable.Columns.Add("Link");
    a_dataGridView.DataSource = a_dataTable;

    n_dataTable.Columns.Add("Title"); n_dataTable.Columns.Add("Notes");
    n_dataGridView.DataSource = n_dataTable;
}
```

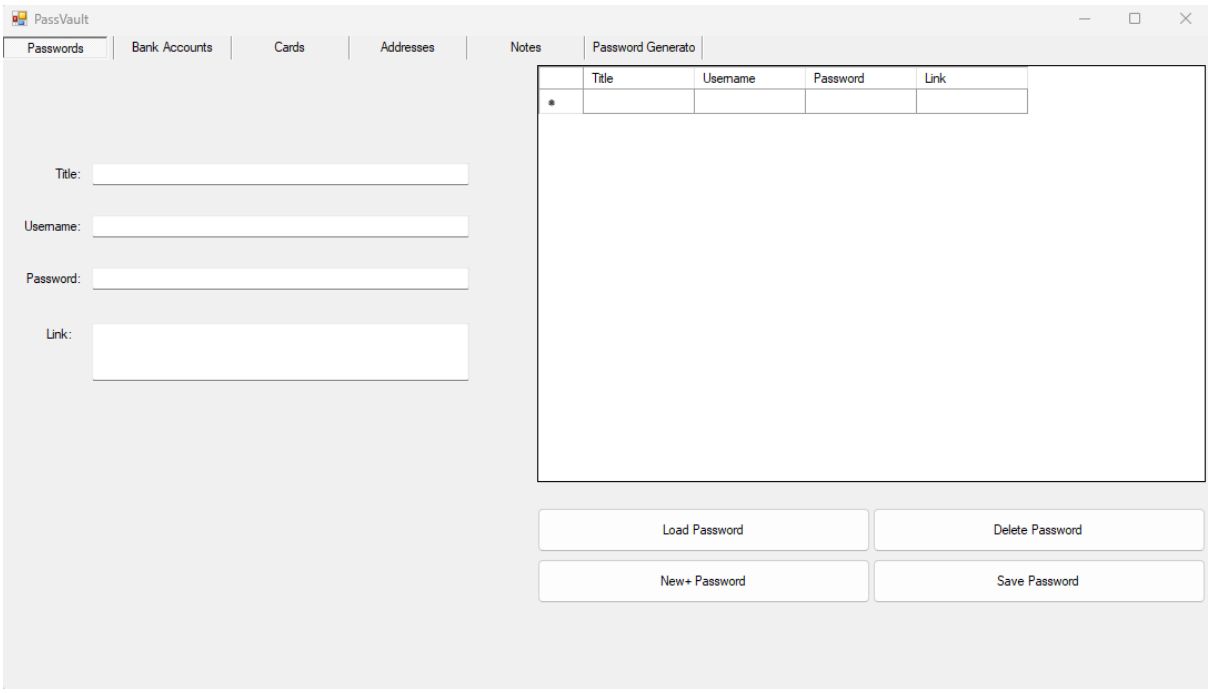
- The naming convention will be the same as the one from the pseudocode to tell which items are for what tab with the shortest possible variable names.
- The data table columns must be declared before they can be used in any other part of the code, so the ‘PassVault_Load’ procedure is run every time the program is run. Each data table will be displayed in its respective tab.

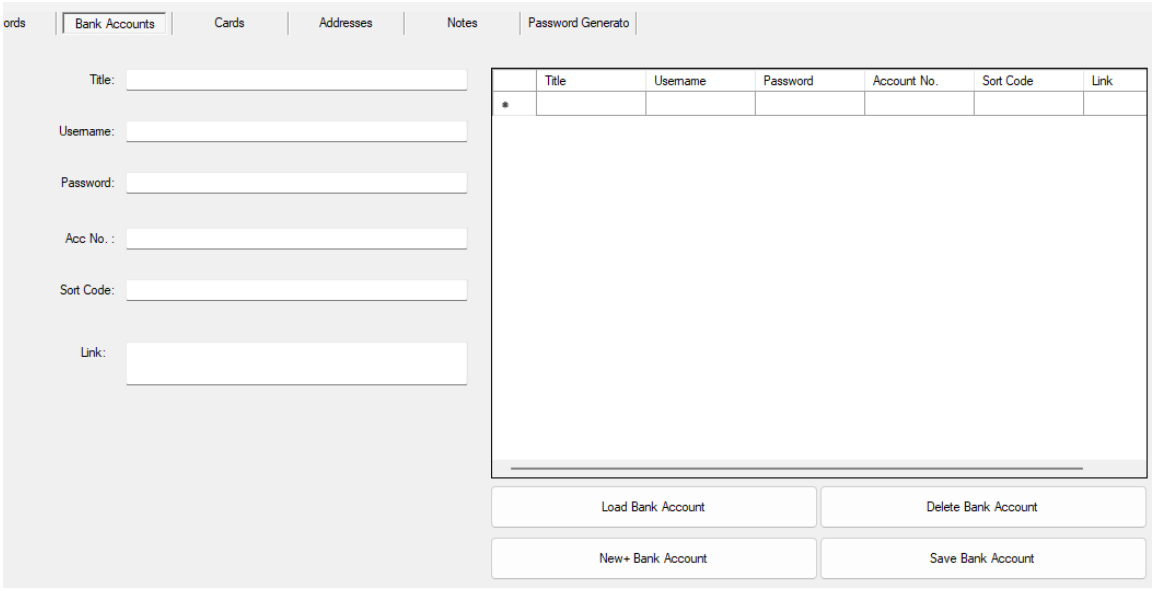
- Each column is the type of information the user can input and will be added to their data tables when the program is loaded.

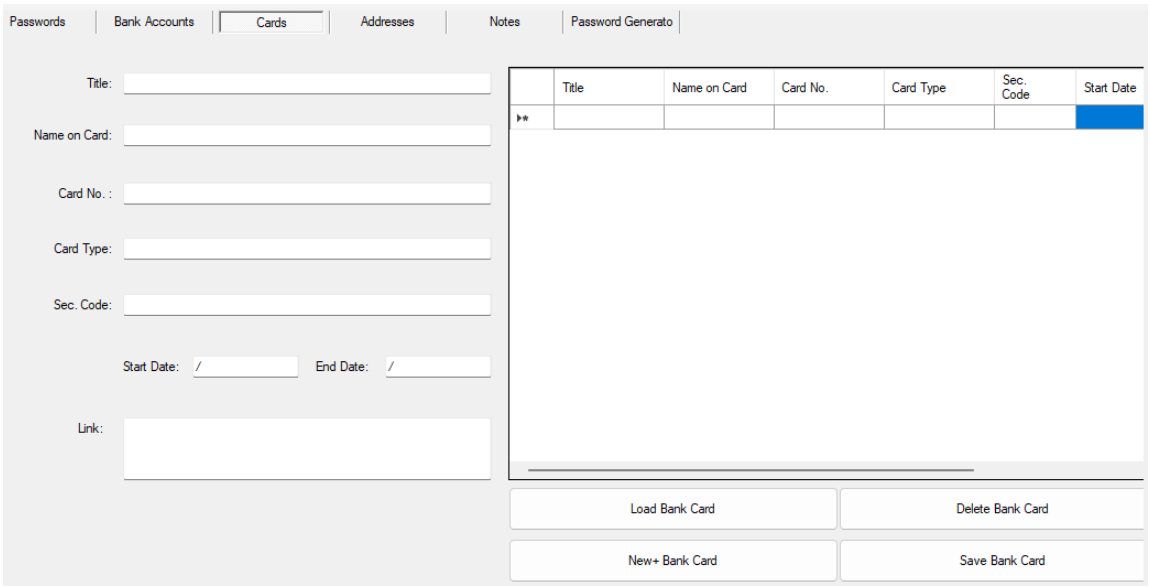
CODE TEST

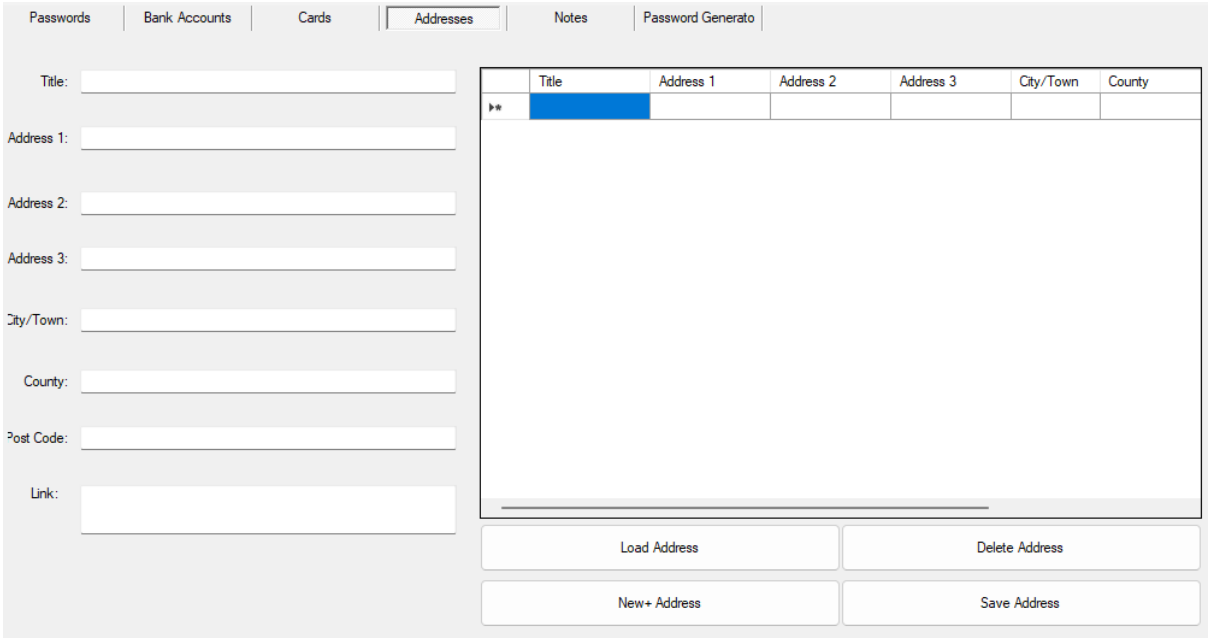
- Loading the Vault

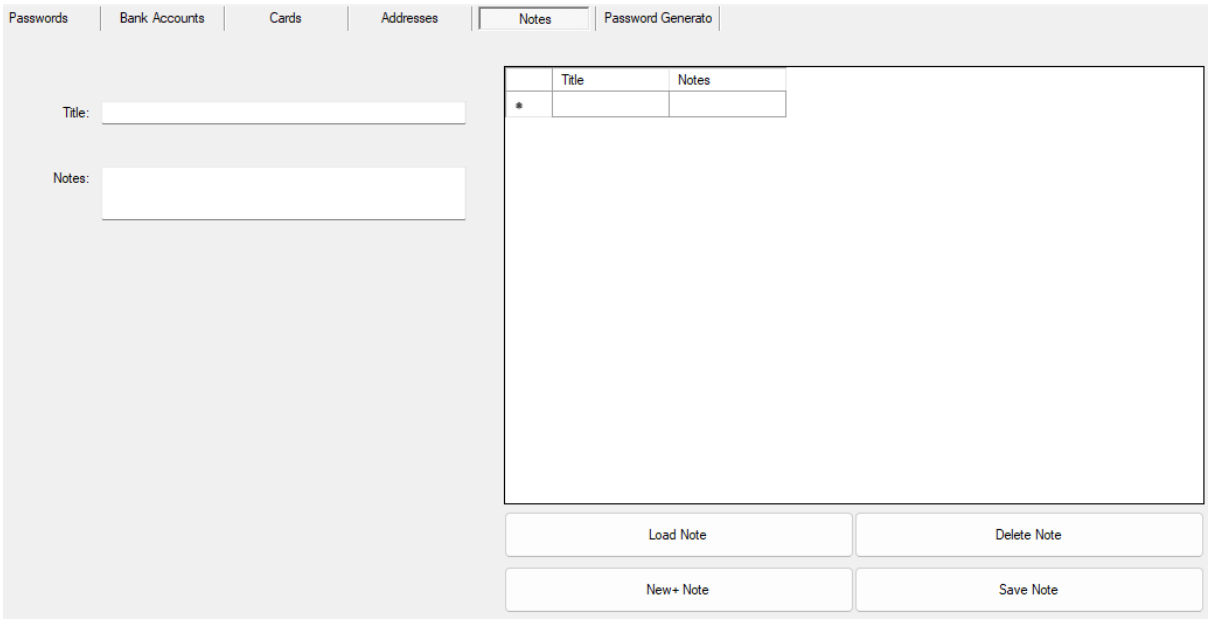
Test Feature	Test Data Result
Running the program should declare each data table in their respective tabs.	All data tables are shown in the tabs below:




ords
Bank Accounts
Cards
Addresses
Notes
Password Generato
Title:
Username:
Password:
Acc No. :
Sort Code:
Link:
Title
Username
Password
Account No.
Sort Code
Link
Load Bank Account
Delete Bank Account
New+ Bank Account
Save Bank Account


Passwords
Bank Accounts
Cards
Addresses
Notes
Password Generato
Title:
Name on Card:
Card No. :
Card Type:
Sec. Code:
Start Date:
End Date:
Link:
Title
Name on Card
Card No.
Card Type
Sec. Code
Start Date
Load Bank Card
Delete Bank Card
New+ Bank Card
Save Bank Card





Passwords

CODED SOLUTION 1

```
// for passwords
DataTable p_dataTable = new DataTable();
bool editingPassword = false;
```

- The data table for passwords is declared here first. Then its rows are declared in the data grid in the PassVault_Load() class.
- 'The editingPassword' variable determines whether the inputted information is to be overwritten.

```
1 reference
private void p_buttonDelete_Click(object sender, EventArgs e)
{
    try { p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].Delete(); }
    catch (Exception) { MessageBox.Show("Select row to delete."); }
}
```

```
0 references
private void p_buttonLoad_Click(object sender, EventArgs e)
{
    try
    {
        p_textBoxTitle.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[0].ToString();
        p_textBoxUsername.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
        p_textBoxPassword.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[2].ToString();
        p_textBoxLink.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[3].ToString();
        editingPassword = true;
    }
    catch (Exception) { MessageBox.Show("Selected empty row."); }
}
```

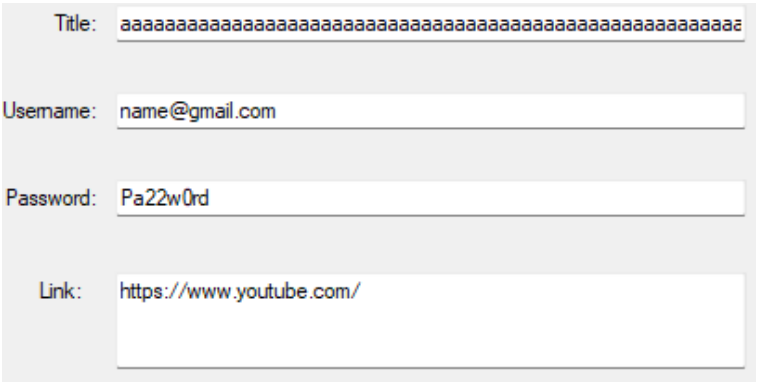
```
1 reference
private void p_dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        p_textBoxTitle.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[0].ToString();
        p_textBoxUsername.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
        p_textBoxPassword.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[2].ToString();
        p_textBoxLink.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[3].ToString();
        editingPassword = true;
    }
    catch (Exception) { MessageBox.Show("Selected empty row."); }
}
```

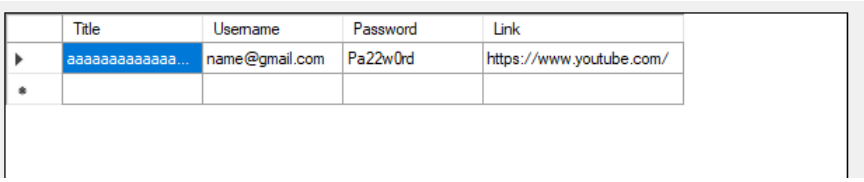
```
1 reference
private void p_buttonNew_Click(object sender, EventArgs e)
{
    p_textBoxTitle.Text = ""; p_textBoxUsername.Text = ""; p_textBoxPassword.Text = ""; p_textBoxLink.Text = "";
}
```

```
1 reference
private void p_buttonSave_Click(object sender, EventArgs e)
{
    if (editingPassword == true)
    {
        p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex]["Title"] = p_textBoxTitle.Text;
        p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex]["Username"] = p_textBoxUsername.Text;
        p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex]["Password"] = p_textBoxPassword.Text;
        p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex]["Link"] = p_textBoxLink.Text;
    }
    else
    {
        p_dataTable.Rows.Add(p_textBoxTitle.Text, p_textBoxUsername.Text, p_textBoxPassword.Text, p_textBoxLink.Text);
    }
    p_textBoxTitle.Text = ""; p_textBoxUsername.Text = ""; p_textBoxPassword.Text = ""; p_textBoxLink.Text = "";
    editingPassword = false;
}
```

CODE TEST 1

- Entered Characters

Test Feature	Test Data Result
Anything entered in the text box in the vault or sign in page must be saved as string to the data grid or overwritten over existing data in it.	Saving a new item: ✓Success  ○ I can enter text in the boxes, especially long text, which the text box can support.

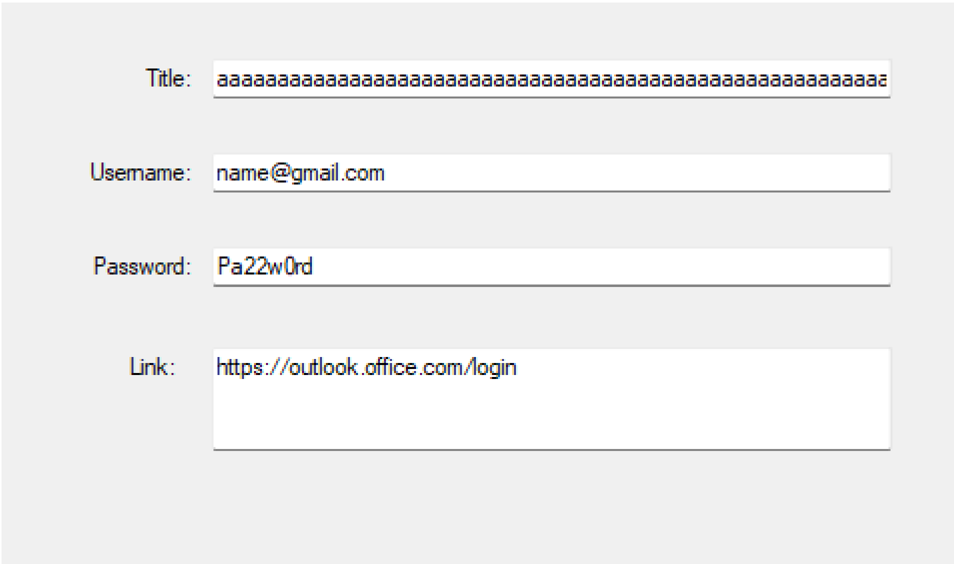


	Title	Username	Password	Link
▶	aaaaaaaaaaaaa...	name@gmail.com	Pa22w0rd	https://www.youtube.com/
*				

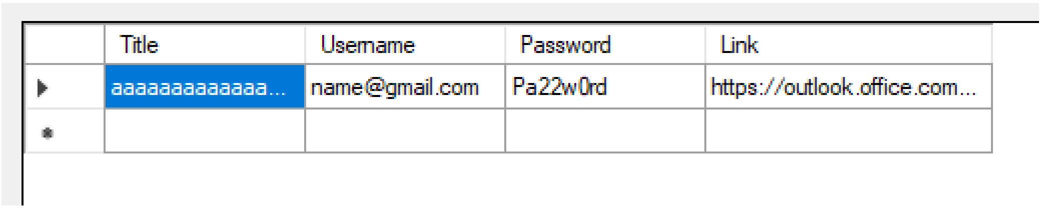
- The data grid has the function to adjust the width of the columns to show longer texts.

- The text disappears after the save button is clicked.

Overwriting an existing item: ✓Success



- A change has been made to the password by changing the link used to launch the login page.

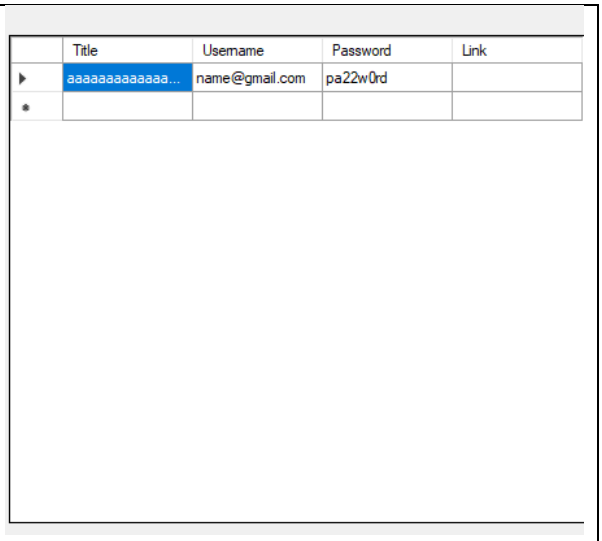


- The data grid records the change when the save button is clicked.

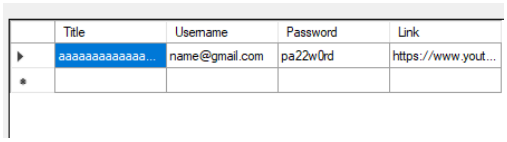
	○ The text disappears after the save button is clicked.
The login page will need to check if the user's input matches any account in the database.	

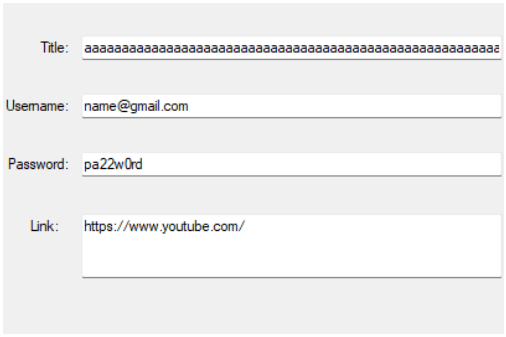
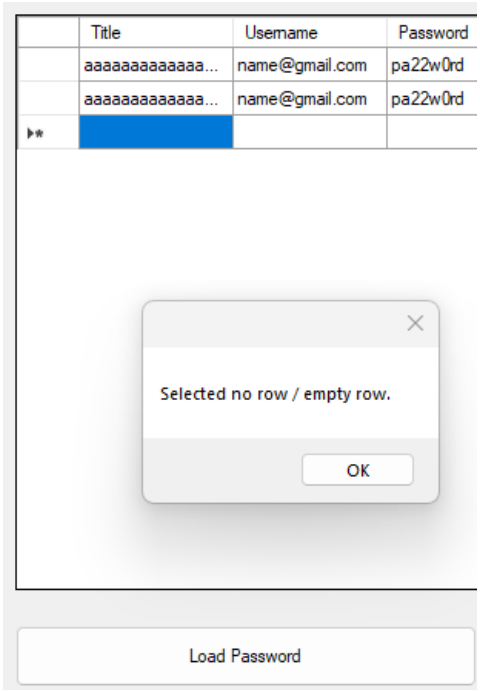
○ Empty String

Test Feature	Test Data Result
If a text box is empty in the log in page, the program needs to output an error message box telling the user to enter information.	

If a text box is empty in the vault, the program will enter the empty string in its corresponding cell in the data grid.	 ✓Success

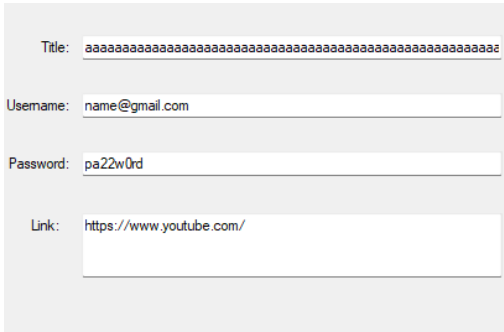
- Button Clicks

Test Feature	Test Input Result
New – when this is clicked, the program needs to clear all text boxes, ready for input by the user.	  ✓Success
Load –	

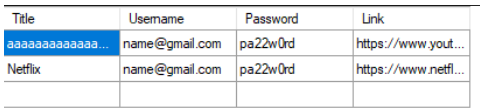
▪ When this is clicked, the information for the selected row will be transferred from the data grid into their corresponding text boxes.	 ✓Success
▪ If a row is not selected first, or if the bottom empty row is selected, an error message box informs the user to do so.	 ○ ‘No row / empty row’ is shown once load is clicked. ✓Success
Save -	

○ When this is clicked, all entered information will be transferred from the text boxes to be overwritten into their corresponding data grid rows.	 ○ A new password for Outlook is saved when new information is inputted into the boxes. ✓Success
○ If a row isn't selected first, the next empty row is where the information is saved.	 ○ The Netflix account is saved into the next available space. ○ A new empty row is generated as the next default save space. ○ This was not intended but it is convenient because it indicates the user can input more passwords if they want to. ✓Success
Delete –	
▪ When this is clicked, the data grid will remove that row permanently.	
▪ If a row is not selected first, an error message box informs the user to do so.	

- Cell Clicks

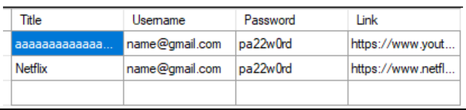
<table>
<tr><th>Test Feature</th><th>Test Input Result</th></tr>
<tr><td>Double click - the information for the double-clicked cell will transfer the whole row's information into their corresponding text boxes.</td><td> 
</td></tr>
<tr><td>Single click – The program will set that as the row to be loaded or deleted, depending on the user's next input.</td><td> 

- The Outlook account is selected.



- The delete button is then clicked, and the Outlook account disappears.

✓Success


</td></tr>
</table>

	○ The next selected account is the YouTube account.  ○ When the YouTube account is single clicked, then load is clicked, the information does not load in the boxes. ☒Success

CODED SOLUTION 2

After several trials, I could not identify why the load button was not working as it was taking up too much time. Since the double click procedure also loads a password to the textboxes and the load button for the password tab does not function correctly, I will simply remove the load button. To maintain consistency throughout my code, I will remove the load buttons for the other tabs because the program can still function correctly without it.

```
// for passwords
DataTable p_dataTable = new DataTable();
bool editingPassword = false;

1 reference
private void p_buttonDelete_Click(object sender, EventArgs e)
{
    try { p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].Delete(); }
    catch (Exception) { MessageBox.Show("Select row to delete."); }
}

1 reference
private void p_dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        p_textBoxTitle.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[0].ToString();
        p_textBoxUsername.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
        p_textBoxPassword.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[2].ToString();
        p_textBoxLink.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[3].ToString();
        editingPassword = true;
    }
    catch (Exception) { MessageBox.Show("Selected no row / empty row."); }
}

1 reference
private void p_buttonNew_Click(object sender, EventArgs e)
{
    p_textBoxTitle.Text = ""; p_textBoxUsername.Text = ""; p_textBoxPassword.Text = ""; p_textBoxLink.Text = "";
}

1 reference
private void p_buttonSave_Click(object sender, EventArgs e)
{
    if (editingPassword == true)
    {
        p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex]["Title"] = p_textBoxTitle.Text;
        p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex]["Username"] = p_textBoxUsername.Text;
        p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex]["Password"] = p_textBoxPassword.Text;
        p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex]["Link"] = p_textBoxLink.Text;
    }
    else
    {
        p_dataTable.Rows.Add(p_textBoxTitle.Text, p_textBoxUsername.Text, p_textBoxPassword.Text, p_textBoxLink.Text);
    }
    p_textBoxTitle.Text = ""; p_textBoxUsername.Text = ""; p_textBoxPassword.Text = ""; p_textBoxLink.Text = "";
    editingPassword = false;
}
```

Bank Accounts

CODED SOLUTION

```
// for bank account
DataTable ba_dataTable = new DataTable();
bool editingBankAccount = false;

1 reference
private void ba_buttonDelete_Click(object sender, EventArgs e)
{
    try { ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].Delete(); }
    catch (Exception) { MessageBox.Show("Select row to delete."); }
}
```

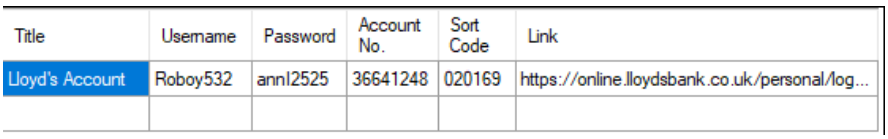
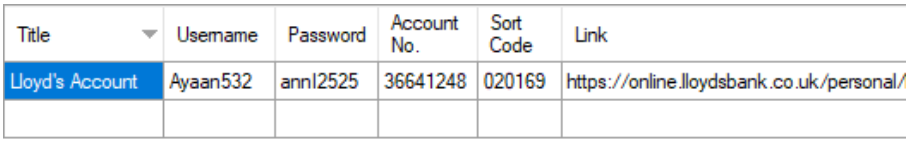
```
1 reference
private void ba_dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        ba_textBoxTitle.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[0].ToString();
        ba_textBoxUsername.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
        ba_textBoxPassword.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[2].ToString();
        ba_textBoxAccNo.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[3].ToString();
        ba_textBoxSortCode.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[4].ToString();
        ba_textBoxLink.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[5].ToString();
        editingBankAccount = true;
    }
    catch (Exception) { MessageBox.Show("Selected empty row."); }
}
```

```
1 reference
private void ba_buttonNew_Click(object sender, EventArgs e)
{
    ba_textBoxTitle.Text = ""; ba_textBoxUsername.Text = ""; ba_textBoxPassword.Text = "";
    ba_textBoxAccNo.Text = ""; ba_textBoxSortCode.Text = ""; ba_textBoxLink.Text = "";
}

1 reference
private void ba_buttonSave_Click(object sender, EventArgs e)
{
    if (editingBankAccount == true)
    {
        ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex]["Title"] = ba_textBoxTitle.Text;
        ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex]["Username"] = ba_textBoxUsername.Text;
        ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex]["Password"] = ba_textBoxPassword.Text;
        ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex]["Account No."] = ba_textBoxAccNo.Text;
        ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex]["Sort Code"] = ba_textBoxSortCode.Text;
        ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex]["Link"] = ba_textBoxLink.Text;
    }
    else
    {
        ba_dataTable.Rows.Add(ba_textBoxTitle.Text, ba_textBoxUsername.Text, ba_textBoxPassword.Text,
        ba_textBoxAccNo.Text, ba_textBoxSortCode.Text, ba_textBoxLink.Text);
    }
    ba_textBoxTitle.Text = ""; ba_textBoxUsername.Text = ""; ba_textBoxPassword.Text = "";
    ba_textBoxAccNo.Text = ""; ba_textBoxSortCode.Text = ""; ba_textBoxLink.Text = "";
    editingBankAccount = false;
}
```

CODE TEST

- Entered Characters

Test Feature	Test Data Result

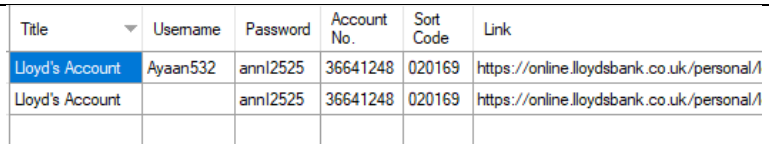
Anything entered in the text box in the vault or sign in page will need to be saved as string to the data grid or data table or overwritten over existing data in them.	Saving a new item: ☑Success  Overwriting an item (username changed to Ayaan532): ☑Success. 
The login page will need to check if the user's input matches any account in the database.	

Title	Username	Password	Account No.	Sort Code	Link
Lloyd's Account	Roboy532	annl2525	36641248	020169	https://online.lloydsbank.co.uk/personal/log...

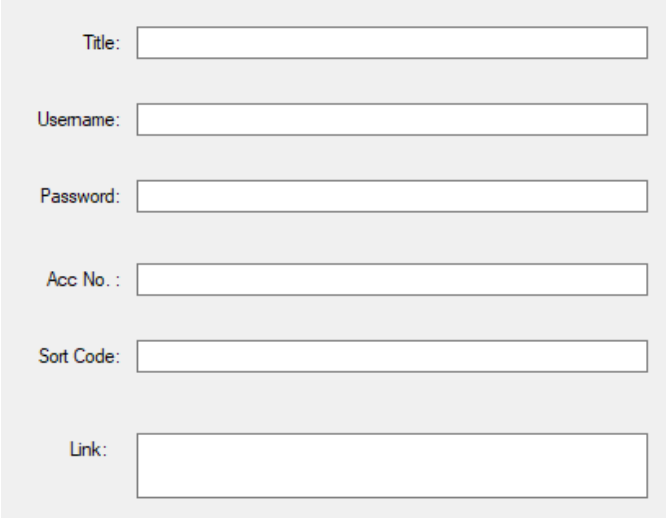
Title	Username	Password	Account No.	Sort Code	Link
Lloyd's Account	Ayaan532	annl2525	36641248	020169	https://online.lloydsbank.co.uk/personal/

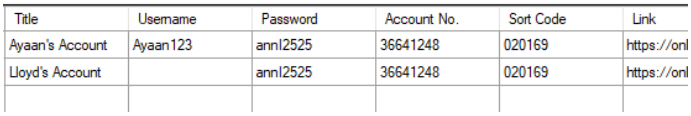
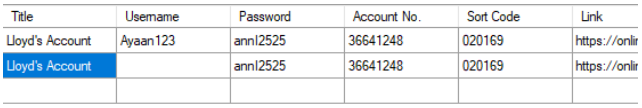
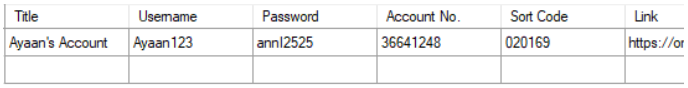
- Empty String

Test Feature	Test Data Result

If a text box is empty in the log in page, the program needs to output an error message box telling the user to enter information.	
If a text box is empty in the vault, the program will enter the empty string in its corresponding cell in the data grid.	 ☑Success

- Button Clicks

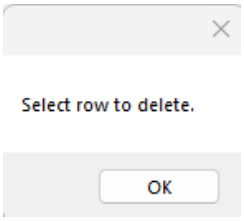
Test Feature	Test Input Result
New – when this is clicked, the program needs to clear all text boxes, ready for input by the user.	 ☑Success

Save -	
○ When this is clicked, all entered information will be transferred from the text boxes to be overwritten into their corresponding data grid rows.	 ○ 'Ayaan's Account' replaces 'Lloyd's Account'. ☑Success
○ If a row isn't selected first, the next empty row is where the information is saved.	 ☑Success
Delete –	
▪ When this is clicked, the data grid will remove that row permanently.	 ○ 'Lloyd's Account' was double clicked and then deleted. ☑Success

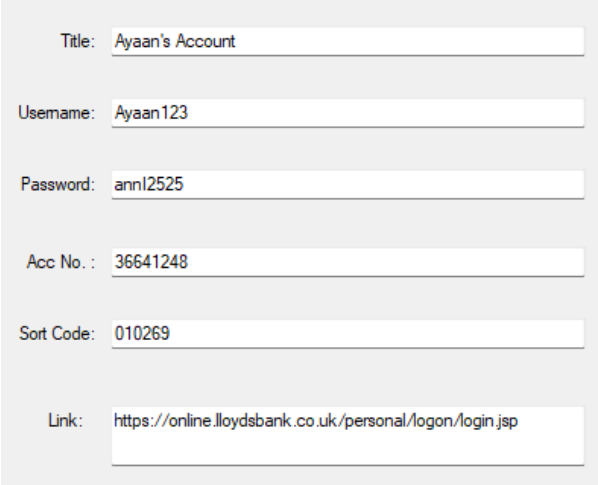
Title	Username	Password	Account No.	Sort Code	Link
Ayaan's Account	Ayaan123	annl2525	36641248	020169	https://onl
Lloyd's Account		annl2525	36641248	020169	https://onl

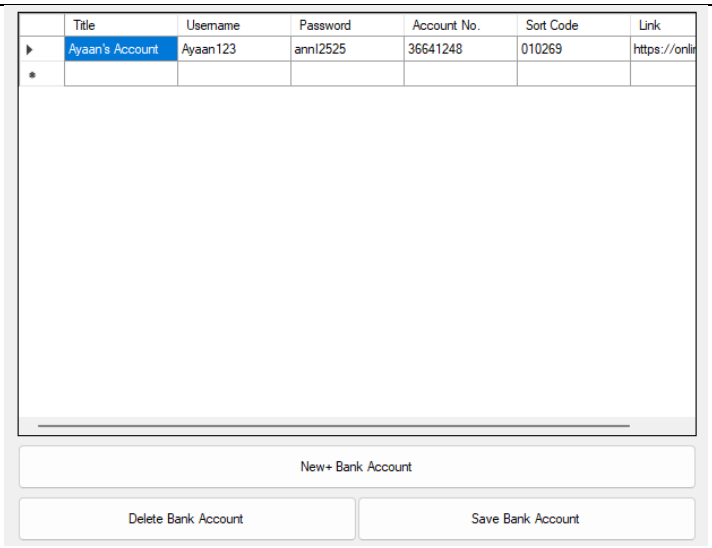
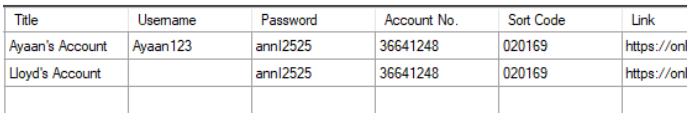
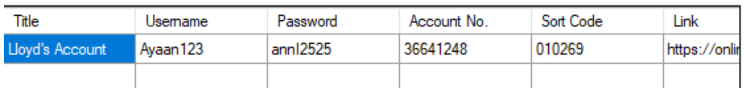
Title	Username	Password	Account No.	Sort Code	Link
Lloyd's Account	Ayaan123	annl2525	36641248	020169	https://onlir
Lloyd's Account		annl2525	36641248	020169	https://onlir

Title	Username	Password	Account No.	Sort Code	Link
Ayaan's Account	Ayaan123	annl2525	36641248	020169	https://or

▪ If a row isn't selected first, an error message box informs the user to do so.	 ☑Success

- Cell Clicks

Test Feature	Test Input Result
Double click - the information for the double-clicked cell will transfer the whole row's information into their corresponding text boxes.	

	 ☑Success
Single click – The program will set that as the row to be overwritten or deleted, depending on the user's next input.	 o Ayaan's account is selected initially. ☑Success  o Delete button is clicked and Ayaan's Account is deleted. ☑Success

Cards

CODED SOLUTION 1

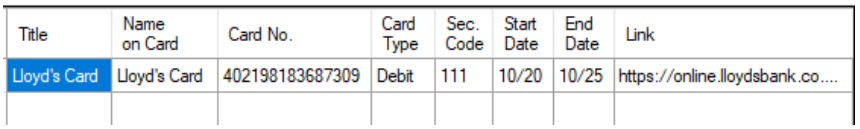
```
// for cards
DataTable c_dataTable = new DataTable();
bool editingCard = false;

1 reference
private void c_buttonDelete_Click(object sender, EventArgs e)
{
    try { c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].Delete(); }
    catch (Exception) { MessageBox.Show("Select row to delete."); }
}
```

```
1 reference
private void c_dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        c_textBoxTitle.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[0].ToString();
        c_textBoxNameOnCard.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
        c_textBoxCardNo.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[2].ToString();
        c_textBoxCardType.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[3].ToString();
        c_textBoxSecCode.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[4].ToString();
        c_textBoxStartDate.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[5].ToString();
        c_textBoxEndDate.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[6].ToString();
        c_textBoxLink.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[7].ToString();
        editingCard = true;
    }
    catch (Exception) { MessageBox.Show("Selected empty row."); }
}
```

```
1 reference
private void c_buttonNew_Click(object sender, EventArgs e)
{
    c_textBoxTitle.Text = ""; c_textBoxNameOnCard.Text = ""; c_textBoxCardNo.Text = ""; c_textBoxCardType.Text = "";
    c_textBoxSecCode.Text = ""; c_textBoxStartDate.Text = ""; c_textBoxEndDate.Text = ""; c_textBoxLink.Text = "";
}

1 reference
private void c_buttonSave_Click(object sender, EventArgs e)
{
    if (editingCard == true)
    {
        c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex]["Title"] = c_textBoxTitle.Text;
        c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex]["Name on Card"] = c_textBoxTitle.Text;
        c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex]["Card No."] = c_textBoxCardNo.Text;
        c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex]["Card Type"] = c_textBoxCardType.Text;
        c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex]["Sec. Code"] = c_textBoxSecCode.Text;
        c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex]["Start Date"] = c_textBoxStartDate.Text;
        c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex]["End Date"] = c_textBoxEndDate.Text;
        c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex]["Link"] = c_textBoxLink.Text;
    }
    else
    {
        c_dataTable.Rows.Add(c_textBoxTitle.Text, c_textBoxNameOnCard.Text,
        c_textBoxCardNo.Text, c_textBoxCardType.Text, c_textBoxSecCode.Text,
        c_textBoxStartDate.Text, c_textBoxEndDate.Text, c_textBoxLink.Text);
    }
    c_textBoxTitle.Text = ""; c_textBoxNameOnCard.Text = ""; c_textBoxCardNo.Text = ""; c_textBoxCardType.Text = "";
    c_textBoxSecCode.Text = ""; c_textBoxStartDate.Text = ""; c_textBoxEndDate.Text = ""; c_textBoxLink.Text = "";
    editingCard = false;
}
```

CODE TEST 1

- Entered Characters

Test Feature	Test Data Result
Anything entered in the text box in the vault or sign in page must be saved as string to the data grid or overwritten over existing data in it.	Saving a new item: ☒Success  Title | Name on Card | Card No. | Card Type | Sec. Code | Start Date | End Date | Link Lloyd's Card | Lloyd's Card | 402198183687309 | Debit | 111 | 10/20 | 10/25 | https://online.lloydsbank.co.... - "Rohan Majid" was entered as test data for the 'Name on Card' column, yet the program saved 'Lloyd's Card' to it.

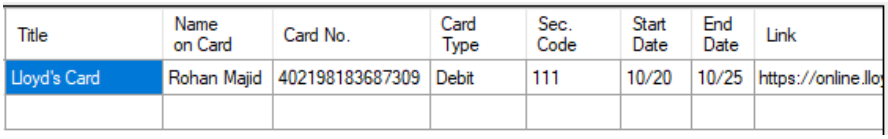
CODED SOLUTION 2

```
1 reference
private void c_buttonSave_Click(object sender, EventArgs e)
{
    if (editingCard == true)
    {
        c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex]["Title"] = c_textBoxTitle.Text;
        c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex]["Name on Card"] = c_textBoxNameOnCard.Text;
```

There was an error in the code that showed 'c_textBoxTitle.Text' instead of the variable 'c_textBoxNameOnCard.Text'. This caused the input for the title to be saved under the 'Name on Card' column.

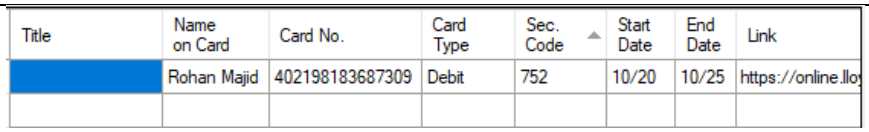
CODE TEST 2

- Entered Characters

<table>
<tr><td>Test Feature</td><td>Test Data Result</td></tr>
<tr><td>Anything entered in the text box in the vault or sign in page must be saved as string to the data grid or overwritten over existing data in it.</td><td>Saving a new item: ☑Success
 
<table>
<tr><th>Title</th><th>Name on Card</th><th>Card No.</th><th>Card Type</th><th>Sec. Code</th><th>Start Date</th><th>End Date</th><th>Link</th></tr>
<tr><td>Lloyd's Card</td><td>Rohan Majid</td><td>402198183687309</td><td>Debit</td><td>111</td><td>10/20</td><td>10/25</td><td>https://online.lloy</td></tr>
</table>
“Rohan Majid” was entered as test data for the ‘Name on Card’ column, yet the program saved ‘Lloyd’s Card’ to it.

<table>
<tr><th>Title</th><th>Name on Card</th><th>Card No.</th><th>Card Type</th><th>Sec. Code</th><th>Start Date</th><th>End Date</th><th>Link</th></tr>
<tr><td>Lloyd's Card</td><td>Rohan Majid</td><td>402198183687309</td><td>Debit</td><td>752</td><td>10/20</td><td>10/25</td><td>https://online.llo</td></tr>
</table>
‘752’ overwrote ‘111’.
Overwriting a new item: ☑Success</td></tr>
<tr><td>The login page will need to check if the user’s input matches any account in the database.</td><td></td></tr>
</table>

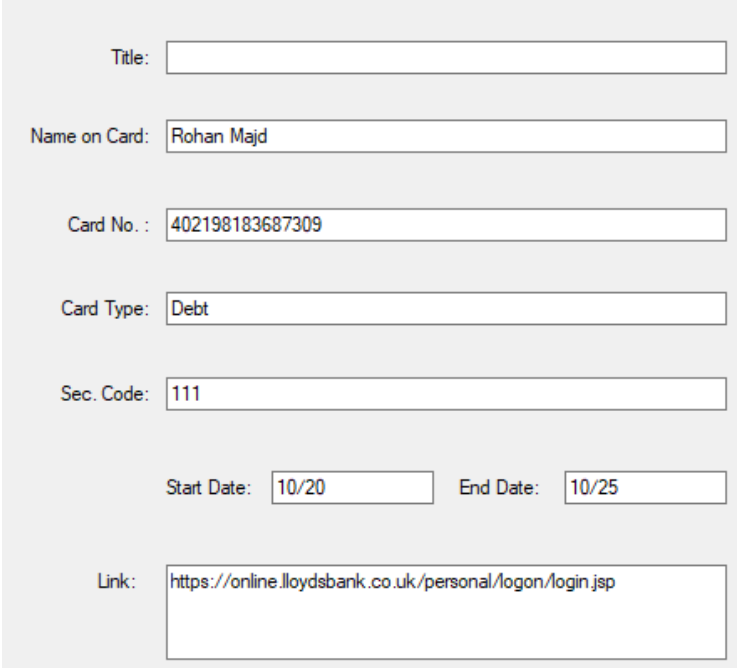
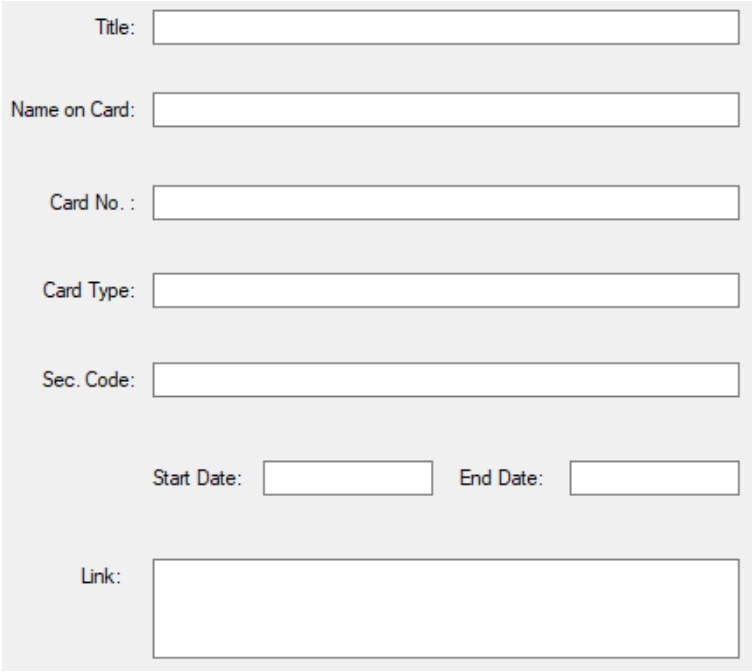
- Empty String

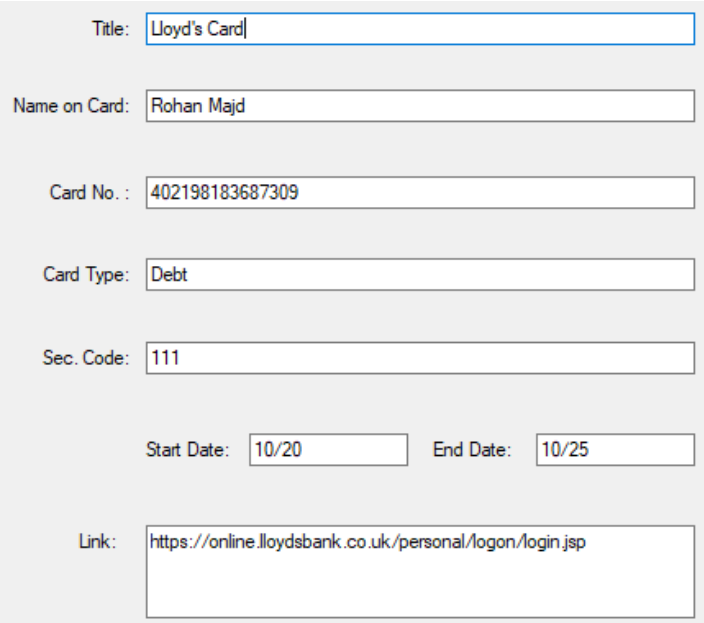
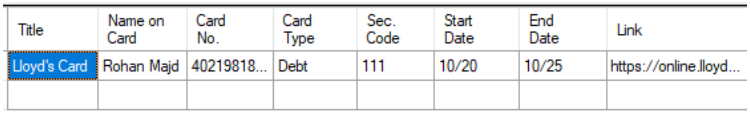
Test Feature	Test Data Result
If a text box is empty in the log in page, the program needs to output an error message box telling the user to enter information.	
If a text box is empty in the vault, the program will enter the empty string in its corresponding cell in the data grid.	 ☑Success

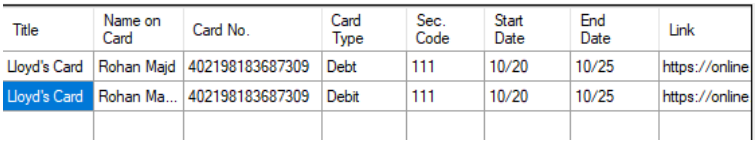
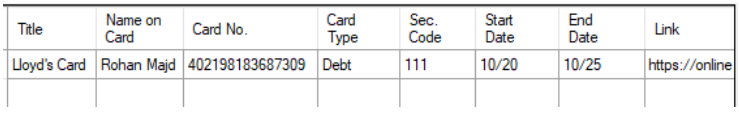
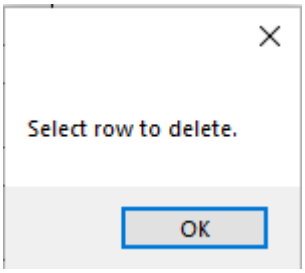
Title	Name on Card	Card No.	Card Type	Sec. Code	Start Date	End Date	Link
	Rohan Majid	402198183687309	Debit	752	10/20	10/25	https://online.lloy

- Button Clicks

Test Feature	Test Input Result
New – when this is clicked, the program needs to clear all text	Before clicking:

boxes, ready for input by the user.	 ☑Success After clicking:  ☑Success

Save -	
○ When this is clicked, all entered information will be transferred from the text boxes to be overwritten into their corresponding data grid rows.	 ○ The first item is loaded. Then 'Lloyd's Card' is inputted. ☑Success  ○ When save is clicked, the input is saved with that password. ☑Success

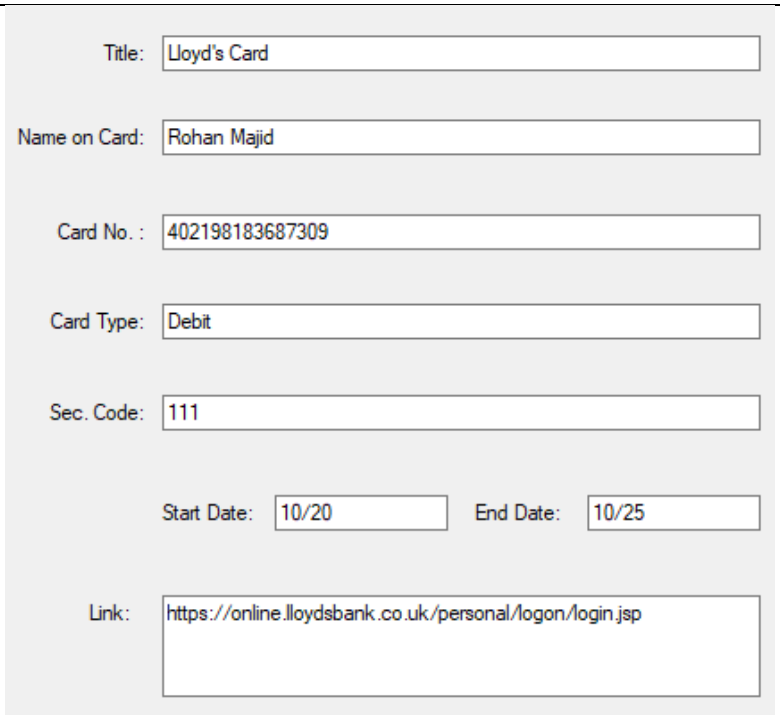
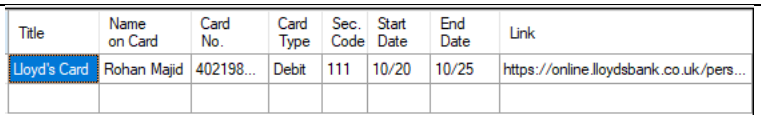
○ If a row isn't selected first, the next empty row is where the information is saved.	 ○ The same item is inputted again without being set a specific location, and it is saved in the next empty row. ☑Success
Delete –	
▪ When this is clicked, the data grid will remove that row permanently.	 ☑Success
▪ If a row isn't selected first, an error message box informs the user to do so.	 ☑Success

Title	Name on Card	Card No.	Card Type	Sec. Code	Start Date	End Date	Link
Lloyd's Card	Rohan Majd	402198183687309	Debt	111	10/20	10/25	https://online
Lloyd's Card	Rohan Ma...	402198183687309	Debt	111	10/20	10/25	https://online

Title	Name on Card	Card No.	Card Type	Sec. Code	Start Date	End Date	Link
Lloyd's Card	Rohan Majd	402198183687309	Debt	111	10/20	10/25	https://online

○ Cell Clicks

Test Feature	Test Input Result

Double click - the information for the double-clicked cell will transfer the whole row's information into their corresponding text boxes.	 ☑Success
Single click – The program will set that as the row to be overwritten or deleted, depending on the user's next input.	 ☑Success

Addresses

CODED SOLUTION 1

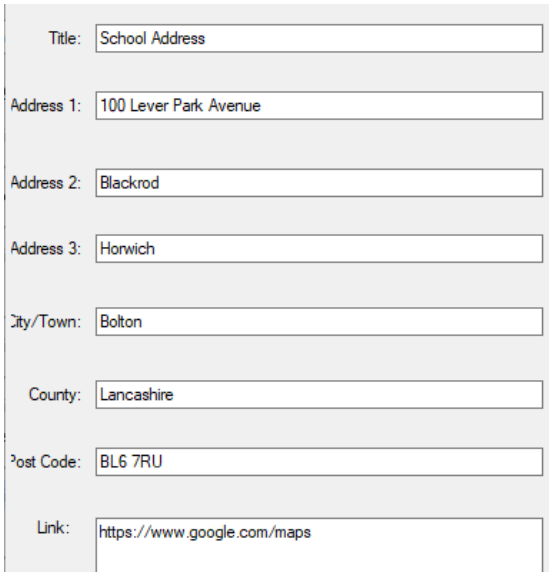
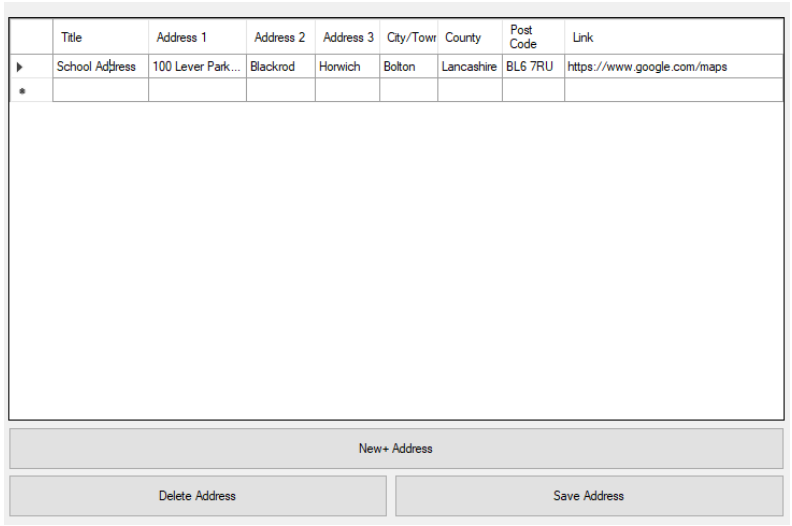
```
// for addresses
DataTable a_dataTable = new DataTable();
bool editingAddress = false;

1 reference
private void a_buttonDelete_Click(object sender, EventArgs e)
{
    try { a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].Delete(); }
    catch (Exception) { MessageBox.Show("Select row to delete."); }
}
```

```
1 reference
private void a_dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        a_textBoxTitle.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[0].ToString();
        a_textBoxAddress1.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
        a_textBoxAddress2.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[2].ToString();
        a_textBoxAddress3.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[3].ToString();
        a_textBoxCity.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[4].ToString();
        a_textBoxCounty.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[5].ToString();
        a_textBoxPostCode.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[6].ToString();
        a_textBoxLink.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[7].ToString();
        editingAddress = true;
    }
    catch (Exception) { MessageBox.Show("Selected empty row."); }
}
```

```
1 reference
private void a_buttonNew_Click(object sender, EventArgs e)
{
    a_textBoxTitle.Text = ""; a_textBoxAddress1.Text = ""; a_textBoxAddress2.Text = ""; a_textBoxAddress3.Text = "";
    a_textBoxCity.Text = ""; a_textBoxCounty.Text = ""; a_textBoxPostCode.Text = ""; a_textBoxLink.Text = "";
}

1 reference
private void a_buttonSave_Click(object sender, EventArgs e)
{
    if (editingAddress == true)
    {
        a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex]["Title"] = a_textBoxTitle.Text;
        a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex]["Address 1"] = a_textBoxAddress1.Text;
        a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex]["Address 2"] = a_textBoxAddress2.Text;
        a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex]["Address 3"] = a_textBoxAddress3.Text;
        a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex]["City"] = a_textBoxCity.Text;
        a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex]["County"] = a_textBoxCounty.Text;
        a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex]["Post Code"] = a_textBoxPostCode.Text;
        a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex]["Link"] = a_textBoxLink.Text;
    }
    else
    {
        a_dataTable.Rows.Add(a_textBoxTitle.Text, a_textBoxAddress1.Text, a_textBoxAddress2.Text, a_textBoxAddress3.Text,
            a_textBoxCity.Text, a_textBoxCounty.Text, a_textBoxPostCode.Text, a_textBoxLink.Text);
    }
    a_textBoxTitle.Text = ""; a_textBoxAddress1.Text = ""; a_textBoxAddress2.Text = ""; a_textBoxAddress3.Text = "";
    a_textBoxCity.Text = ""; a_textBoxCounty.Text = ""; a_textBoxPostCode.Text = ""; a_textBoxLink.Text = "";
    editingAddress = false;
}
```

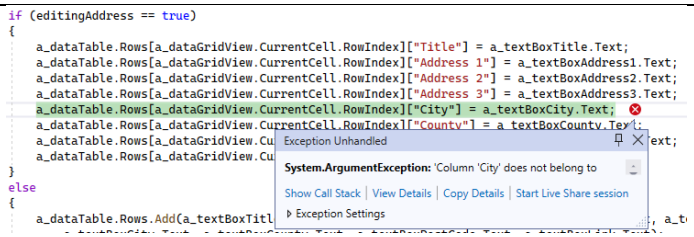
CODE TEST 1

- Entered Characters

Test Feature	Test Data Result
Anything entered in the text box in the vault or sign in page must be saved as string to the data grid or overwritten over existing data in it.	  ○ Save clicked! ☑Success

The login page will need to check if the user's input matches any account in the database.	

- Empty String

Test Feature	Test Data Result
If a text box is empty in the log in page, the program needs to output an error message box telling the user to enter information.	
If a text box is empty in the vault, the program will enter the empty string in its corresponding cell in the data grid.	 ○ The error shows because the name of the column, 'City', does not match the column added to the data grid, which was ladled 'City/Town'.

	○ I also noticed that the 'County' name did not match correctly with the column in the data grid. It should be named 'County/State'. ☒Success

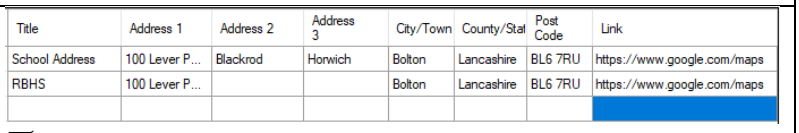
CODED SOLUTION 2

I renamed the titles for the city and county columns to 'City/Town' and 'County/State' respectively.

```
a_dataTable.Columns.Add("Title"); a_dataTable.Columns.Add("Address 1"); a_dataTable.Columns.Add("Address 2");
a_dataTable.Columns.Add("Address 3"); a_dataTable.Columns.Add("City/Town"); a_dataTable.Columns.Add("County/State");
a_dataTable.Columns.Add("Post Code"); a_dataTable.Columns.Add("Link"); a_dataGridView.DataSource = a_dataTable;
```

CODE TEST 2

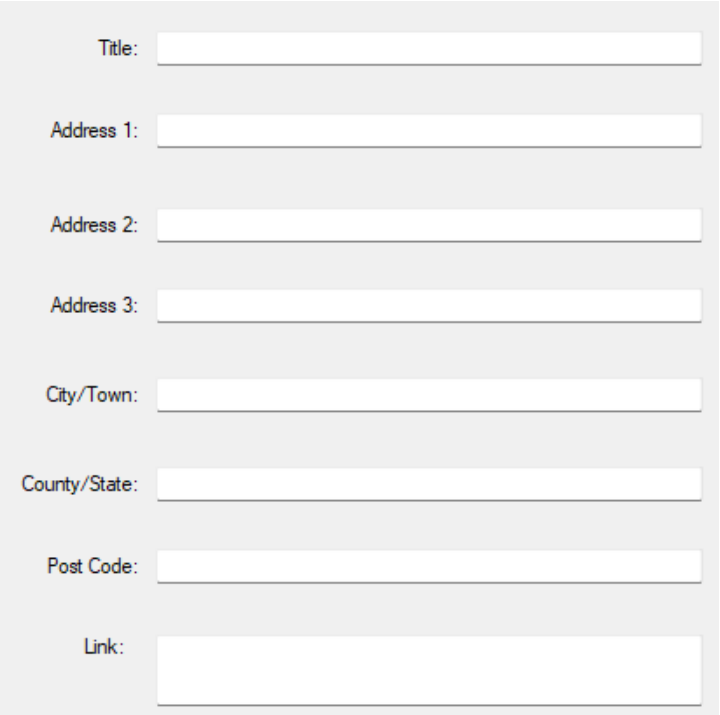
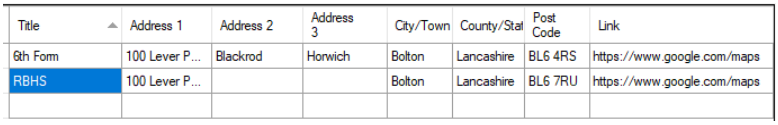
- Empty String

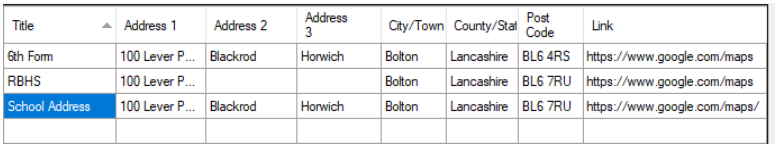
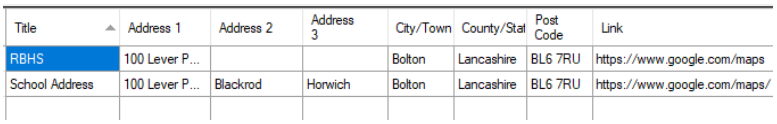
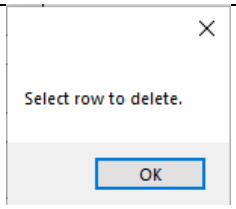
Test Feature	Test Data Result
If a text box is empty in the vault, the program will enter the empty string in its corresponding cell in the data grid.	 ☑Success

Title	Address 1	Address 2	Address 3	City/Town	County/Stat	Post Code	Link
School Address	100 Lever P...	Blackrod	Horwich	Bolton	Lancashire	BL6 7RU	https://www.google.com/maps
RBHS	100 Lever P...			Bolton	Lancashire	BL6 7RU	https://www.google.com/maps

- Button Clicks

Test Feature	Test Input Result

New – when this is clicked, the program needs to clear all text boxes, ready for input by the user.	 ☑Success
Save -	
○ When this is clicked, all entered information will be transferred from the text boxes to be overwritten into their corresponding data grid rows.	 Title | Address 1 | Address 2 | Address 3 | City/Town | County/Stat | Post Code | Link 6th Form | 100 Lever P... | Blackrod | Horwich | Bolton | Lancashire | BL6 4RS | https://www.google.com/maps RBHS | 100 Lever P... | | | Bolton | Lancashire | BL6 7RU | https://www.google.com/maps ○ The top title was overwritten from 'School Address' to '6th Form'. And the next two columns were made blank. ☑Success

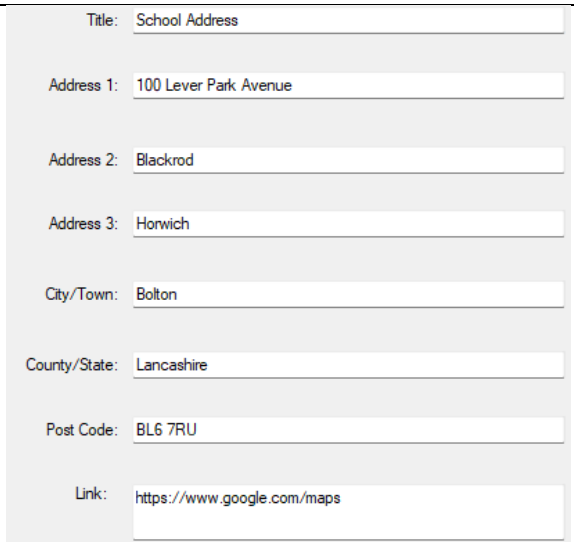
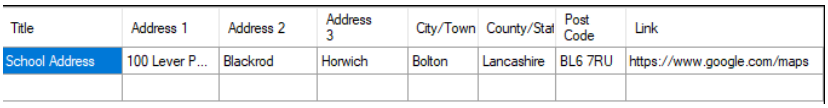
○ If a row isn't selected first, the next empty row is where the information is saved.	 ○ 'School Address' is inputted again and saved in the next empty space. ☑Success
Delete –	
▪ When this is clicked, the data grid will remove that row permanently.	 ☑Success
▪ If a row isn't selected first, an error message box informs the user to do so.	 ☑Success

Title	Address 1	Address 2	Address 3	City/Town	County/Stat	Post Code	Link
6th Form	100 Lever P...	Blackrod	Horwich	Bolton	Lancashire	BL6 4RS	https://www.google.com/maps
RBHS	100 Lever P...			Bolton	Lancashire	BL6 7RU	https://www.google.com/maps
School Address	100 Lever P...	Blackrod	Horwich	Bolton	Lancashire	BL6 7RU	https://www.google.com/maps/

Title	Address 1	Address 2	Address 3	City/Town	County/Stat	Post Code	Link
RBHS	100 Lever P...			Bolton	Lancashire	BL6 7RU	https://www.google.com/maps
School Address	100 Lever P...	Blackrod	Horwich	Bolton	Lancashire	BL6 7RU	https://www.google.com/maps/

○ Cell Clicks

Test Feature	Test Input Result

Double click - the information for the double-clicked cell will transfer the whole row's information into their corresponding text boxes.	 ☑Success
Single click – The program will set that as the row to be overwritten or deleted, depending on the user's next input.	 ○ 'RBHS' item was deleted. ☑Success

Notes

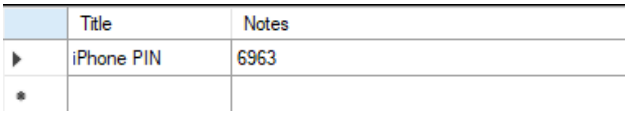
CODED SOLUTION

```
// for notes
DataTable n_dataTable = new DataTable();
bool editingNote = false;

1 reference
private void n_buttonDelete_Click(object sender, EventArgs e)
{
    try { n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex].Delete(); }
    catch (Exception) { MessageBox.Show("Select row to delete."); }
}
```

```
1 reference
private void n_dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        n_textBoxTitle.Text = n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex].ItemArray[0].ToString();
        n_textBoxNotes.Text = n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
        editingNote = true;
    }
    catch (Exception)
    { MessageBox.Show("Selected empty row."); }
}

1 reference
private void n_buttonNew_Click(object sender, EventArgs e)
{
    n_textBoxTitle.Text = ""; n_textBoxNotes.Text = "";
    //n_dataGridView.DataSource = n_dataTable;
}

1 reference
private void n_buttonSave_Click(object sender, EventArgs e)
{
    if (editingNote == true)
    {
        n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex]["Title"] = n_textBoxTitle.Text;
        n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex]["Notes"] = n_textBoxNotes.Text;
    }
    else // This is if the note is new
    {
        n_dataTable.Rows.Add(n_textBoxTitle.Text, n_textBoxNotes.Text);
    }
    n_textBoxTitle.Text = ""; n_textBoxNotes.Text = "";
    editingNote = false;
}
```

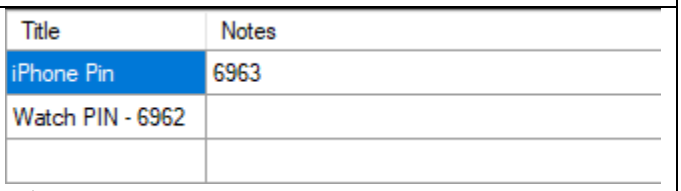
CODE TEST

- Entered Characters

Test Feature	Test Data Result
Anything entered in the text box in the vault or sign in page must be saved as string to the data grid or overwritten over existing data in it.	 ☑Success
The login page will need to check if the user's	

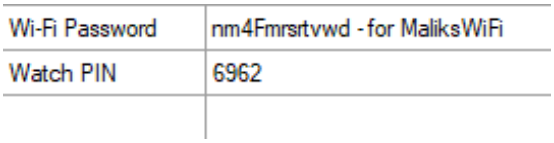
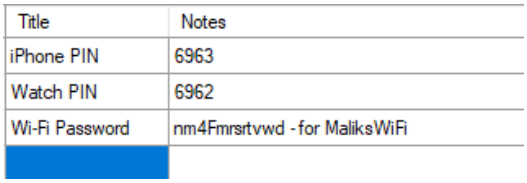
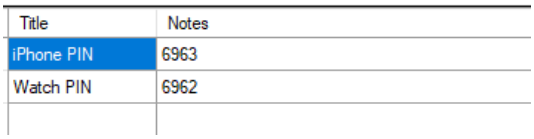
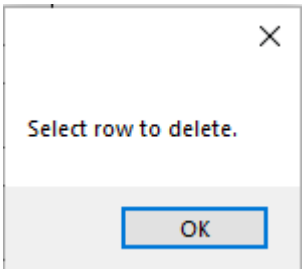
input matches any account in the database.	

- Empty String

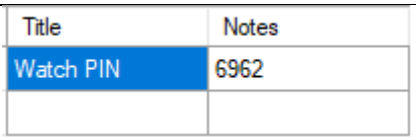
Test Feature	Test Data Result
If a text box is empty in the log in page, the program needs to output an error message box telling the user to enter information.	
If a text box is empty in the vault, the program will enter the empty string in its corresponding cell in the data grid.	 ☑Success

- Button Clicks

Test Feature	Test Input Result
New – when this is clicked, the program needs to clear all text boxes, ready for input by the user.	 ○ Text boxes become blank when 'New+' clicked. ☑Success

Save -	
○ When this is clicked, all entered information will be transferred from the text boxes to be overwritten into their corresponding data grid rows.	 ☑Success
○ If a row isn't selected first, the next empty row is where the information is saved.	 ☑Success
Delete –	
▪ When this is clicked, the data grid will remove that row permanently.	 ☑Success
▪ If a row isn't selected first, an error message box informs the user to do so.	 ☑Success

- Cell Clicks

Test Feature	Test Input Result
Double click - the information for the double-clicked cell will transfer the whole row's information into their corresponding text boxes.	 ☑Success
Single click – The program will set that as the row to be overwritten or deleted, depending on the user's next input.	 ○ When the iPhone pin was selected, it was then deleted. ☑Success

Password Generator

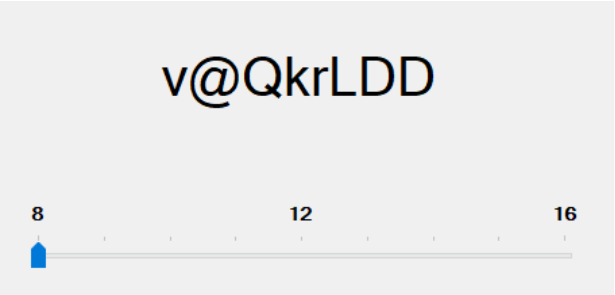
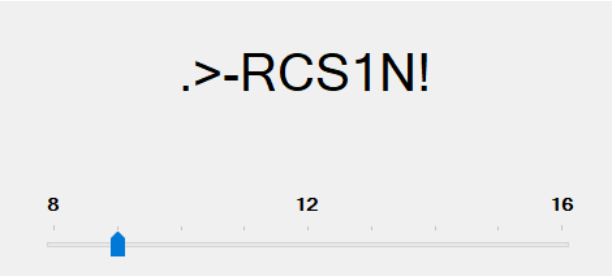
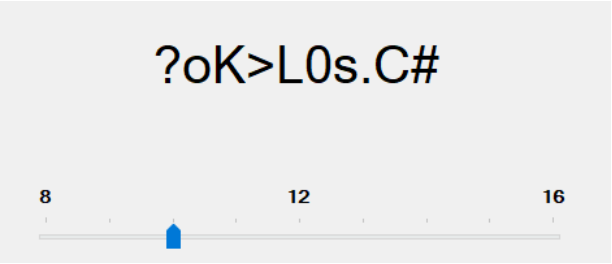
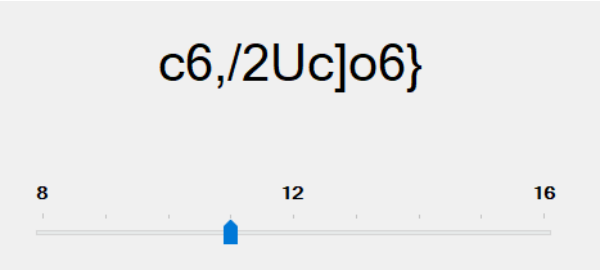
CODED SOLUTION

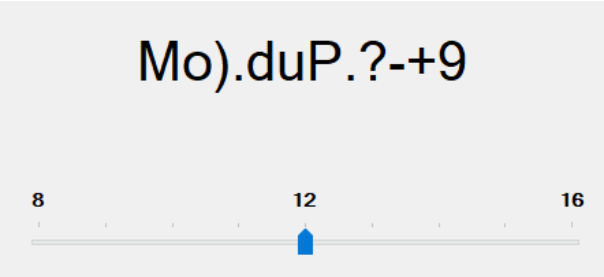
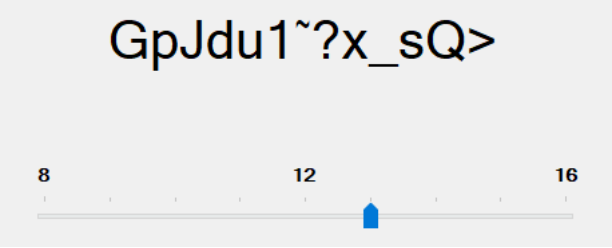
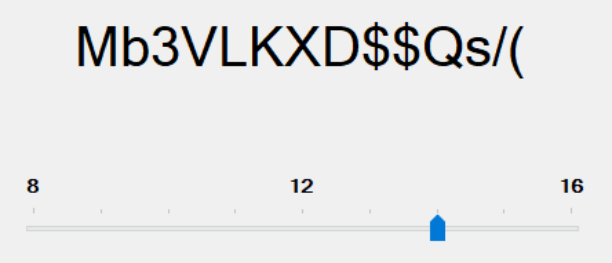
```
// Password Generator
Random RandomCharacter = new Random();
int currentPasswordLength = 0;

1 reference
private void trackBarPasswordLength_Scroll(object sender, EventArgs e)
{
    currentPasswordLength = trackBarPasswordLength.Value;
    randomPasswordGenerator(currentPasswordLength);
}

1 reference
private void randomPasswordGenerator(int passwordLength)
{

    char[] Characters =
    {
            'Q','W','E','R','T','Y','U','I','O','P','A','S','D','F','G','H','J','K','L','Z',
            'X','C','V','B','N','M','q','w','e','r','t','y','u','i','o','p','a','s','d','f',
            'g','h','j','k','l','z','x','c','v','b','n','m','1','2','3','4','5','6','7','8',
            '9','0','!','£','$','%','^','&','*','(',')','_','-','+','=','{','}','[',']',';',
            ':','@','#','~',',','<','>','.','/','?'
    };

    string randomPassword = "";

    for ( int i = 0; i < passwordLength; i++ )
    {
        randomPassword += Characters[RandomCharacter.Next(0,90)];
    }

    labelPassword.Text = randomPassword;

}

1 reference
private void buttonCopyPassword_Click(object sender, EventArgs e)
{
    Clipboard.SetText(labelPassword.Text);
}
```

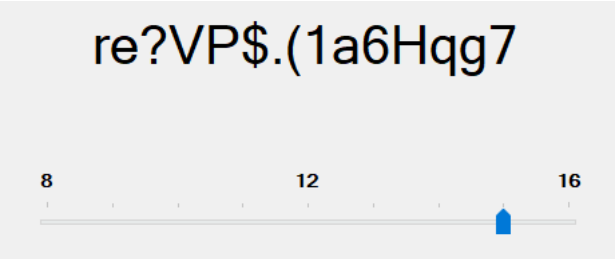
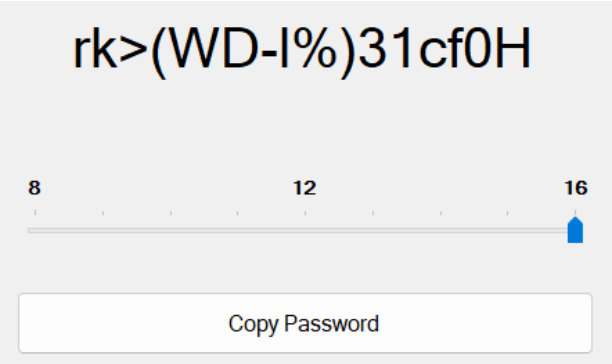
CODE TEST

- Sliding

Test Feature	Test Input Result

Drag Slider – The slider's index must declare the length of the randomly generated password.	 ○ The index matches the password length. ☑Success  ○ ☑Success  ○ ☑Success 

	○ ☑Success  ○ ☑Success  ○ ☑Success  ○ ☑Success

	 ○ ☑Success  ○ ☑Success
Copy Password – when clicked, the generated password is copied to the clipboard.	 ○ Previous password was copied into the textbox☑Success

Hashing Algorithm

CODED SOLUTION

```
// Hashing Algorithm
0 references
public string toHash256( string passwordInput)
{
    var sha256 = SHA256.Create();
    byte[] bytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(passwordInput));

    var hashedPassword = new StringBuilder();
    for (int i = 0; i < bytes.Length; i++ )
    {
        hashedPassword.Append(bytes[i].ToString("x2"));
    }
    return hashedPassword.ToString();
}
```

CODE TEST

- Hashing

Test Feature	Test Input Result
Hash some string in the program.	`labelPassword.Text = toHash256(randomPassword);` ○ I decided to edit the password generator temporarily so that it displays hashed passwords. 336206a5a377c84bee b9455b8eafa7df44be ○ These are hashed passwords of different lengths. ☑Success

DATABASE DEVELOPMENT & TESTING

Some Recurring tweaks were made throughout multiple sections of code:

- editingX = false is after every buttonNew_Click(). I realised without this, if the user clicks 'new' after loading information, the program will still think it is being edited and will overwrite data there instead of saving it in a new place.
- There is a new ID column at the start of each table in the data tables in the program and in Access. This is to uniquely identify each item the user inputs. So, when data is saved, each 'x_dataTable.Rows[x_dataGrid.CurrentCell.RowIndex].ItemArray[x]' index will be +1.
- 'x_query' is SQL code that retrieves a row in the data table.
- The OleDbConnection establishes a connection to a database using the OleDb data provider. The OleDb data provider is used to connect to data sources such as Microsoft Access.
- The OleDbDataAdapter is a class that provides the communication between a database and a DataTable. It provides methods to fetch data from a data source and populate a DataTable and to update the data source with changes made to the DataTable.
- OleDbCommand is used when prewritten SQL statements will be utilised when accessing a method.
- GetX() is a new method that is used to access data from a specific table in a database and display it in the data grid. The method is called when the vault is loaded and when the data table is updated (but it is cleared before the new information is inserted, or else duplicates will be created).

Here are the methods and OleDb classes:

```
OleDbConnection connection = new OleDbConnection("Provider=Microsoft.ACE.OleDb.16.0; " +
    "Data Source = D:\\PassVault\\PassVaultDatabase.accdb");

OleDbCommand p_command;
OleDbCommand ba_command;
OleDbCommand c_command;
OleDbCommand a_command;
OleDbCommand n_command;

OleDbDataAdapter p_dataAdapter;
OleDbDataAdapter ba_dataAdapter;
OleDbDataAdapter c_dataAdapter;
OleDbDataAdapter a_dataAdapter;
OleDbDataAdapter n_dataAdapter;

DataTable p_dataTable = new DataTable();
DataTable ba_dataTable = new DataTable();
DataTable c_dataTable = new DataTable();
DataTable a_dataTable = new DataTable();
DataTable n_dataTable = new DataTable();

bool editingPassword = false;
bool editingBankAccount = false;
bool editingCard = false;
bool editingAddress = false;
bool editingNote = false;
```

```
3 references
void GetPasswords()
{
    p_dataAdapter = new OleDbDataAdapter("SELECT * FROM Passwords", connection);
    connection.Open();
    p_dataAdapter.Fill(p_dataTable);
    p_dataGridView.DataSource = p_dataTable;
    connection.Close();
}
```

```
3 references
void GetBankAccounts()
{
    ba_dataAdapter = new OleDbDataAdapter("SELECT * FROM BankAccounts", connection);
    connection.Open();
    ba_dataAdapter.Fill(ba_dataTable);
    ba_dataGridView.DataSource = ba_dataTable;
    connection.Close();
}
```

```
3 references
void GetCards()
{
    c_dataAdapter = new OleDbDataAdapter("SELECT * FROM Cards", connection);
    connection.Open();
    c_dataAdapter.Fill(c_dataTable);
    c_dataGridView.DataSource = c_dataTable;
    connection.Close();
}
```

```
3 references
void GetAddresses()
{
    a_dataAdapter = new OleDbDataAdapter("SELECT * FROM Addresses", connection);
    connection.Open();
    a_dataAdapter.Fill(a_dataTable);
    a_dataGridView.DataSource = a_dataTable;
    connection.Close();
}
```

```
4 references
void GetNotes()
{
    n_dataAdapter = new OleDbDataAdapter("SELECT * FROM Notes", connection);
    connection.Open();
    n_dataAdapter.Fill(n_dataTable);
    n_dataGridView.DataSource = n_dataTable;
    connection.Close();
}
```

```
1 reference
private void PassVault_Load(object sender, EventArgs e)
{

    GetPasswords();

    GetBankAccounts();

    GetCards();

    GetAddresses();

    GetNotes();

}
```

I will focus on the notes section for now. It is the easiest and simplest section to edit because it only has three rows (for the ID, title, and notes). Once I have tested it, I will copy the code for every other type of password that could be saved into their respective sections. I will only need to change variable names for each section.

Login/ Sign Up

Due to a lack of programming knowledge, time constraints, and a lack of resources (a lack of a Microsoft Access database engine). I was unable to create a login system for multiple users. Instead, I had to limit to one user per device.

Notes

CODED SOLUTION 1

```
1 reference
private void n_buttonDelete_Click(object sender, EventArgs e)
{
    try
    {
        n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex].Delete(); // front end (data grid)

        // back end (Access table)
        string n_query = "Delete FROM Notes WHERE NoteID=@NoteID";

        n_command = new OleDbCommand(n_query, connection);
        n_command.Parameters.AddWithValue("@NoteID", Convert.ToInt32(n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex]["NoteID"]));

        connection.Open();
        n_command.ExecuteNonQuery();
        connection.Close();

        n_dataTable.Clear();
        GetNotes();
    }
    catch (Exception) { MessageBox.Show("Select row to delete."); }
}
```

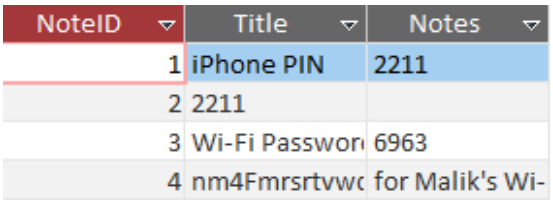
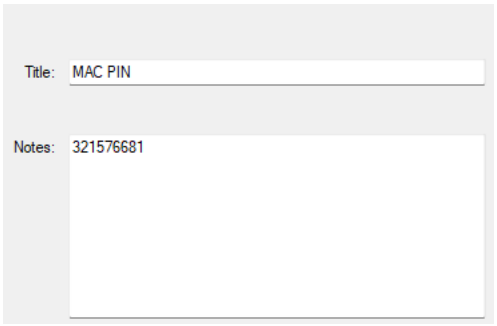
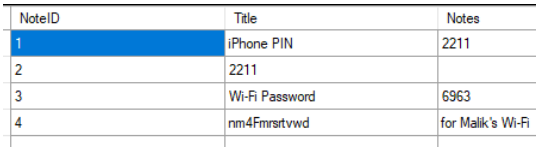
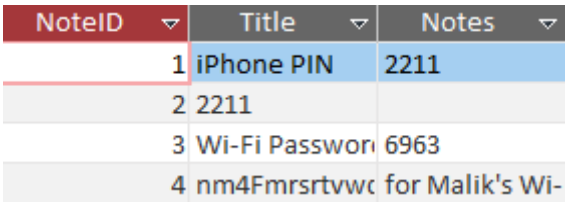
```
1 reference
private void n_dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        n_textBoxTitle.Text = n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
        n_textBoxNotes.Text = n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex].ItemArray[2].ToString();
        editingNote = true;
    }
    catch (Exception)
    { MessageBox.Show("Selected empty row."); }
}
```

```
1 reference
private void n_buttonNew_Click(object sender, EventArgs e)
{
    n_textBoxTitle.Text = ""; n_textBoxNotes.Text = "";
    editingNote = false;
}
```

```
1 reference
private void n_buttonSave_Click(object sender, EventArgs e)
{
    if (editingNote == true)
    {
        string n_query = "UPDATE Notes SET Title=@Title, Notes=@Notes " +
            "WHERE NoteID=@NoteID";

        n_command = new OleDbCommand(n_query, connection);
        n_command.Parameters.AddWithValue("@NoteID", Convert.ToInt32(n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex]["NoteID"]));
        n_command.Parameters.AddWithValue("@Title", n_textBoxTitle.Text);
        n_command.Parameters.AddWithValue("@Notes", n_textBoxNotes.Text);

        connection.Open();
        n_command.ExecuteNonQuery();
        connection.Close();

        n_dataTable.Clear();
        GetNotes();
    }

    else // This is if the note is new
    {
        string n_query = "INSERT INTO Notes (Title,Notes) VALUES (@Title,@Notes)";

        n_command = new OleDbCommand(n_query, connection);
        n_command.Parameters.AddWithValue("@Title", n_textBoxTitle.Text);
        n_command.Parameters.AddWithValue("@Notes", n_textBoxNotes.Text);
        connection.Open();
        n_command.ExecuteNonQuery();
        connection.Close();

        n_dataTable.Clear();
        GetNotes();
    }

    n_textBoxTitle.Text = ""; n_textBoxNotes.Text = "";
    editingNote = false;

    /* ...
}
```

- Button Clicks

Test Data –	<u>Notes:</u> - iPhone PIN - 2211 - 2211 - Wi-Fi Password - 6963 - nm4Fmrsrtvwd - for MaliksWiFi
Save -	
○ If a row isn't selected first, the next empty row of the Access table is where the information is saved. ○ The ID column must also update automatically.	○ Click one of these buttons in their corresponding tab: save password, save address, save card, save bank account, and save note. NoteID | Title | Notes 1 | iPhone PIN | 2211 2 | 2211 | 3 | Wi-Fi Password | 6963 4 | nm4Fmrsrtvwd | for Malik's Wi-Fi

	 ☑Success
○ If a row is selected, that is where information is overwritten.	○ The second row is selected. The row is then edited by changing the title and notes to what is below.    ○ Rows changed in neither apps.

	☒Success

CODED SOLUTION 2

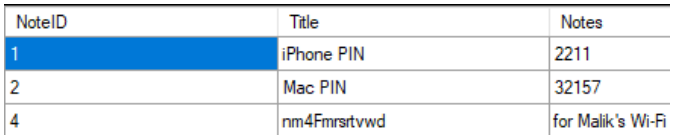
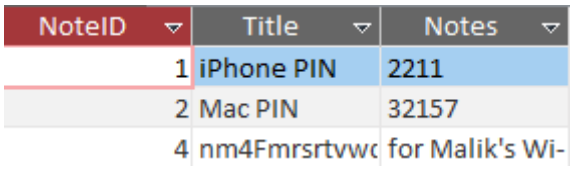
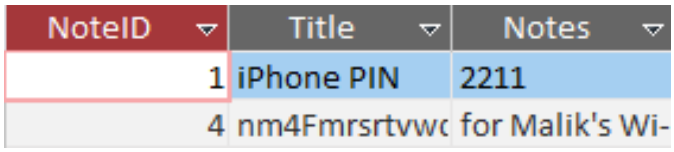
```
n_command = new OleDbCommand(n_query, connection);
n_command.Parameters.AddWithValue("@Title", n_textBoxTitle.Text);
n_command.Parameters.AddWithValue("@Notes", n_textBoxNotes.Text);
n_command.Parameters.AddWithValue("@NoteID", Convert.ToInt32(n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex]["NoteID"]));
```

The bottom line was originally in the second position. The new position allows the title and notes column to be edited too.

CODE TEST 2

- Button Clicks

Save -	
When the cell is double clicked, all entered information will be transferred from the text boxes to be overwritten into their corresponding data tables in Access.	○ The second row is loaded again.  NoteID | Title | Notes 1 | iPhone PIN | 2211 2 | Mac PIN | 32157 3 | Wi-Fi Password | 6963 4 | nm4Fmrsrtvwd | for Malik's Wi-Fi ☑Success

Delete -	
When this is clicked, the data grid will remove that row permanently, in the data grid and in Access.	 NoteID | Title | Notes 1 | iPhone PIN | 2211 2 | Mac PIN | 32157 4 | nm4Fmrsrtvwd | for Malik's Wi-Fi  NoteID | Title | Notes 1 | iPhone PIN | 2211 2 | Mac PIN | 32157 4 | nm4Fmrsrtvwc | for Malik's Wi- ○ I selected the 4th row, but the 3rd row was deleted when 'delete' was clicked.  NoteID | Title | Notes 1 | iPhone PIN | 2211 4 | nm4Fmrsrtvwd | for Malik's Wi-Fi  NoteID | Title | Notes 1 | iPhone PIN | 2211 4 | nm4Fmrsrtvwc | for Malik's Wi- ○ I tried to delet the 4th row again to ckeck if I'm going mad, but it still deleted the row above it. ☒Success

CODED SOLUTION 3

```
1 reference
private void n_buttonDelete_Click(object sender, EventArgs e)
{
    try
    {
        n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex].Delete(); // front end (data grid)

        // back end (Access table)
        string n_query = "Delete FROM Notes WHERE NoteID=@NoteID";

        n_command = new OleDbCommand(n_query, connection);
        n_command.Parameters.AddWithValue("@NoteID", Convert.ToInt32(n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex]["NoteID"]));

        connection.Open();
        n_command.ExecuteNonQuery();
        connection.Close();

        //n_dataTable.Clear();
        /*  ...

        n_dataTable = new DataTable();
        n_dataAdapter = new OleDbDataAdapter("SELECT *FROM Notes", connection);
        connection.Open();
        n_dataAdapter.Fill(n_dataTable);
        n_dataGridView.DataSource = n_dataTable;
        connection.Close();
    }
    catch (Exception) { MessageBox.Show("Select row to delete."); }
}
```

In this code, instead of calling the method, I replaced it with the code inside it to see what the problem is better. I replaced the 'n_dataTable.Clear()' method with a new data table, to correct the indexing of the rows. I commented it out instead of deleting it and re-typing it later and I will do the same for subsequent code.

CODE TEST 3

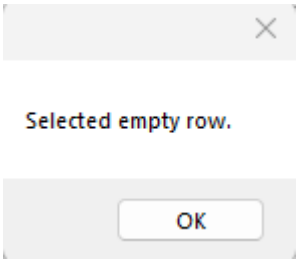
- Button Clicks

Delete -	
When this is clicked, the data grid will remove that row permanently, in the data grid and in Access.	<table><tr><th>NoteID</th><th>Title</th><th>Notes</th></tr><tr><td>1</td><td>iPhone PIN</td><td>2211</td></tr><tr><td>4</td><td>nm4Fmrsrtvwd</td><td>for Malik's Wi-Fi</td></tr><tr><td>6</td><td>Mac PIN</td><td>2211</td></tr><tr><td>7</td><td>Wi-Fi Password</td><td>6963</td></tr></table> - The deleted data is re-entered.

NoteID	Title	Notes
4	nm4Fmrsrtvwd	for Malik's Wi-Fi
6	Mac PIN	2211
7	Wi-Fi Password	6963

NoteID	Title	Notes
1	iPhone PIN	2211
4	nm4Fmrsrtvwc	for Malik's Wi-
6	Mac PIN	2211
7	Wi-Fi Passwor	6963

- The Data is deleted in the data grid buut not the database table



- This also appeared when I did not even do this.

NoteID	Title	Notes
4	nm4Fmrsrtvwd	for Malik's Wi-Fi
6	Mac PIN	2211
7	Wi-Fi Password	6963

- The indexing of the rows is wrong. They are all +1, so when the second row is double clicked, the contents of the first row is loaded.

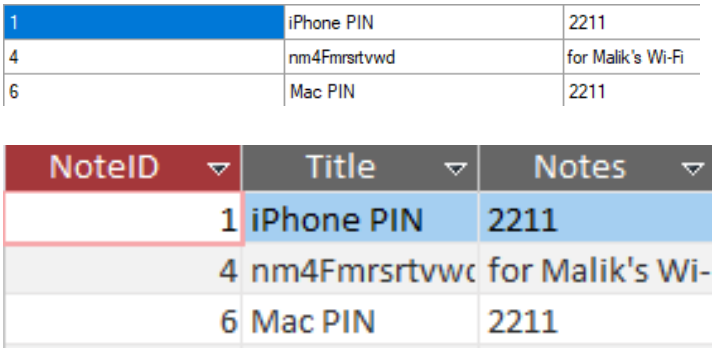
☒Success

```
1 reference
private void n_buttonDelete_Click(object sender, EventArgs e)
{
    try
    {
        //n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex].Delete(); // front end (data grid)

        // back end (Access table)
        string n_query = "Delete FROM Notes WHERE NoteID=@NoteID";

        n_command = new OleDbCommand(n_query, connection);
        n_command.Parameters.AddWithValue("@NoteID", Convert.ToInt32(n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex]["NoteID"]));

        connection.Open();
        n_command.ExecuteNonQuery();
        connection.Close();

        /*
        n_dataAdapter = new OleDbDataAdapter("SELECT * FROM Notes", connection);
        connection.Open();
        n_dataAdapter.Fill(n_dataTable);
        n_dataGridView.DataSource = n_dataTable;
        connection.Close();
        */

        n_dataTable.Clear();
        n_dataAdapter = new OleDbDataAdapter("SELECT *FROM Notes", connection);
        connection.Open();
        n_dataAdapter.Fill(n_dataTable);
        n_dataGridView.DataSource = n_dataTable;
        connection.Close();
    }
    catch (Exception) { MessageBox.Show("Select row to delete."); }
}
```

I realised the commented line on the top of the try block is irrelevant. The data grid matches whatever is in the Access database, so if something is deleted there, the data grid will copy that.

CODE TEST 4

- Button Clicks

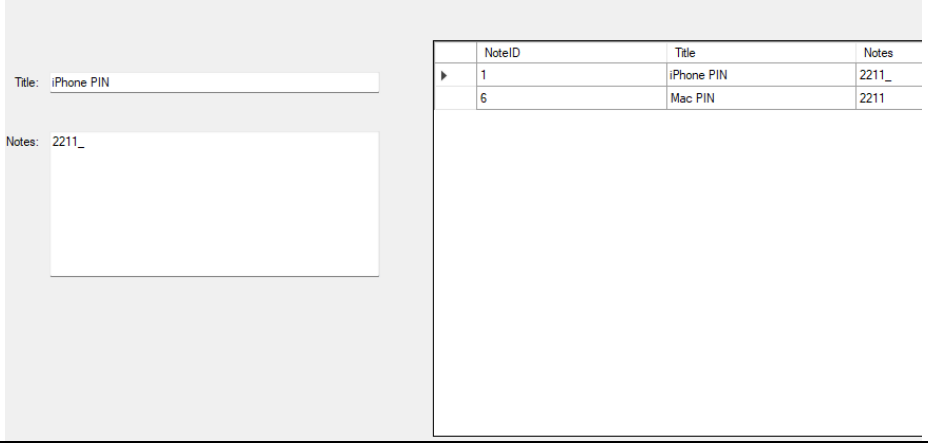
Delete -	
When this is clicked, the data grid will remove that row permanently, in the data grid and in Access.	

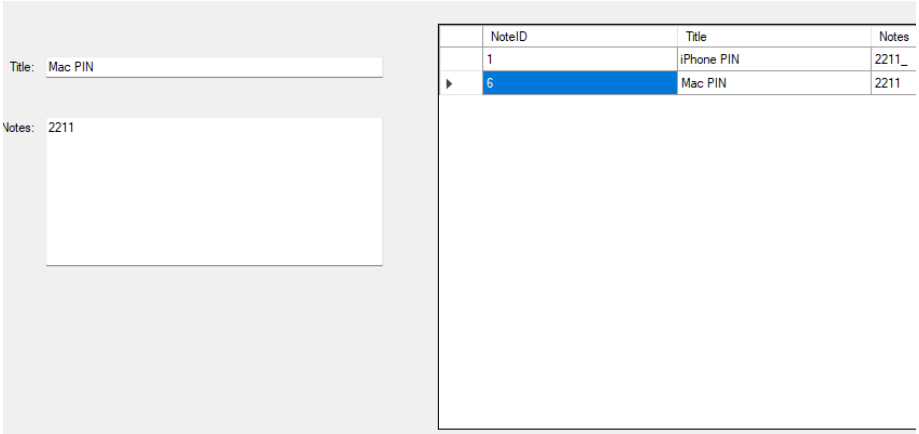
- Finally, the data table matches what is in the data grid.

NoteID	Title	Notes
1	iPhone PIN	2211
6	Mac PIN	2211

NoteID	Title	Notes
1	iPhone PIN	2211
6	Mac PIN	2211

- I tried again on the second row, and it still worked.



	 ○ The indexing is correct too. When the rows are double clicked, the right information is loaded in the right text boxes. ☑Success

CODED SOLUTION 4B

```
private void n_buttonDelete_Click(object sender, EventArgs e)
{
    try
    {
        string n_query = "Delete FROM Notes WHERE NoteID=@NoteID";

        n_command = new OleDbCommand(n_query, connection);
        n_command.Parameters.AddWithValue("@NoteID", Convert.ToInt32(n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex]["NoteID"]));

        connection.Open();
        n_command.ExecuteNonQuery();
        connection.Close();

        n_dataTable.Clear();
        GetNotes();
    }
    catch (Exception) { MessageBox.Show("Select row to delete."); }
}
```

Unnecessary comments or lines of code have been removed.

Passwords

CODED SOLUTION

```
// for passwords

1 reference
private void p_buttonDelete_Click(object sender, EventArgs e)
{
    try
    {
        string p_query = "Delete FROM Passwords WHERE PasswordID=@PasswordID";

        p_command = new OleDbCommand(p_query, connection);
        p_command.Parameters.AddWithValue("@PasswordID", Convert.ToInt32(p_dataTable.Rows[p_dataGridView.CurrentCell.
            RowIndex]["PasswordID"]));

        connection.Open();
        p_command.ExecuteNonQuery();
        connection.Close();

        p_dataTable.Clear();
        GetPasswords();
    }
    catch (Exception) { MessageBox.Show("Select row to delete."); }
}

1 reference
private void p_dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        p_textBoxTitle.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
        p_textBoxUsername.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[2].ToString();
        p_textBoxPassword.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[3].ToString();
        p_textBoxLink.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[4].ToString();
        editingPassword = true;
    }
    catch (Exception) { MessageBox.Show("Selected no row / empty row."); }
}

1 reference
private void p_buttonNew_Click(object sender, EventArgs e)
{
    p_textBoxTitle.Text = ""; p_textBoxUsername.Text = ""; p_textBoxPassword.Text = ""; p_textBoxLink.Text = "";
    editingPassword = false;
}
```

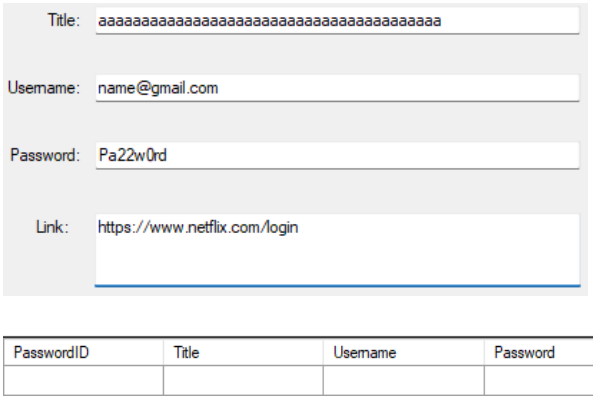
```
1 reference
private void p_buttonSave_Click(object sender, EventArgs e)
{
    if (editingPassword == true)
    {
        string p_query = "UPDATE Passwords SET Title=@Title, Username=@Username, Password=@Password, Link=@Link " +
            "WHERE PasswordID=@PaswordID";

        p_command = new OleDbCommand(p_query, connection);
        p_command.Parameters.AddWithValue("@Title", p_textBoxTitle.Text);
        p_command.Parameters.AddWithValue("@Username", p_textBoxUsername.Text);
        p_command.Parameters.AddWithValue("@Password", p_textBoxPassword.Text);
        p_command.Parameters.AddWithValue("@Link", p_textBoxLink.Text);
        p_command.Parameters.AddWithValue("@PasswordID", Convert.ToInt32(p_dataTable.Rows[p_dataGridView.CurrentCell.
            RowIndex]["PasswordID"]));

        connection.Open();
        p_command.ExecuteNonQuery();
        connection.Close();

        p_dataTable.Clear();
        GetPasswords();
    }
    else
    {
        string p_query = "INSERT INTO Password (Title, Username, Password, Link) VALUES (@Title, @Username, " +
            "@Password, @Link)";

        p_command = new OleDbCommand(p_query, connection);
        p_command.Parameters.AddWithValue("@Title", p_textBoxTitle.Text);
        p_command.Parameters.AddWithValue("@Username", p_textBoxUsername.Text);
        p_command.Parameters.AddWithValue("@Password", p_textBoxPassword.Text);
        p_command.Parameters.AddWithValue("@Link", p_textBoxLink.Text);
        connection.Open();
        p_command.ExecuteNonQuery();
        connection.Close();

        p_dataTable.Clear();
        GetPasswords();
    }
    /* ...
    p_textBoxTitle.Text = ""; p_textBoxUsername.Text = ""; p_textBoxPassword.Text = ""; p_textBoxLink.Text = "";
    editingPassword = false;
}
```

CODE TEST

- Button Clicks

Test Data -	<u>Passwords:</u> - a* 40 (This is to test how long string is displayed in Access cells) - Netflix - name@gmail.com

	- Pa22w0rd - https://www.netflix.com/login/ - YouTube - name@gmail.com - @nnI2525 - https://www.youtube.com/ - Outlook - 16krosahm@rbhs.co.uk - Bb161616 - https://www.outlook.office.com/login/ - Gmail - name@gmail.com - Tt151515 https://mail.google.com/mail/
Save -	

○ If a row isn't selected first, the next empty row of the Access table is where the information is saved. ○ ID column must update automatically too.	 ○ 'System.Data.OleDb.OleDbException: Syntax error in INSERT INTO statement.' Was the output. ☒Success

CODED SOLUTION 2

```
1 reference
private void n_buttonSave_Click(object sender, EventArgs e) //if existing item is being overwritten
{
    if (editingNote == true)
    {
        string query = "UPDATE Notes SET n_Title=@n_Title, n_Notes=@n_Notes " +
            "WHERE NoteID=@NoteID";

        n_command = new OleDbCommand(query, connection);
        n_command.Parameters.AddWithValue("@n_Title", n_textBoxTitle.Text);
        n_command.Parameters.AddWithValue("@n_Notes", n_textBoxNotes.Text);
        n_command.Parameters.AddWithValue("@NoteID", Convert.ToInt32(n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex]["NoteID"]));

        connection.Open();
        n_command.ExecuteNonQuery();
        connection.Close();

        n_dataTable.Clear();
        GetNotes();
    }
    else //if the item is new
    {
        string query = "INSERT INTO Notes (n_Title,n_Notes) VALUES (@n_Title,@n_Notes)";

        n_command = new OleDbCommand(query, connection);
        n_command.Parameters.AddWithValue("@n_Title", n_textBoxTitle.Text);
        n_command.Parameters.AddWithValue("@n_Notes", n_textBoxNotes.Text);

        connection.Open();
        n_command.ExecuteNonQuery();
        connection.Close();

        n_dataTable.Clear();
        GetNotes();
    }

    n_textBoxTitle.Text = ""; n_textBoxNotes.Text = "";
    editingNote = false;
}
```

```
1 reference
private void p_buttonSave_Click(object sender, EventArgs e)
{
    if (editingPassword == true) //if existing item is being overwritten
    {
        string query = "UPDATE Passwords SET p_Title=@p_Title, p_Username=@p_Username, p_Password=@p_Password, p_Link=@p_Link " +
            "WHERE PasswordID=@PasswordID";

        p_command = new OleDbCommand(query, connection);
        p_command.Parameters.AddWithValue("@p_Title", p_textBoxTitle.Text);
        p_command.Parameters.AddWithValue("@p_Username", p_textBoxUsername.Text);
        p_command.Parameters.AddWithValue("@p_Password", p_textBoxPassword.Text);
        p_command.Parameters.AddWithValue("@p_Link", p_textBoxLink.Text);
        p_command.Parameters.AddWithValue("@PasswordID", Convert.ToInt32(p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex]["PasswordID"]));
        connection.Open();
        p_command.ExecuteNonQuery();
        connection.Close();

        p_dataTable.Clear();
        GetPasswords();
    }
    else //if the item is new
    {
        string query = "INSERT INTO Passwords (p_Title, p_Username, p_Password, p_Link) VALUES (@p_Title, @p_Username, @p_Password, @p_Link)";

        p_command = new OleDbCommand(query, connection);
        p_command.Parameters.AddWithValue("@p_Title", p_textBoxTitle.Text);
        p_command.Parameters.AddWithValue("@p_Username", p_textBoxUsername.Text);
        p_command.Parameters.AddWithValue("@p_Password", p_textBoxPassword.Text);
        p_command.Parameters.AddWithValue("@p_Link", p_textBoxLink.Text);

        connection.Open();
        p_command.ExecuteNonQuery();
        connection.Close();

        p_dataTable.Clear();
        GetPasswords();
    }

    p_textBoxTitle.Text = ""; p_textBoxUsername.Text = ""; p_textBoxPassword.Text = ""; p_textBoxLink.Text = "";
    editingPassword = false;

}
```

```
1 reference
private void ba_buttonSave_Click(object sender, EventArgs e) //if existing item is being overwritten
{
    if (editingBankAccount == true)
    {
        string query = "UPDATE BankAccounts SET ba_Title=@ba_Title, ba_Username=@ba_Username, ba_Password=@ba_Password, " +
            "ba_Account Number=@ba_AccountNumber, ba_Sort Code=@ba_SortCode, ba_Link=@ba_Link " +
            "WHERE Bank AccountID=@BankAccountID";

        ba_command = new OleDbCommand(query, connection);
        ba_command.Parameters.AddWithValue("@ba_Title", ba_textBoxTitle.Text);
        ba_command.Parameters.AddWithValue("@ba_Username", ba_textBoxUsername.Text);
        ba_command.Parameters.AddWithValue("@ba_Password", ba_textBoxPassword.Text);
        ba_command.Parameters.AddWithValue("@ba_AccountNumber", ba_textBoxAccNo.Text);
        ba_command.Parameters.AddWithValue("@ba_SortCode", ba_textBoxSortCode.Text);
        ba_command.Parameters.AddWithValue("@ba_Link", ba_textBoxLink.Text);
        ba_command.Parameters.AddWithValue("@BankAccountID", Convert.ToInt32(ba_dataTable.Rows[ba_dataGridView.
            CurrentCell.RowIndex]["BankAccountID"]));

        connection.Open();
        ba_command.ExecuteNonQuery();
        connection.Close();

        ba_dataTable.Clear();
        GetBankAccounts();
    }
    else //if the item is new
    {
        string query = "INSERT INTO Notes (ba_Title, ba_Username, ba_Password, ba_Account Number, ba_Sort Code, ba_Link) " +
            "VALUES (@ba_Title, @ba_Username, @ba_Password, @ba_AccountNumber, @ba_SortCode, @ba_Link)";

        ba_command = new OleDbCommand(query, connection);
        ba_command.Parameters.AddWithValue("@ba_Title", ba_textBoxTitle.Text);
        ba_command.Parameters.AddWithValue("@ba_Username", ba_textBoxUsername.Text);
        ba_command.Parameters.AddWithValue("@ba_Password", ba_textBoxPassword.Text);
        ba_command.Parameters.AddWithValue("@ba_AccountNumber", ba_textBoxAccNo.Text);
        ba_command.Parameters.AddWithValue("@ba_SortCode", ba_textBoxSortCode.Text);
        ba_command.Parameters.AddWithValue("@ba_Link", ba_textBoxLink.Text);
        connection.Open();
        ba_command.ExecuteNonQuery();
        connection.Close();

        ba_dataTable.Clear();
        GetBankAccounts();
    }

    ba_textBoxTitle.Text = ""; ba_textBoxUsername.Text = ""; ba_textBoxPassword.Text = "";
    ba_textBoxAccNo.Text = ""; ba_textBoxSortCode.Text = ""; ba_textBoxLink.Text = "";
    editingBankAccount = false;
}
```

```
1 reference
private void c_buttonSave_Click(object sender, EventArgs e)
{
    if (editingCard == true) //if existing item is being overwritten
    {
        string query = "UPDATE Cards SET c_Title=@c_Title, c_Name on Card=@c_NameOnCard, c_Card Number=@c_CardNumber, " +
            "c_Card Type=@c_CardType, c_Security Code=@c_SecurityCode, c_Start Date=@c_StartDate, c_End Date=@c_EndDate, c_Link=@c_Link" +
            "WHERE CardID=@CardID";

        c_command = new OleDbCommand(query, connection);
        c_command.Parameters.AddWithValue("@c_Title", c_textBoxTitle.Text);
        c_command.Parameters.AddWithValue("@c_NameOnCard", c_textBoxNameOnCard.Text);
        c_command.Parameters.AddWithValue("@c_CardNumber", c_textBoxCardNo.Text);
        c_command.Parameters.AddWithValue("@c_CardType", c_textBoxCardType.Text);
        c_command.Parameters.AddWithValue("@c_SecurityCode", c_textBoxSecCode.Text);
        c_command.Parameters.AddWithValue("@c_StartDate", c_textBoxStartDate.Text);
        c_command.Parameters.AddWithValue("@c_EndDate", c_textBoxEndDate.Text);
        c_command.Parameters.AddWithValue("@c_Link", c_textBoxLink.Text);
        c_command.Parameters.AddWithValue("@CardID", Convert.ToInt32(c_dataTable.Rows[c_dataGridView.CurrentCell.
            RowIndex]["CardID"]));
        connection.Open();
        c_command.ExecuteNonQuery();
        connection.Close();

        c_dataTable.Clear();
        GetCards();
    }
    else //if the item is new
    {
        string query = "INSERT INTO Notes (c_Title, c_Name on Card, c_Card Number, c_Card Type, c_Security Code, c_Start Date, " +
            "c_End Date, c_Link) " +
            "VALUES (@c_Title,@c_NameOnCard, @c_CardNumber, @c_CardType, @c_SecurityCode, @c_StartDate, @c_EndDate, @c_Link)";

        c_command = new OleDbCommand(query, connection);
        c_command.Parameters.AddWithValue("@c_Title", c_textBoxTitle.Text);
        c_command.Parameters.AddWithValue("@c_NameOnCard", c_textBoxNameOnCard.Text);
        c_command.Parameters.AddWithValue("@c_CardNumber", c_textBoxCardNo.Text);
        c_command.Parameters.AddWithValue("@c_CardType", c_textBoxCardType.Text);
        c_command.Parameters.AddWithValue("@c_SecurityCode", c_textBoxSecCode.Text);
        c_command.Parameters.AddWithValue("@c_StartDate", c_textBoxStartDate.Text);
        c_command.Parameters.AddWithValue("@c_EndDate", c_textBoxEndDate.Text);
        c_command.Parameters.AddWithValue("@c_Link", c_textBoxLink.Text);
        connection.Open();
        c_command.ExecuteNonQuery();
        connection.Close();

        c_dataTable.Clear();
        GetCards();
    }

    c_textBoxTitle.Text = ""; c_textBoxNameOnCard.Text = ""; c_textBoxCardNo.Text = ""; c_textBoxCardType.Text = "";
    c_textBoxSecCode.Text = ""; c_textBoxStartDate.Text = ""; c_textBoxEndDate.Text = ""; c_textBoxLink.Text = "";
    editingCard = false;
}
```

```
1 reference
private void a_buttonSave_Click(object sender, EventArgs e) //if existing item is being overwritten
{
    if (editingAddress == true)
    {
        string query = "UPDATE Address SET a_Title=@a_Title, a_Address 1=@a_Address1, " +
        "a_Address 2=@a_Address2, a_Address 3=@a_Address3, a_City/Town=@a_City/Town, a_County/State=@a_County/State, " +
        "a_Postcode=@a_Postcode, a_Link=@a_Link" +
        "WHERE AddressID=@AddressID";

        a_command = new OleDbCommand(query, connection);
        a_command.Parameters.AddWithValue("@a_Title", a_textBoxTitle.Text);
        a_command.Parameters.AddWithValue("@a_Address 1", a_textBoxAddress1.Text);
        a_command.Parameters.AddWithValue("@a_Address 2", a_textBoxAddress2.Text);
        a_command.Parameters.AddWithValue("@a_Address 3", a_textBoxAddress3.Text);
        a_command.Parameters.AddWithValue("@a_City/Town", a_textBoxCity.Text);
        a_command.Parameters.AddWithValue("@a_County/State", a_textBoxCounty.Text);
        a_command.Parameters.AddWithValue("@a_Postcode", a_textBoxPostcode.Text);
        a_command.Parameters.AddWithValue("@a_AddressID", Convert.ToInt32(a_dataTable.Rows[a_dataGridView.
            CurrentCell.RowIndex]["AddressID"]));

        connection.Open();
        a_command.ExecuteNonQuery();
        connection.Close();

        a_dataTable.Clear();
        GetAddresses();
    }
    else //if the item is new
    {
        string query = "INSERT INTO Address (a_Title, a_Address 1, a_Address 2, a_Address 3, a_City/Town, " +
            "a_County/State, a_Postcode) " +
            "VALUES (@a_Title ,@a_Address1, @a_Address2, @a_Address3, @a_City/Town, @a_County/State, @a_Postcode)";

        a_command = new OleDbCommand(query, connection);
        a_command.Parameters.AddWithValue("@a_Title", a_textBoxTitle.Text);
        a_command.Parameters.AddWithValue("@a_Address1", a_textBoxAddress1.Text);
        a_command.Parameters.AddWithValue("@a_Address2", a_textBoxAddress2.Text);
        a_command.Parameters.AddWithValue("@a_Address3", a_textBoxAddress3.Text);
        a_command.Parameters.AddWithValue("@a_City/Town", a_textBoxCity.Text);
        a_command.Parameters.AddWithValue("@a_County/State", a_textBoxCounty.Text);
        a_command.Parameters.AddWithValue("@a_Postcode", a_textBoxPostcode.Text);
        connection.Open();
        a_command.ExecuteNonQuery();
        connection.Close();

        a_dataTable.Clear();
        GetAddresses();
    }

    a_textBoxTitle.Text = ""; a_textBoxAddress1.Text = ""; a_textBoxAddress2.Text = ""; a_textBoxAddress3.Text = "";
    a_textBoxCity.Text = ""; a_textBoxCounty.Text = ""; a_textBoxPostcode.Text = ""; a_textBoxLink.Text = "";
    editingAddress = false;
}
```

Typo - ‘Password’ was changed to ‘Passwords’.

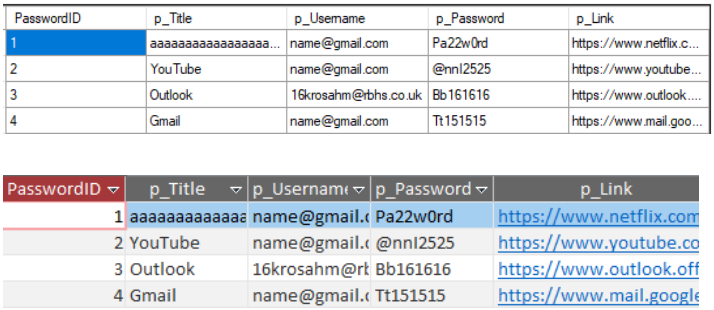
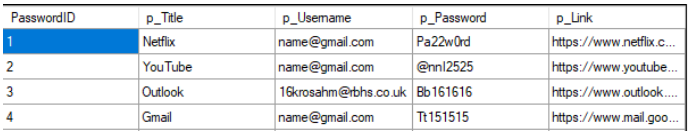
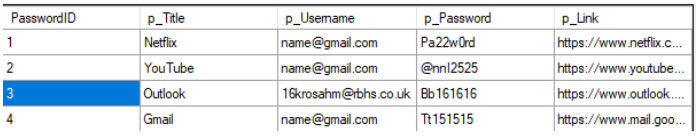
Each field name was made unique by adding prefixes. The SQL queries were ambiguous because there were multiple fields with the same name, so the query did not know which one to select from which table.

The query names did not need to be unique because they were declared as new ones each time they were run, so they were all named 'query'.

CODE TEST 2

- Button Clicks

Test Data -	Passwords: - a* 40 (This is to test how long string is displayed - in Access cells) - Netflix - name@gmail.com - Pa22w0rd - https://www.netflix.com/login/ - YouTube - name@gmail.com - @nnI2525 - https://www.youtube.com/ - Outlook - 16krosahm@rbhs.co.uk - Bb161616 - https://www.outlook.office.com/login/ - Gmail - name@gmail.com - Tt151515 - https://mail.google.com/mail/

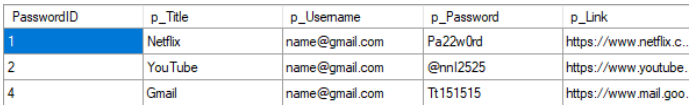
Save -	
○ If a row isn't selected first, the next empty row of the Access table is where the information is saved. ○ ID column must also update automatically.	 ☑Success
○ If a row is selected, that is where information is overwritten.	 ○ Title changed to 'Netflix'. ☑Success
Delete –	
When this is clicked, the data grid will remove that row permanently, in the data grid and in Access.	 ○ 3rd row selected.

PasswordID	p_Title	p_Username	p_Password	p_Link
1	aaaaaaaaaaaaaaaa...	name@gmail.com	Pa22w0rd	https://www.netflix.c...
2	YouTube	name@gmail.com	@nnl2525	https://www.youtube...
3	Outlook	16krosahm@rbhs.co.uk	Bb161616	https://www.outlook....
4	Gmail	name@gmail.com	Tt151515	https://www.mail.goo...

PasswordID	p_Title	p_Username	p_Password	p_Link
1	aaaaaaaaaaaaa	name@gmail.(	Pa22w0rd	https://www.netflix.com
2	YouTube	name@gmail.(	@nnl2525	https://www.youtube.co
3	Outlook	16krosahm@rb	Bb161616	https://www.outlook.off
4	Gmail	name@gmail.(	Tt151515	https://www.mail.google

PasswordID	p_Title	p_Username	p_Password	p_Link
1	Netflix	name@gmail.com	Pa22w0rd	https://www.netflix.c...
2	YouTube	name@gmail.com	@nnl2525	https://www.youtube...
3	Outlook	16krosahm@rbhs.co.uk	Bb161616	https://www.outlook....
4	Gmail	name@gmail.com	Tt151515	https://www.mail.goo...

PasswordID	p_Title	p_Username	p_Password	p_Link
1	Netflix	name@gmail.com	Pa22w0rd	https://www.netflix.c...
2	YouTube	name@gmail.com	@nnl2525	https://www.youtube...
3	Outlook	16krosahm@rbhs.co.uk	Bb161616	https://www.outlook....
4	Gmail	name@gmail.com	Tt151515	https://www.mail.goo...

	 ☑Success

PasswordID	p_Title	p_Username	p_Password	p_Link
1	Netflix	name@gmail.com	Pa22w0rd	https://www.netflix.c..
2	YouTube	name@gmail.com	@nnl2525	https://www.youtube.
4	Gmail	name@gmail.com	Tt151515	https://www.mail.goo.

Bank Accounts

CODED SOLUTION

```
// for bank account
1 reference
private void ba_buttonDelete_Click(object sender, EventArgs e)
{
    try
    {
        string query = "Delete FROM BankAccounts WHERE BankAccountID=@BankAccountID";

        ba_command = new OleDbCommand(query, connection);
        ba_command.Parameters.AddWithValue("@BankAccountID", Convert.ToInt32(ba_dataTable.Rows[ba_dataGridView.
            CurrentCell.RowIndex]["BankAccountID"]));

        connection.Open();
        ba_command.ExecuteNonQuery();
        connection.Close();

        ba_dataTable.Clear();
        GetBankAccounts();
    }
    catch (Exception) { MessageBox.Show("Select row to delete."); }
}
```

```
1 reference
private void ba_dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        ba_textBoxTitle.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
        ba_textBoxUsername.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[2].ToString();
        ba_textBoxPassword.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[3].ToString();
        ba_textBoxAccNo.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[4].ToString();
        ba_textBoxSortCode.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[5].ToString();
        ba_textBoxLink.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[6].ToString();
        editingBankAccount = true;
    }
    catch (Exception) { MessageBox.Show("Selected empty row."); }
}

1 reference
private void ba_buttonNew_Click(object sender, EventArgs e)
{
    ba_textBoxTitle.Text = ""; ba_textBoxUsername.Text = ""; ba_textBoxPassword.Text = "";
    ba_textBoxAccNo.Text = ""; ba_textBoxSortCode.Text = ""; ba_textBoxLink.Text = "";
    editingBankAccount = false;
}
```

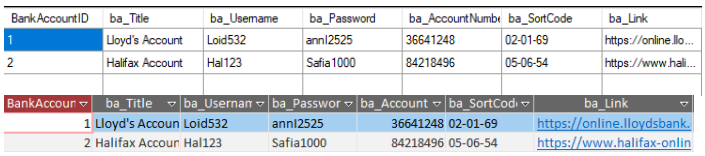
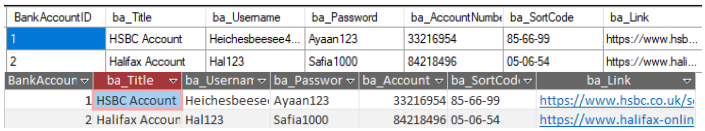
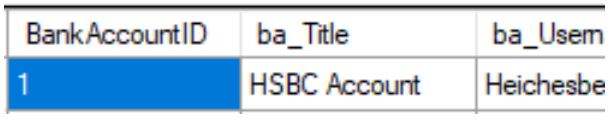
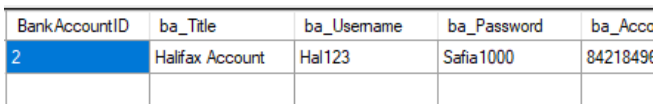
```
1 reference
private void ba_buttonSave_Click(object sender, EventArgs e) //if existing item is being overwritten
{
    if (editingBankAccount == true)
    {
        string query = "UPDATE BankAccounts SET ba_Title=@ba_Title, ba_Username=@ba_Username, ba_Password=@ba_Password, " +
            "ba_AccountNumber=@ba_AccountNumber, ba_SortCode=@ba_SortCode, ba_Link=@ba_Link " +
            "WHERE BankAccountID=@BankAccountID";

        ba_command = new OleDbCommand(query, connection);
        ba_command.Parameters.AddWithValue("@ba_Title", ba_textBoxTitle.Text);
        ba_command.Parameters.AddWithValue("@ba_Username", ba_textBoxUsername.Text);
        ba_command.Parameters.AddWithValue("@ba_Password", ba_textBoxPassword.Text);
        ba_command.Parameters.AddWithValue("@ba_AccountNumber", ba_textBoxAccNo.Text);
        ba_command.Parameters.AddWithValue("@ba_SortCode", ba_textBoxSortCode.Text);
        ba_command.Parameters.AddWithValue("@ba_Link", ba_textBoxLink.Text);
        ba_command.Parameters.AddWithValue("@BankAccountID", Convert.ToInt32(ba_dataTable.Rows[ba_dataGridView.
            CurrentCell.RowIndex]["BankAccountID"]));

        connection.Open();
        ba_command.ExecuteNonQuery();
        connection.Close();

        ba_dataTable.Clear();
        GetBankAccounts();
    }
    else //if the item is new
    {
        string query = "INSERT INTO BankAccounts (ba_Title, ba_Username, ba_Password, ba_AccountNumber, ba_SortCode, ba_Link) " +
            "VALUES (@ba_Title, @ba_Username, @ba_Password, @ba_AccountNumber, @ba_SortCode, @ba_Link)";

        ba_command = new OleDbCommand(query, connection);
        ba_command.Parameters.AddWithValue("@ba_Title", ba_textBoxTitle.Text);
        ba_command.Parameters.AddWithValue("@ba_Username", ba_textBoxUsername.Text);
        ba_command.Parameters.AddWithValue("@ba_Password", ba_textBoxPassword.Text);
        ba_command.Parameters.AddWithValue("@ba_AccountNumber", ba_textBoxAccNo.Text);
        ba_command.Parameters.AddWithValue("@ba_SortCode", ba_textBoxSortCode.Text);
        ba_command.Parameters.AddWithValue("@ba_Link", ba_textBoxLink.Text);
        connection.Open();
        ba_command.ExecuteNonQuery();
        connection.Close();

        ba_dataTable.Clear();
        GetBankAccounts();
    }

    ba_textBoxTitle.Text = ""; ba_textBoxUsername.Text = ""; ba_textBoxPassword.Text = "";
    ba_textBoxAccNo.Text = ""; ba_textBoxSortCode.Text = ""; ba_textBoxLink.Text = "";
    editingBankAccount = false;
}
```

CODE TEST

- Button Clicks

Test Data -	<u>Bank Accounts:</u> - Lloyd's Account - Loid532 - annl2525 - 36641248

	- 02-01-69 - https://online.lloydsbank.co.uk/personal/logon/login.jsp - Halifax Account - Hal123 - Safia1000 - 84218496 - 05-06-54 - https://www.halifax-online.co.uk/personal/logon/login.jsp - HSBC Account - Heichesbeesee456 - Ayaan123 - 33216954 - 85-66-99 - https://www.hsbc.co.uk/security/
Save -	

○ If a row isn't selected first, the next empty row of the Access table is where the information is saved. ○ The ID column must also update automatically.	 ☑Success
○ If a row is selected, that is where information is overwritten.	 ☑Success
Delete –	
When this is clicked, the data grid will remove that row permanently, in the data grid and in Access.	 ○ 1st row selected.  ☑Success

Cards

CODED SOLUTION

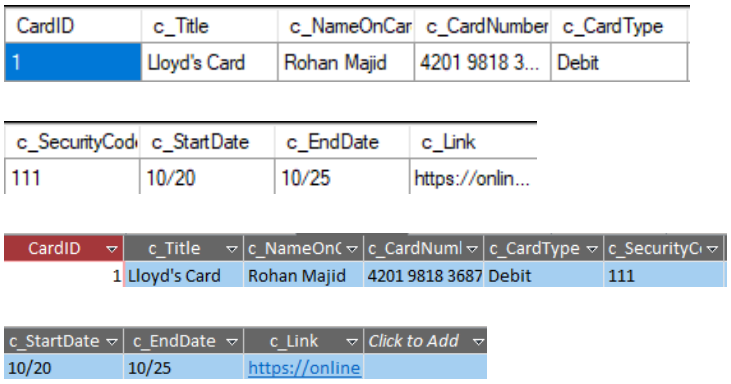
```
// for cards
1 reference
private void c_buttonDelete_Click(object sender, EventArgs e)
{
    try
    {
        string query = "Delete FROM Cards WHERE CardID=@CardID";

        c_command = new OleDbCommand(query, connection);
        c_command.Parameters.AddWithValue("@CardID", Convert.ToInt32(c_dataTable.Rows[c_dataGridView.
            CurrentCell.RowIndex]["CardID"]));

        connection.Open();
        c_command.ExecuteNonQuery();
        connection.Close();

        c_dataTable.Clear();
        GetCards();
    }
    catch (Exception) { MessageBox.Show("Select row to delete."); }
}

1 reference
private void c_dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        c_textBoxTitle.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
        c_textBoxNameOnCard.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[2].ToString();
        c_textBoxCardNo.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[3].ToString();
        c_textBoxCardType.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[4].ToString();
        c_textBoxSecCode.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[5].ToString();
        c_textBoxStartDate.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[6].ToString();
        c_textBoxEndDate.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[7].ToString();
        c_textBoxLink.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[8].ToString();
        editingCard = true;
    }
    catch (Exception) { MessageBox.Show("Selected empty row."); }
}

1 reference
private void c_buttonNew_Click(object sender, EventArgs e)
{
    c_textBoxTitle.Text = ""; c_textBoxNameOnCard.Text = ""; c_textBoxCardNo.Text = ""; c_textBoxCardType.Text = "";
    c_textBoxSecCode.Text = ""; c_textBoxStartDate.Text = ""; c_textBoxEndDate.Text = ""; c_textBoxLink.Text = "";
    editingCard = false;
}
```

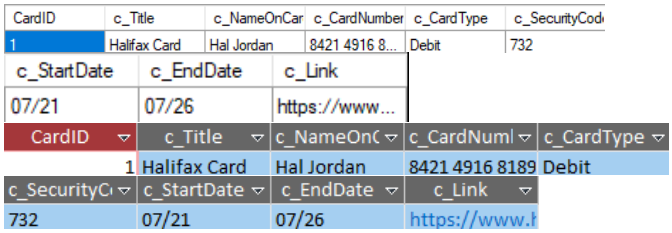
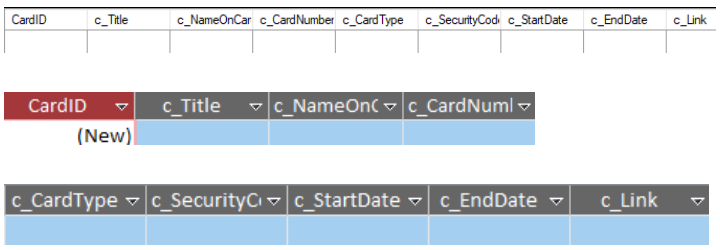
```
1 reference
private void c_buttonSave_Click(object sender, EventArgs e)
{
    if (editingCard == true) //if existing item is being overwritten
    {
        string query = "UPDATE Cards SET c_Title=@c_Title, c_NameOnCard=@c_NameOnCard, c_CardNumber=@c_CardNumber, " +
            "c_CardType=@c_CardType, c_SecurityCode=@c_SecurityCode, c_StartDate=@c_StartDate, c_EndDate=@c_EndDate, c_Link=@c_Link " +
            "WHERE CardID=@CardID";

        c_command = new OleDbCommand(query, connection);
        c_command.Parameters.AddWithValue("@c_Title", c_textBoxTitle.Text);
        c_command.Parameters.AddWithValue("@c_NameOnCard", c_textBoxNameOnCard.Text);
        c_command.Parameters.AddWithValue("@c_CardNumber", c_textBoxCardNo.Text);
        c_command.Parameters.AddWithValue("@c_CardType", c_textBoxCardType.Text);
        c_command.Parameters.AddWithValue("@c_SecurityCode", c_textBoxSecCode.Text);
        c_command.Parameters.AddWithValue("@c_StartDate", c_textBoxStartDate.Text);
        c_command.Parameters.AddWithValue("@c_EndDate", c_textBoxEndDate.Text);
        c_command.Parameters.AddWithValue("@c_Link", c_textBoxLink.Text);
        c_command.Parameters.AddWithValue("@CardID", Convert.ToInt32(c_dataTable.Rows[c_dataGridView.CurrentCell.
            RowIndex]["CardID"]));
        connection.Open();
        c_command.ExecuteNonQuery();
        connection.Close();

        c_dataTable.Clear();
        GetCards();
    }
    else //if the item is new
    {
        string query = "INSERT INTO Cards (c_Title, c_NameOnCard, c_CardNumber, c_CardType, c_SecurityCode, c_StartDate, " +
            "c_EndDate, c_Link) " +
            "VALUES (@c_Title, @c_NameOnCard, @c_CardNumber, @c_CardType, @c_SecurityCode, @c_StartDate, @c_EndDate, @c_Link)";

        c_command = new OleDbCommand(query, connection);
        c_command.Parameters.AddWithValue("@c_Title", c_textBoxTitle.Text);
        c_command.Parameters.AddWithValue("@c_NameOnCard", c_textBoxNameOnCard.Text);
        c_command.Parameters.AddWithValue("@c_CardNumber", c_textBoxCardNo.Text);
        c_command.Parameters.AddWithValue("@c_CardType", c_textBoxCardType.Text);
        c_command.Parameters.AddWithValue("@c_SecurityCode", c_textBoxSecCode.Text);
        c_command.Parameters.AddWithValue("@c_StartDate", c_textBoxStartDate.Text);
        c_command.Parameters.AddWithValue("@c_EndDate", c_textBoxEndDate.Text);
        c_command.Parameters.AddWithValue("@c_Link", c_textBoxLink.Text);
        connection.Open();
        c_command.ExecuteNonQuery();
        connection.Close();

        c_dataTable.Clear();
        GetCards();
    }

    c_textBoxTitle.Text = ""; c_textBoxNameOnCard.Text = ""; c_textBoxCardNo.Text = ""; c_textBoxCardType.Text = "";
    c_textBoxSecCode.Text = ""; c_textBoxStartDate.Text = ""; c_textBoxEndDate.Text = ""; c_textBoxLink.Text = "";
    editingCard = false;
}
```

CODE TEST

- Button Clicks

Test Data -	<u>Cards:</u> - Lloyd's Card - Rohan Majid - 4201 9818 3687 3309

	- Debit - 111 - 10/20 - 10/25 - https://online.lloydsbank.co.uk/personal/logon/login.jsp - Halif ax Card - Hal Jordan - 8421 4916 8189 7899 - Debit - 732 - 07/21 - 07/26 - https://www.halifax-online.co.uk/personal/logon/login.jsp
Save -	
○ If a row isn't selected first, the next empty row of the Access table is where the information is saved. ○ The ID column must also update automatically.	 ☑Success

○ If a row is selected, that is where information is overwritten.	 ☑Success
Delete –	
When this is clicked, the data grid will remove that row permanently, in the data grid and in Access.	 ☑Success

Addresses

CODED SOLUTION 1

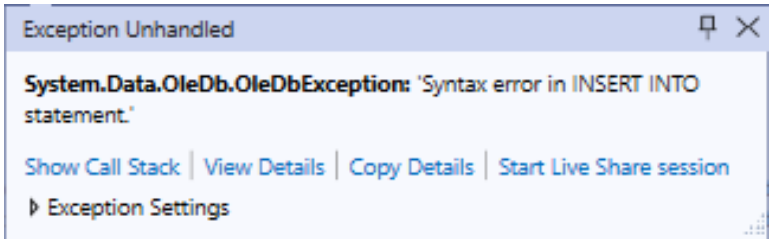
```
// for addresses
1 reference
private void a_buttonDelete_Click(object sender, EventArgs e)
{
    try
    {
        string query = "Delete FROM Addresses WHERE AddressID=@AddressID";

        a_command = new OleDbCommand(query, connection);
        a_command.Parameters.AddWithValue("@AddressID", Convert.ToInt32(a_dataTable.Rows[a_dataGridView.
            CurrentCell.RowIndex]["AddressID"]));

        connection.Open();
        a_command.ExecuteNonQuery();
        connection.Close();

        a_dataTable.Clear();
        GetAddresses();
    }
    catch (Exception) { MessageBox.Show("Select row to delete."); }
}

1 reference
private void a_dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        a_textBoxTitle.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
        a_textBoxAddress1.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[2].ToString();
        a_textBoxAddress2.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[3].ToString();
        a_textBoxAddress3.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[4].ToString();
        a_textBoxCity.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[5].ToString();
        a_textBoxCounty.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[6].ToString();
        a_textBoxPostcode.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[7].ToString();
        a_textBoxLink.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[8].ToString();
        editingAddress = true;
    }
    catch (Exception) { MessageBox.Show("Selected empty row."); }
}

1 reference
private void a_buttonNew_Click(object sender, EventArgs e)
{
    a_textBoxTitle.Text = ""; a_textBoxAddress1.Text = ""; a_textBoxAddress2.Text = ""; a_textBoxAddress3.Text = "";
    a_textBoxCity.Text = ""; a_textBoxCounty.Text = ""; a_textBoxPostcode.Text = ""; a_textBoxLink.Text = "";
    editingAddress = false;
}
```

```
1 reference
private void a_buttonSave_Click(object sender, EventArgs e) //if existing item is being overwritten
{
    if (editingAddress == true)
    {
        string query = "UPDATE Address SET a_Title=@a_Title, a_Address1=@a_Address1, " +
        "a_Address2=@a_Address2, a_Address3=@a_Address3, a_City/Town=@a_City/Town, a_County/State=@a_County/State, " +
        "a_Postcode=@a_Postcode, a_Link=@a_Link" +
        "WHERE AddressID=@AddressID";

        a_command = new OleDbCommand(query, connection);
        a_command.Parameters.AddWithValue("@a_Title", a_textBoxTitle.Text);
        a_command.Parameters.AddWithValue("@a_Address 1", a_textBoxAddress1.Text);
        a_command.Parameters.AddWithValue("@a_Address 2", a_textBoxAddress2.Text);
        a_command.Parameters.AddWithValue("@a_Address 3", a_textBoxAddress3.Text);
        a_command.Parameters.AddWithValue("@a_City/Town", a_textBoxCity.Text);
        a_command.Parameters.AddWithValue("@a_County/State", a_textBoxCounty.Text);
        a_command.Parameters.AddWithValue("@a_Postcode", a_textBoxPostcode.Text);
        a_command.Parameters.AddWithValue("@a_AddressID", Convert.ToInt32(a_dataTable.Rows[a_dataGridView.
            CurrentCell.RowIndex]["AddressID"]));
        connection.Open();
        a_command.ExecuteNonQuery();
        connection.Close();

        a_dataTable.Clear();
        GetAddresses();
    }
    else //if the item is new
    {
        string query = "INSERT INTO Addresses (a_Title, a_Address1, a_Address2, a_Address3, a_City/Town, " +
            "a_County/State, a_Postcode) " +
            "VALUES (@a_Title ,@a_Address1, @a_Address2, @a_Address3, @a_City/Town, @a_County/State, @a_Postcode)";

        a_command = new OleDbCommand(query, connection);
        a_command.Parameters.AddWithValue("@a_Title", a_textBoxTitle.Text);
        a_command.Parameters.AddWithValue("@a_Address1", a_textBoxAddress1.Text);
        a_command.Parameters.AddWithValue("@a_Address2", a_textBoxAddress2.Text);
        a_command.Parameters.AddWithValue("@a_Address3", a_textBoxAddress3.Text);
        a_command.Parameters.AddWithValue("@a_City/Town", a_textBoxCity.Text);
        a_command.Parameters.AddWithValue("@a_County/State", a_textBoxCounty.Text);
        a_command.Parameters.AddWithValue("@a_Postcode", a_textBoxPostcode.Text);
        connection.Open();
        a_command.ExecuteNonQuery();
        connection.Close();

        a_dataTable.Clear();
        GetAddresses();
    }

    a_textBoxTitle.Text = ""; a_textBoxAddress1.Text = ""; a_textBoxAddress2.Text = ""; a_textBoxAddress3.Text = "";
    a_textBoxCity.Text = ""; a_textBoxCounty.Text = ""; a_textBoxPostcode.Text = ""; a_textBoxLink.Text = "";
    editingAddress = false;
}
```

CODE TEST 1

- Button Clicks

Test Data -	Addresses:

	- School Address (or 6th Form or RBHS) - 100 Lever Park Avenue - Blackrod - Horwich - Bolton - Lancashire - BL6 7RU - https://www.google.com/maps
Save -	
○ If a row isn't selected, the next empty row of the Access table is where the information is saved. ○ The ID column must also update automatically.	 ○ The error is because the program can't tell if '/' is a special character or part of the name. ☑Success

CODED SOLUTION 2

```
private void a_buttonSave_Click(object sender, EventArgs e) //if existing item is being overwritten
{
    if (editingAddress == true)
    {
        string query = "UPDATE Addresses SET a_Title=@a_Title, a_Address1=@a_Address1, " +
            "a_Address2=@a_Address2, a_Address3=@a_Address3, a_CityTown=@a_CityTown, a_CountyState=@a_CountyState, " +
            "a_Postcode=@a_Postcode, a_Link=@a_Link " +
            "WHERE AddressID=@AddressID";

        a_command = new OleDbCommand(query, connection);
        a_command.Parameters.AddWithValue("@a_Title", a_textBoxTitle.Text);
        a_command.Parameters.AddWithValue("@a_Address1", a_textBoxAddress1.Text);
        a_command.Parameters.AddWithValue("@a_Address2", a_textBoxAddress2.Text);
        a_command.Parameters.AddWithValue("@a_Address3", a_textBoxAddress3.Text);
        a_command.Parameters.AddWithValue("@a_CityTown", a_textBoxCity.Text);
        a_command.Parameters.AddWithValue("@a_CountyState", a_textBoxCounty.Text);
        a_command.Parameters.AddWithValue("@a_Postcode", a_textBoxPostcode.Text);
        a_command.Parameters.AddWithValue("@a_Link", a_textBoxLink.Text);
        a_command.Parameters.AddWithValue("@a_AddressID", Convert.ToInt32(a_dataTable.Rows[a_dataGridView.
            CurrentCell.RowIndex]["AddressID"]));
        connection.Open();
        a_command.ExecuteNonQuery();
        connection.Close();

        a_dataTable.Clear();
        GetAddresses();
    }
```

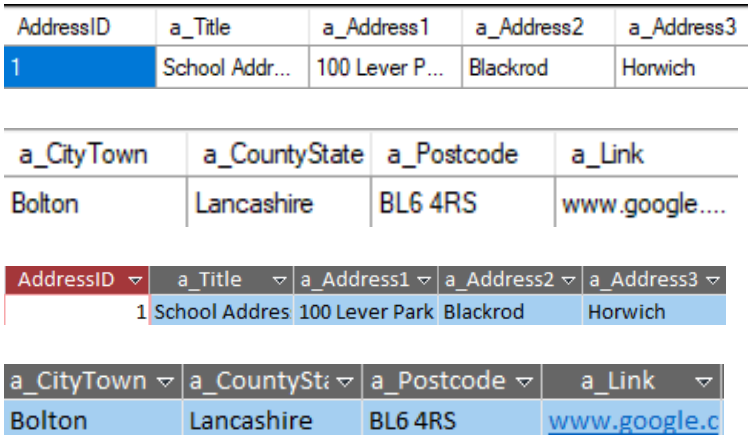
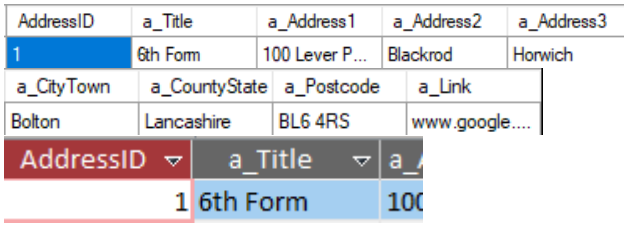
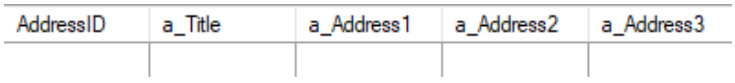
```
else //if the item is new
{
    string query = "INSERT INTO Addresses (a_Title, a_Address1, a_Address2, a_Address3, a_CityTown, a_CountyState, a_Postcode, a_Link) " +
        "VALUES (@a_Title, @a_Address1, @a_Address2, @a_Address3, @a_CityTown, @a_CountyState, @a_Postcode, @a_Link)";

    a_command = new OleDbCommand(query, connection);
    a_command.Parameters.AddWithValue("@a_Title", a_textBoxTitle.Text);
    a_command.Parameters.AddWithValue("@a_Address1", a_textBoxAddress1.Text);
    a_command.Parameters.AddWithValue("@a_Address2", a_textBoxAddress2.Text);
    a_command.Parameters.AddWithValue("@a_Address3", a_textBoxAddress3.Text);
    a_command.Parameters.AddWithValue("@a_CityTown", a_textBoxCity.Text);
    a_command.Parameters.AddWithValue("@a_CountyState", a_textBoxCounty.Text);
    a_command.Parameters.AddWithValue("@a_Postcode", a_textBoxPostcode.Text);
    a_command.Parameters.AddWithValue("@a_Link", a_textBoxLink.Text);
    connection.Open();
    a_command.ExecuteNonQuery();
    connection.Close();

    a_dataTable.Clear();
    GetAddresses();
}

a_textBoxTitle.Text = ""; a_textBoxAddress1.Text = ""; a_textBoxAddress2.Text = ""; a_textBoxAddress3.Text = "";
a_textBoxCity.Text = ""; a_textBoxCounty.Text = ""; a_textBoxPostcode.Text = ""; a_textBoxLink.Text = "";
editingAddress = false;
```

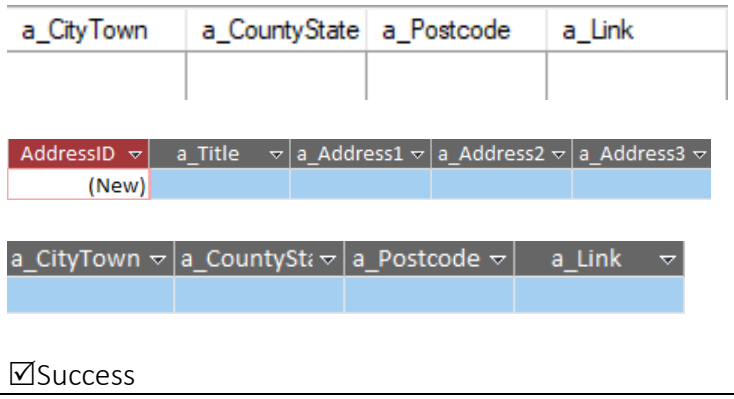
The ‘/’ character is no longer in the names.

CODE TEST 2

- Button Clicks

Test Data -	<u>Addresses:</u> - School Address (or 6th Form or RBHS)

	- 100 Lever Park Avenue - Blackrod - Horwich - Bolton - Lancashire - BL6 7RU - https://www.google.com/maps
Save -	
○ If a row isn't selected, the next empty row of the Access table is where the information is saved. ○ The ID column must also update automatically.	 ☑Success
○ If a row is selected, that is where information is overwritten.	 ☑Success
Delete –	
When this is clicked, the data grid will	

remove that row permanently, in the data grid and in Access.	 ☑Success

DATA VALIDATION

The data in each database will be saved as text, except for notes (as long text) and IDs (as auto-numbers) and links (as hyperlinks). This is because the data will be stored as string (text) in the data grids and will be copied and pasted to login pages as string.

EVALUATION

In this section, I will review my initial plan and intentions for the project and see how well I was able to achieve them by considering any feedback from stakeholders and successful or failed objectives.

USER FEEDBACK & TESTING

In this section, stakeholders will practice black box testing. This is when the person testing has no knowledge of the internal structure of the application.

The stakeholders will be the same people involved in the questionnaire (ages 13, 17(x3), 18, 30, 44, and 47). I'll observe and record how they interact with the project, and they must inform me on how well the project meets the original requirements. I will send a copy of the app to them to run on their personal laptop, the teacher's computer, or the computer room in my sixth form.

They will use the app without help from me to see how user-friendly it will be when I complete development. However, if they struggle to use it, I may give them prompts to help them. After they have tested the app, I will ask them about all my observations of them for further explanation of their behaviour.

Login/ Sign Up

Each user was asked about whether they would've preferred a login/ sign up system. Most users only used their personal computer, so they did not mind being restricted to one account. However, two were disappointed they couldn't sign in on other devices and access their passwords from there.

Passwords, Bank Accounts, Cards, Addresses, Notes

User Input	Observation & Explanation
Saving new password.	○ All users intuitively understood they should enter information in text boxes, then save them. ○ They thought the link boxes were irrelevant in some password types. ○ They wished they could annotate or include extra information about the password.
Loading existing password.	○ One user knew to double-click the data grid row to load it because he was familiar with CRUD designs. ○ The rest were initially unsure how to load data in the text boxes. I had to prompt them to double click on the text boxes.

	○ Some users tried to click the link in the data grid to launch it and failed.
Editing existing password.	○ They made the correct assumption to edit the text boxes and click save again.
Deleting a password	○ Most users intuitively understood they should single-click information in text boxes, then click delete. ○ One user double-clicked, then deleted. They thought the information was stored in the text boxes, not the data grid, so they wanted to load the information to the text boxes first. ○ That user was also confused as to why the information was still in the text boxes after the password was deleted. ○ Users were concerned they couldn't retrieve information after it was deleted if they deleted something accidently.
Data validating.	○ One user tried to enter text in text boxes only meant for numbers. They were expecting only numbers to be enterable. ○ The users said they were expecting the program to prompt them to include stronger passwords to their vault by adding capitals or special characters or an appropriate length.

Closing and reopening the vault.	○ They were asked to close and reopen the vault to test the save feature of the program. All their information was saved.

Password Generator

User Input	Observation & Explanation
Changing length.	○ All users slid the slider across the screen to their desired password lengths.
Copying password	○ Some of them altered the passwords slightly when copied. They added a word they preferred to be in the password.

FUNCTIONAL CRITERIA MET

(State the criteria, state if met, state how it can be in future development (e.g., how usability, maintenance can be improved)).

Appropriate menus that the user would expect for the password manager to have:	Success Level	How I succeeded /Further Development

○ The menus will be 16:9, for desktops.	☑Accomplished ☐Partially ☐Failed	○ Everyone nowadays has a modern PC or laptop that they can use the application on.
○ One panel will have the types of information stored by the user: Passwords, Bank Accounts, Cards, Addresses, and Secure Notes. Each one of these types will record information in a grid, pre-built by Visual Studios.	☑Accomplished ☐Partially ☐Failed	○ Visual Studio had built in tab controls. I used them instead of programming panels, which would have been a more familiar layout to the user. I could change the tab arrangement from horizontal to vertical, so they look like another panel.
○ The other panel will have the associated information in that type shown in a data grid.	☑Accomplished ☐Partially ☐Failed	○ The data grid feature made it easy to organise the user's information so they can view it easier.

These are the following things the user can input for each category of information. Each one will have their own textbox where the user can input it	Success	How I succeeded /Further Development

and have their own column in the data grid:		
○ Passwords: Title, Username, Password, Link ○ Bank Accounts: Title, Username, Password, Account Number, Sort Code, Link ○ Cards: Title, Name on Card, Card Number, Card Type, Security Code, Start Date, Finished Date, Link ○ Addresses: Title, Address 1, Address 2, Address 3, City/Town, County, Post Code, Link ○ Secure Notes: Title, Notes	☑Accomplished ☐Partially ☐Failed	○ The users said the app included every type of sensitive information they'd want to include.
○ The title is meant to be what the user chooses to call that password, address, or note for them to distinguish it better.	☐Accomplished ☑Partially ☐Failed	○ One user put 'Mr' in the title section, so it can be made cleared to the user what to put.
○ The link is the hyperlink that will be launched by the	☐Accomplished ☑Partially	○ The user could click and launch the link from the data grid to save time copying and pasting it.

user so that they can input the login information quicker. They may simply copy and paste it into the corresponding boxes.	☐Failed	○ The link box was left blank or seen as irrelevant by most users, and they wished for a note box to elaborate on the password type. So, a note box can be added instead of link boxes.

Overall, a help tab can be added to give the user basic descriptions of how to use the data grid and text boxes and how to use the buttons.

A login and sign-up screen to create a new PassVault account, or log into an existing one.	Success	How I succeeded /Further Development

○ It will require a master username (which will be the user's email) ○ It needs one master password. The user will only need to remember the master password to access all other passwords on their account. ○ The user creates a master account with the master username and the master password (this is how the password managers mentioned above work)	☐Accomplished ☐Partially ☑Failed	○ The users wished they were able to access the passwords from another device, so a login and sign-up page would be required for such a purpose. ○ The Access database could include another table for master accounts, containing the master username and master password, just like how I initially wanted. ○ The Access database could be stored on a server the program will access via the internet.

A hashing algorithm for the user's passwords.	Success	How I succeeded /Further Development
○ To prevent unauthorised access to the user's passwords in the database.	☐Accomplished ☑Partially ☐Failed	○ I programmed the hashing algorithm but failed to use it when saving passwords because the program was meant to save the hashed password to the database, but when the vault was closed and loaded again, it loaded the hashed version of the password, and they wouldn't be able to view the original password.

○ If the master password is inaccessible, a hacker cannot view the other passwords connected to that master username.	☐Accomplished ☐Partially ☑Failed	○ If I include the master account feature, the rest of the passwords saved to it will be secure. ○ The hashing algorithm is easier to implement in a login system because the user's hashed master password is saved in the database. ○ The user's input in the master password text box is then hashed and compared to any hashed passwords in the database.

random password function	Success	How I succeeded /Further Development
○ A random password function if the user wants to produce a new strong password.	☑Accomplished ☐Partially ☐Failed	○ All users were pleased with the range of lengths (8 – 16 characters), because they satisfy all websites.

Navigation	Success	How I succeeded /Further Development
○ The users find it simple and easy to navigate between the password types and the password generator.	☑Accomplished ☐Partially ☐Failed	○ All sub-menus were arranged in one tab panel at the top of the app. Each user knew to click on the tab they wanted to move onto.

○ If the user has multiple user profiles on their PC or laptop, each profile will have their own storage that cannot be shared across users, so they will effectively save each password to their own profile.	☑Accomplished ☐Partially ☐Failed	○ The users tried opening the app on different accounts on their desktops, and the passwords on the other account weren't displayed, so other users of their desktop won't see their sensitive information.

LIMITATIONS

- The lack of server space to store the passwords.
  - A login /sign up system couldn't be made.
  - The user can't access their data on other devices.

- The lack of data security.
  - The hashing algorithm could not be implemented because of no login system because hashed string cannot be reversed.
  - The data can be accessed by another user by gaining unauthorised access to the database.

- The lack of data validation.
  - The user will not be corrected if:
    - They enter a password that is too long or too short.
    - They use the same password multiple times.
    - They include irrelevant characters in a textbox (e.g., including characters in the card number box)
  - The user cannot interact with a hyperlink that they saved; if they click on it in the data grid, it does not open the web page.

- There is no recycling bin, so the user cannot retrieve anything deleted.

- The unfamiliar layout of the menu.
  - It follows a typical CRUD menu design. The menu has textboxes adjacent to the data grid, with each text box representing a column in the data grid. The user would normally edit the information directly on the data grid.

BIBLIOGRAPHY

Images –

LastPass Support - https://www.passwordmanager.com/nordpass-review/

NordPass Reviews from Website - https://www.passwordmanager.com/nordpass-review/

1Password Website Items Section - https://support.1password.com/1password-com-items/

Proton Support Section of Website - https://www.google.com/imgres?imgurl=https%3A%2F%2Fproton.me%2Fstatic%2Fd89a77f52758f91d6372f19354ab5187%2F8690f%2FLastPass-4.jpg&tbnid=AgKxDvEOeFE7UM&vet=10CEwQMyiLAWoXChMI-KT-25SG_gIVAAAAAB0AAAAAEBQ..i&imgrefurl=https%3A%2F%2Fproton.me%2Fsupport%2Flast-pass&docid=_2GDY_b3pdUuIM&w=792&h=560&q=lastpass%20login&safe=active&ved=0CEwQMyiLAWoXChMI-KT-25SG_gIVAAAAAB0AAAAAEBQ

Sarah Hension Blog - https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.sarahhenson.co.uk%2Fblog%2Fhow-to-remember-your-passwords-use-lastpass&psig=AOvVaw21i59uJzeRAefmdhrBBqCY&ust=1680351881926000&source=images&cd=vfe&ved=0CAwQjRxqFwoTCPik_tuUhv4CFQAAAAAdAAAAABAq

Vector Stock Website – https://www.vectorstock.com/royalty-free-vector/login-web-screen-with-blue-design-template-vector-10019125

https://www.vectorstock.com/royalty-free-vector/sign-up-screen-mobile-app-ui-for-registration-vector-34982486

APPENDIX

QUESTIONNAIRE

I am planning to create a personal information manager for my A-Level Computer Science project. Please answer the following questions so I can determine the best and most relevant features for the application.

State your age:

1) Do you usually use the same password for all your accounts?

- Always
- Often
- Sometimes
- No

2) Would you pay a subscription for a password manager? Please state a reason for your answer.

- Yes
- No

3) List any sensitive information you would store on a personal information manager:

4) Will you remember to record new passwords/ personal information to the application?

- Yes
- No

5) Do you feel comfortable saving passwords or personal information to web browsers like Google? Please state a reason for your answer.

- Yes
- No

6) Do you already use your phone's password storage services, such as iCloud Keychain, Google's password manager?

- Yes
- No

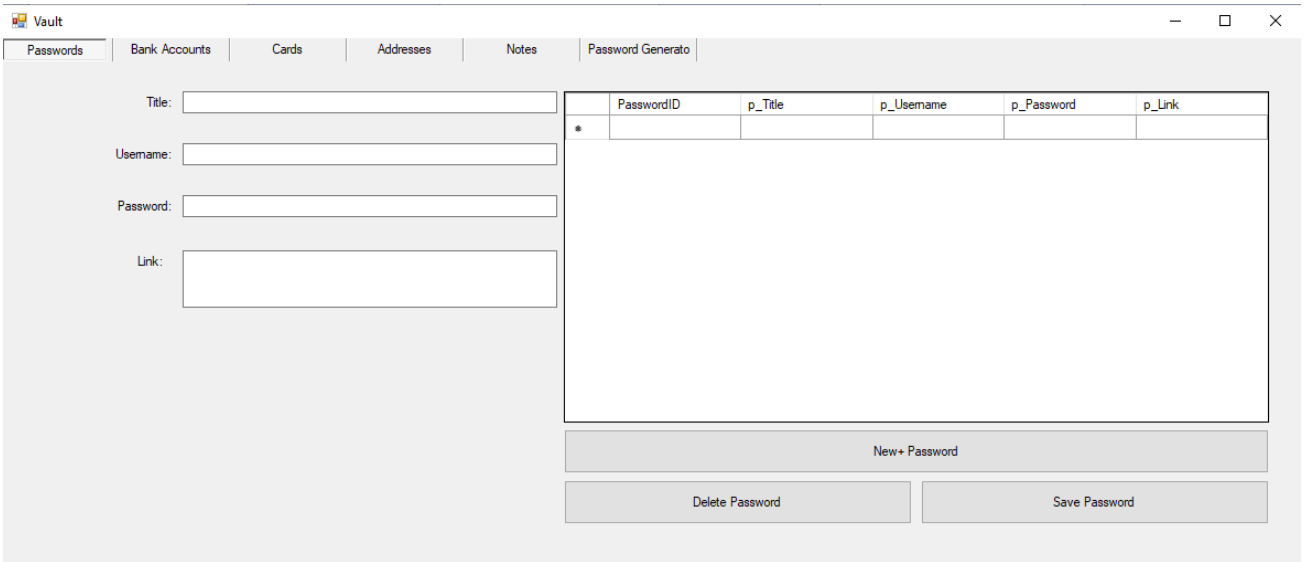
7) Do you use your PC or laptop's password storage services, such as Mac keychains, Chromebook password manager? Please state which one(s) you use.

- Yes
- No

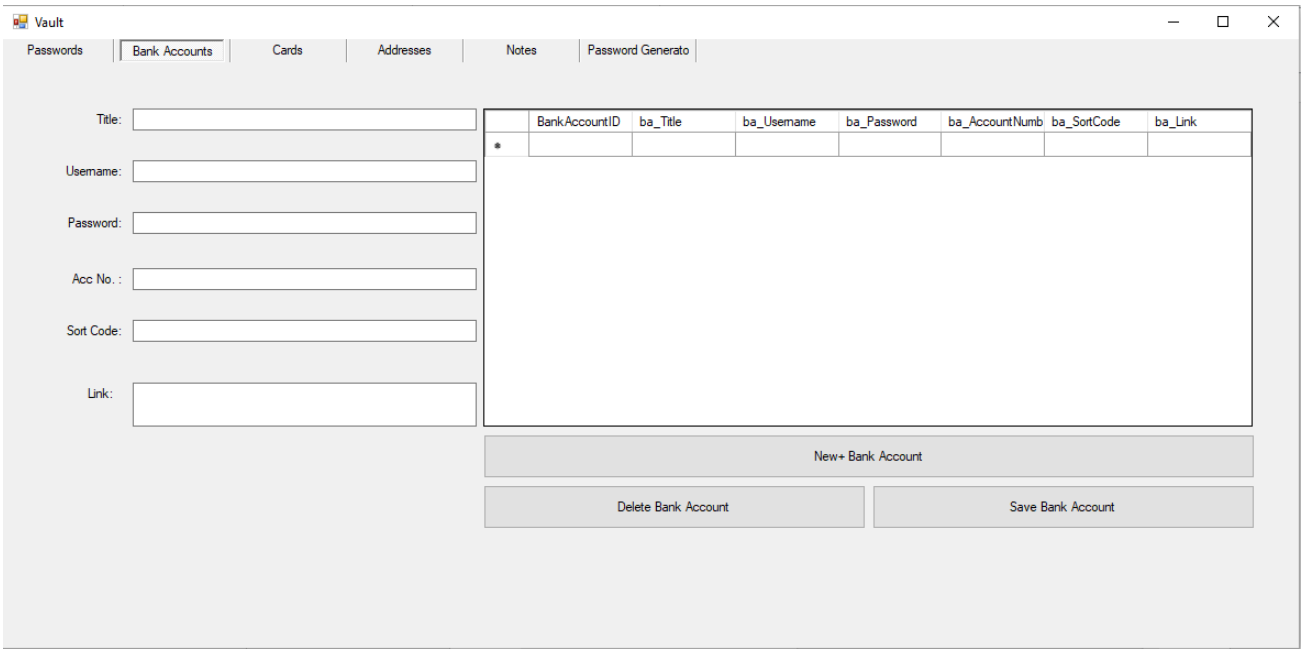
8) Would you trust a password manager to store sensitive information over web browsers?

- Yes
- No

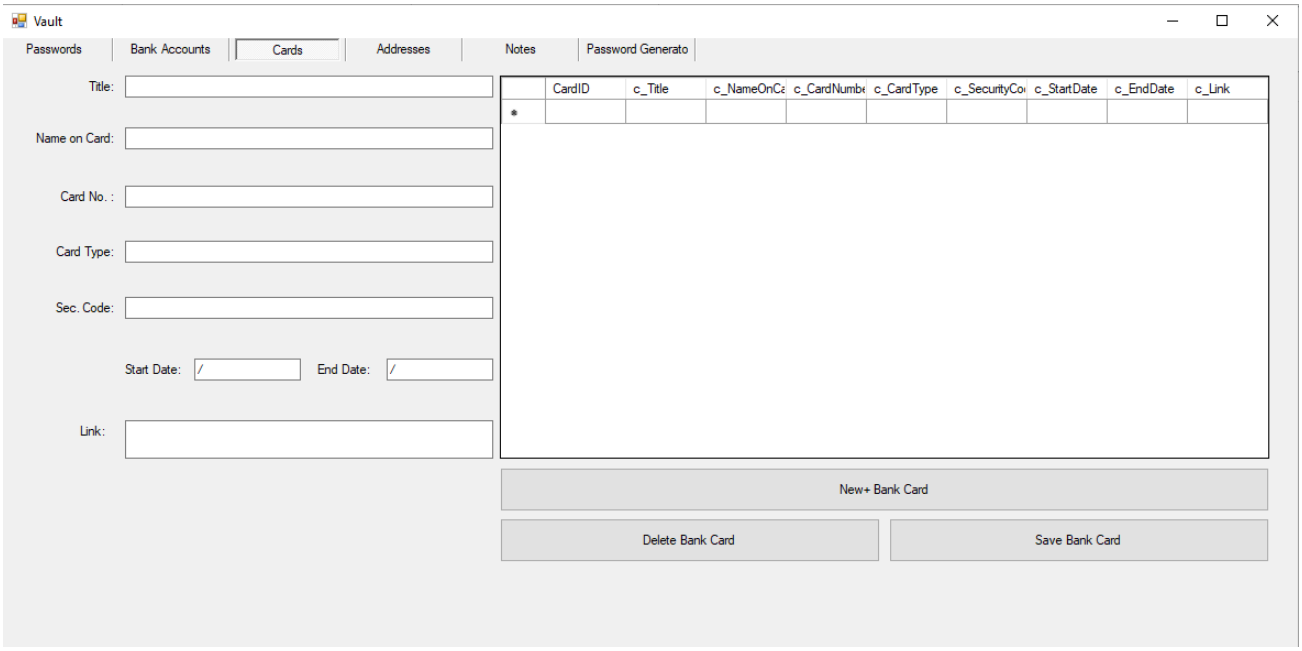
WINFOWS FORM APP DESIGN



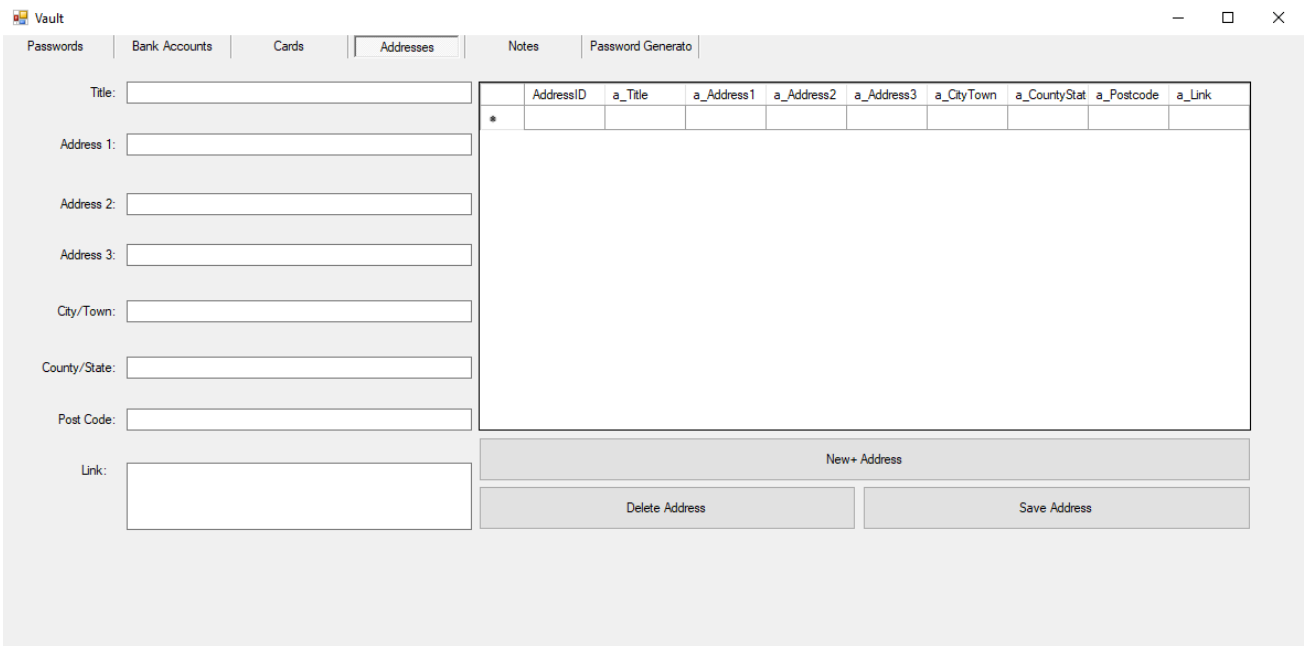
Passwords tab.



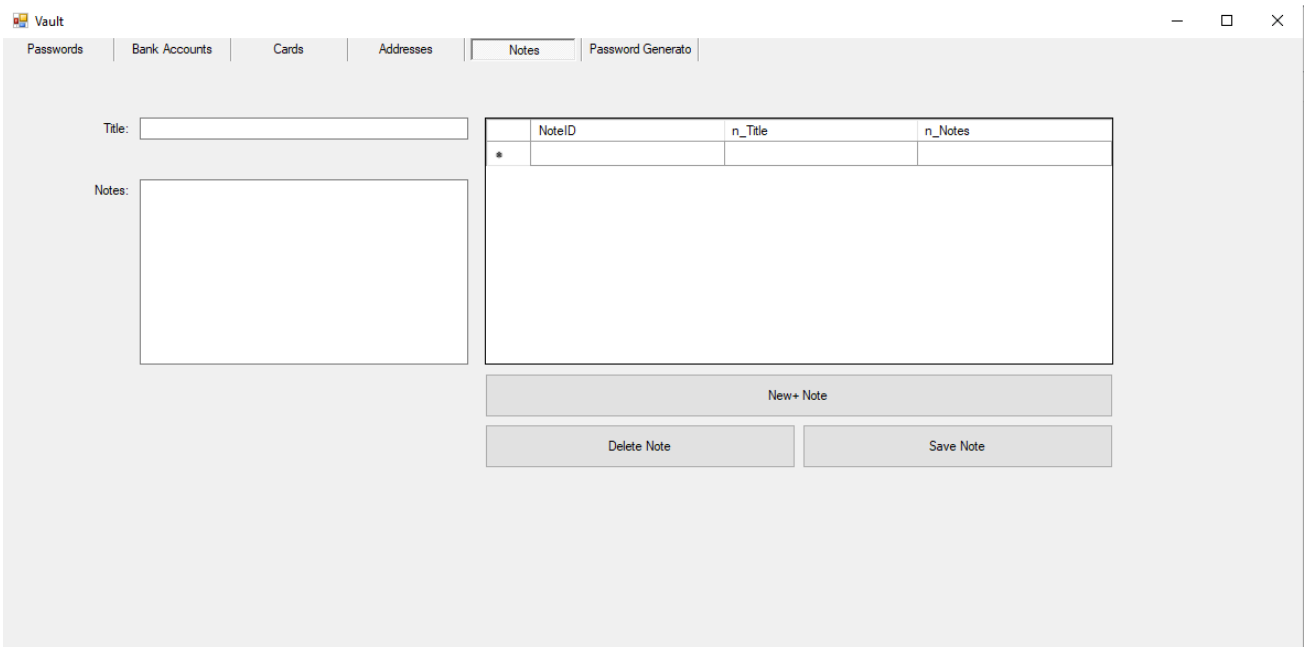
Bank accounts tab.



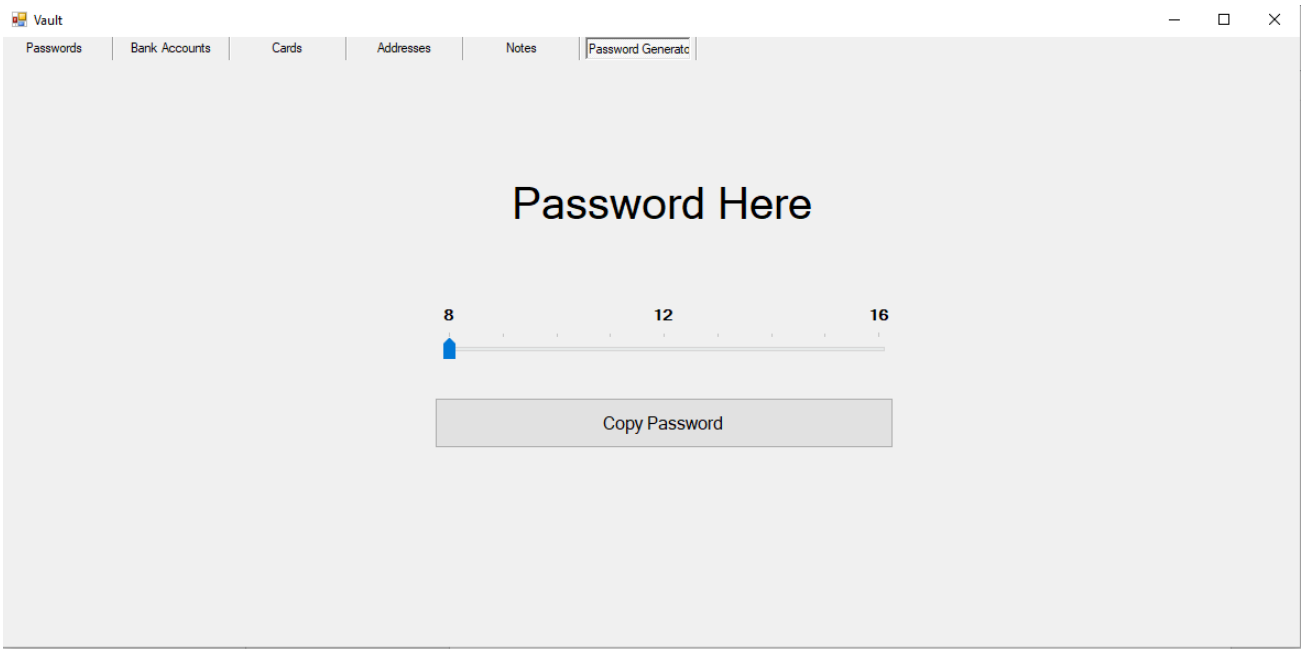
Cards tab.



Addresses tab.

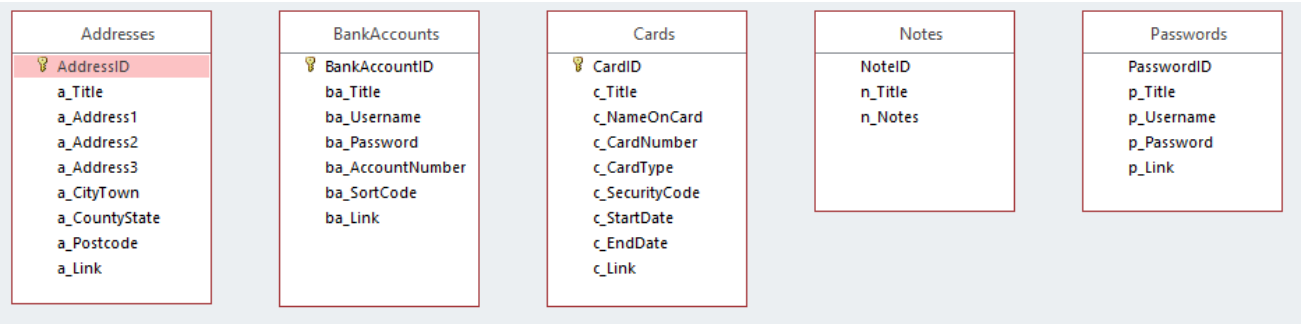


Notes tab.



Generate Password.

ACCESS DATABASE TABLES



The tables shown here are all linked to their respective tabs of the Vault. Each datagrid reads and writes data into the tables. This acts as the save feature of the vault.

COMPLETE PROGRAM CODE

Main Program

```
1   using System;
2   using System.Windows.Forms;
3
4   namespace PassVault_Multi_Menu_Version
5   {
        0 references
6       static class Program
7       {
8           /// <summary>
9           /// The main entry point for the application.
10          /// </summary>
11          [STAThread]
            0 references
12          static void Main()
13          {
14              Application.EnableVisualStyles();
15              Application.SetCompatibleTextRenderingDefault(false);
16              Application.Run(new Vault());
17          }
18      }
19  }
20
```

This section loads the Windows Form design and the Code for it, which are two seperate sections.

Vault Form

```
1   using System;
2   using System.Data;
3   using System.Data.OleDb;
4   using System.Security.Cryptography;
5   using System.Text;
6   using System.Windows.Forms;
7
8   namespace PassVault_Multi_Menu_Version
9   {
        3 references
10      public partial class Vault : Form
11      {
12
13          OleDbConnection connection = new OleDbConnection("Provider=Microsoft.ACE.OleDb.16.0;" +
14              " Data Source = D:\\PassVault\\PassVaultDatabase.accdb");
15
16          OleDbCommand p_command;
17          OleDbCommand ba_command;
18          OleDbCommand c_command;
19          OleDbCommand a_command;
20          OleDbCommand n_command;
21
22          OleDbDataAdapter p_dataAdapter;
23          OleDbDataAdapter ba_dataAdapter;
24          OleDbDataAdapter c_dataAdapter;
25          OleDbDataAdapter a_dataAdapter;
26          OleDbDataAdapter n_dataAdapter;
27
28          DataTable p_dataTable = new DataTable();
29          DataTable ba_dataTable = new DataTable();
30          DataTable c_dataTable = new DataTable();
31          DataTable a_dataTable = new DataTable();
32          DataTable n_dataTable = new DataTable();
33
34          bool editingPassword = false;
35          bool editingBankAccount = false;
36          bool editingCard = false;
37          bool editingAddress = false;
38          bool editingNote = false;
```

```
4 references
void GetPasswords()
{
    p_dataAdapter = new OleDbDataAdapter("SELECT *FROM Passwords", connection);
    connection.Open();
    p_dataAdapter.Fill(p_dataTable);
    p_dataGridView.DataSource = p_dataTable;
    connection.Close();
}

4 references
void GetBankAccounts()
{
    ba_dataAdapter = new OleDbDataAdapter("SELECT *FROM BankAccounts", connection);
    connection.Open();
    ba_dataAdapter.Fill(ba_dataTable);
    ba_dataGridView.DataSource = ba_dataTable;
    connection.Close();
}

4 references
void GetCards()
{
    c_dataAdapter = new OleDbDataAdapter("SELECT *FROM Cards", connection);
    connection.Open();
    c_dataAdapter.Fill(c_dataTable);
    c_dataGridView.DataSource = c_dataTable;
    connection.Close();
}

4 references
void GetAddresses()
{
    a_dataAdapter = new OleDbDataAdapter("SELECT *FROM Addresses", connection);
    connection.Open();
    a_dataAdapter.Fill(a_dataTable);
    a_dataGridView.DataSource = a_dataTable;
    connection.Close();
}

4 references
void GetNotes()
{
    n_dataAdapter = new OleDbDataAdapter("SELECT *FROM Notes", connection);
    connection.Open();
    n_dataAdapter.Fill(n_dataTable);
    n_dataGridView.DataSource = n_dataTable;
    connection.Close();
}
```

```
85
86
87
                1 reference
88              public Vault()
89              {
90                  InitializeComponent();
91              }
92
93
94
                1 reference
95              private void PassVault_Load(object sender, EventArgs e)
96              {
97                  GetPasswords();
98
99                  GetBankAccounts();
100
101                 GetCards();
102
103                 GetAddresses();
104
105                 GetNotes();
106             }
107
108
```

```
110             // for notes
                1 reference
111             private void n_buttonDelete_Click(object sender, EventArgs e)
112             {
113                 try
114                 {
115                     string query = "Delete FROM Notes WHERE NoteID=@NoteID";
116
117                     n_command = new OleDbCommand(query, connection);
118                     n_command.Parameters.AddWithValue("@NoteID", Convert.ToInt32(n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex]["NoteID"]));
119
120                     connection.Open();
121                     n_command.ExecuteNonQuery();
122                     connection.Close();
123
124                     n_dataTable.Clear();
125                     GetNotes();
126                 }
127                 catch (Exception) { MessageBox.Show("Select row to delete."); }
128             }
129
                1 reference
130             private void n_dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
131             {
132                 try
133                 {
134                     n_textBoxTitle.Text = n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
135                     n_textBoxNotes.Text = n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex].ItemArray[2].ToString();
136                     editingNote = true;
137                 }
138                 catch (Exception)
139                 { MessageBox.Show("Selected empty row."); }
140             }
```

```
1 reference
private void n_buttonNew_Click(object sender, EventArgs e)
{
    n_textBoxTitle.Text = ""; n_textBoxNotes.Text = "";
    editingNote = false;
}

1 reference
private void n_buttonSave_Click(object sender, EventArgs e) //if existing item is being overwritten
{
    if (editingNote == true)
    {
        string query = "UPDATE Notes SET n_Title=@n_Title, n_Notes=@n_Notes " +
            "WHERE NoteID=@NoteID";

        n_command = new OleDbCommand(query, connection);
        n_command.Parameters.AddWithValue("@n_Title", n_textBoxTitle.Text);
        n_command.Parameters.AddWithValue("@n_Notes", n_textBoxNotes.Text);
        n_command.Parameters.AddWithValue("@NoteID", Convert.ToInt32(n_dataTable.Rows[n_dataGridView.CurrentCell.RowIndex]["NoteID"]));

        connection.Open();
        n_command.ExecuteNonQuery();
        connection.Close();

        n_dataTable.Clear();
        GetNotes();
    }
    else //if the item is new
    {
        string query = "INSERT INTO Notes (n_Title,n_Notes) VALUES (@n_Title,@n_Notes)";

        n_command = new OleDbCommand(query, connection);
        n_command.Parameters.AddWithValue("@n_Title", n_textBoxTitle.Text);
        n_command.Parameters.AddWithValue("@n_Notes", n_textBoxNotes.Text);

        connection.Open();
        n_command.ExecuteNonQuery();
        connection.Close();

        n_dataTable.Clear();
        GetNotes();
    }

    n_textBoxTitle.Text = ""; n_textBoxNotes.Text = "";
    editingNote = false;
}
```

```
189         // for passwords
            1 reference
190         private void p_buttonDelete_Click(object sender, EventArgs e)
191         {
192             try
193             {
194                 string query = "Delete FROM Passwords WHERE PasswordID=@PasswordID";
195
196                 p_command = new OleDbCommand(query, connection);
197                 p_command.Parameters.AddWithValue("@PasswordID", Convert.ToInt32(p_dataTable.Rows[p_dataGridView.
198                     CurrentCell.RowIndex]["PasswordID"]));
199
200                 connection.Open();
201                 p_command.ExecuteNonQuery();
202                 connection.Close();
203
204                 p_dataTable.Clear();
205                 GetPasswords();
206             }
207             catch (Exception) { MessageBox.Show("Select row to delete."); }
208         }
209
            1 reference
210         private void p_dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
211         {
212             try
213             {
214                 p_textBoxTitle.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
215                 p_textBoxUsername.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[2].ToString();
216                 p_textBoxPassword.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[3].ToString();
217                 p_textBoxLink.Text = p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex].ItemArray[4].ToString();
218                 editingPassword = true;
219             }
220             catch (Exception) { MessageBox.Show("Selected no row / empty row."); }
221         }
222
```

```
            1 reference
223         private void p_buttonNew_Click(object sender, EventArgs e)
224         {
225             p_textBoxTitle.Text = ""; p_textBoxUsername.Text = ""; p_textBoxPassword.Text = ""; p_textBoxLink.Text = "";
226             editingPassword = false;
227         }
228
```

```
1 reference
private void p_buttonSave_Click(object sender, EventArgs e)
{
    if (editingPassword == true) //if existing item is being overwritten
    {
        string query = "UPDATE Passwords SET p_Title=@p_Title, p_Username=@p_Username, p_Password=@p_Password, p_Link=@p_Link " +
            "WHERE PasswordID=@PasswordID";

        p_command = new OleDbCommand(query, connection);
        p_command.Parameters.AddWithValue("@p_Title", p_textBoxTitle.Text);
        p_command.Parameters.AddWithValue("@p_Username", p_textBoxUsername.Text);
        p_command.Parameters.AddWithValue("@p_Password", p_textBoxPassword.Text);
        p_command.Parameters.AddWithValue("@p_Link", p_textBoxLink.Text);
        p_command.Parameters.AddWithValue("@PasswordID", Convert.ToInt32(p_dataTable.Rows[p_dataGridView.CurrentCell.RowIndex]["PasswordID"]));
        connection.Open();
        p_command.ExecuteNonQuery();
        connection.Close();

        p_dataTable.Clear();
        GetPasswords();
    }
    else //if the item is new
    {
        string query = "INSERT INTO Passwords (p_Title, p_Username, p_Password, p_Link) VALUES (@p_Title, @p_Username, @p_Password, @p_Link)";

        p_command = new OleDbCommand(query, connection);
        p_command.Parameters.AddWithValue("@p_Title", p_textBoxTitle.Text);
        p_command.Parameters.AddWithValue("@p_Username", p_textBoxUsername.Text);
        p_command.Parameters.AddWithValue("@p_Password", toHash256(p_textBoxPassword.Text));
        p_command.Parameters.AddWithValue("@p_Link", p_textBoxLink.Text);

        connection.Open();
        p_command.ExecuteNonQuery();
        connection.Close();

        p_dataTable.Clear();
        GetPasswords();
    }

    p_textBoxTitle.Text = ""; p_textBoxUsername.Text = ""; p_textBoxPassword.Text = ""; p_textBoxLink.Text = "";
    editingPassword = false;

}
```

```
274         // for bank account
            1 reference
275         private void ba_buttonDelete_Click(object sender, EventArgs e)
276         {
277             try
278             {
279                 string query = "Delete FROM BankAccounts WHERE BankAccountID=@BankAccountID";
280
281                 ba_command = new OleDbCommand(query, connection);
282                 ba_command.Parameters.AddWithValue("@BankAccountID", Convert.ToInt32(ba_dataTable.Rows[ba_dataGridView.
283                     CurrentCell.RowIndex]["BankAccountID"]));
284
285                 connection.Open();
286                 ba_command.ExecuteNonQuery();
287                 connection.Close();
288
289                 ba_dataTable.Clear();
290                 GetBankAccounts();
291             }
292             catch (Exception) { MessageBox.Show("Select row to delete."); }
293         }
294
            1 reference
295         private void ba_dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
296         {
297             try
298             {
299                 ba_textBoxTitle.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
300                 ba_textBoxUsername.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[2].ToString();
301                 ba_textBoxPassword.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[3].ToString();
302                 ba_textBoxAccNo.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[4].ToString();
303                 ba_textBoxSortCode.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[5].ToString();
304                 ba_textBoxLink.Text = ba_dataTable.Rows[ba_dataGridView.CurrentCell.RowIndex].ItemArray[6].ToString();
305                 editingBankAccount = true;
306             }
307             catch (Exception) { MessageBox.Show("Selected empty row."); }
308         }
309
            1 reference
310         private void ba_buttonNew_Click(object sender, EventArgs e)
311         {
312             ba_textBoxTitle.Text = ""; ba_textBoxUsername.Text = ""; ba_textBoxPassword.Text = "";
313             ba_textBoxAccNo.Text = ""; ba_textBoxSortCode.Text = ""; ba_textBoxLink.Text = "";
314             editingBankAccount = false;
315         }
```

```
1 reference
private void ba_buttonSave_Click(object sender, EventArgs e) //if existing item is being overwritten
{
    if (editingBankAccount == true)
    {
        string query = "UPDATE BankAccounts SET ba_Title=@ba_Title, ba_Username=@ba_Username, ba_Password=@ba_Password, " +
            "ba_AccountNumber=@ba_AccountNumber, ba_SortCode=@ba_SortCode, ba_Link=@ba_Link " +
            "WHERE BankAccountID=@BankAccountID";

        ba_command = new OleDbCommand(query, connection);
        ba_command.Parameters.AddWithValue("@ba_Title", ba_textBoxTitle.Text);
        ba_command.Parameters.AddWithValue("@ba_Username", ba_textBoxUsername.Text);
        ba_command.Parameters.AddWithValue("@ba_Password", ba_textBoxPassword.Text);
        ba_command.Parameters.AddWithValue("@ba_AccountNumber", ba_textBoxAccNo.Text);
        ba_command.Parameters.AddWithValue("@ba_SortCode", ba_textBoxSortCode.Text);
        ba_command.Parameters.AddWithValue("@ba_Link", ba_textBoxLink.Text);
        ba_command.Parameters.AddWithValue("@BankAccountID", Convert.ToInt32(ba_dataTable.Rows[ba_dataGridView.
            CurrentCell.RowIndex]["BankAccountID"]));

        connection.Open();
        ba_command.ExecuteNonQuery();
        connection.Close();

        ba_dataTable.Clear();
        GetBankAccounts();
    }
    else //if the item is new
    {
        string query = "INSERT INTO BankAccounts (ba_Title, ba_Username, ba_Password, ba_AccountNumber, ba_SortCode, ba_Link) " +
            "VALUES (@ba_Title, @ba_Username, @ba_Password, @ba_AccountNumber, @ba_SortCode, @ba_Link)";

        ba_command = new OleDbCommand(query, connection);
        ba_command.Parameters.AddWithValue("@ba_Title", ba_textBoxTitle.Text);
        ba_command.Parameters.AddWithValue("@ba_Username", ba_textBoxUsername.Text);
        ba_command.Parameters.AddWithValue("@ba_Password", ba_textBoxPassword.Text);
        ba_command.Parameters.AddWithValue("@ba_AccountNumber", ba_textBoxAccNo.Text);
        ba_command.Parameters.AddWithValue("@ba_SortCode", ba_textBoxSortCode.Text);
        ba_command.Parameters.AddWithValue("@ba_Link", ba_textBoxLink.Text);
        connection.Open();
        ba_command.ExecuteNonQuery();
        connection.Close();

        ba_dataTable.Clear();
        GetBankAccounts();
    }

    ba_textBoxTitle.Text = ""; ba_textBoxUsername.Text = ""; ba_textBoxPassword.Text = "";
    ba_textBoxAccNo.Text = ""; ba_textBoxSortCode.Text = ""; ba_textBoxLink.Text = "";
    editingBankAccount = false;
}
```

```
369         // for cards
            1 reference
370         private void c_buttonDelete_Click(object sender, EventArgs e)
371         {
372             try
373             {
374                 string query = "Delete FROM Cards WHERE CardID=@CardID";
375
376                 c_command = new OleDbCommand(query, connection);
377                 c_command.Parameters.AddWithValue("@CardID", Convert.ToInt32(c_dataTable.Rows[c_dataGridView.
378                     CurrentCell.RowIndex]["CardID"]));
379
380                 connection.Open();
381                 c_command.ExecuteNonQuery();
382                 connection.Close();
383
384                 c_dataTable.Clear();
385                 GetCards();
386             }
387             catch (Exception) { MessageBox.Show("Select row to delete."); }
388         }
389
            1 reference
390         private void c_dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
391         {
392             try
393             {
394                 c_textBoxTitle.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
395                 c_textBoxNameOnCard.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[2].ToString();
396                 c_textBoxCardNo.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[3].ToString();
397                 c_textBoxCardType.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[4].ToString();
398                 c_textBoxSecCode.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[5].ToString();
399                 c_textBoxStartDate.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[6].ToString();
400                 c_textBoxEndDate.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[7].ToString();
401                 c_textBoxLink.Text = c_dataTable.Rows[c_dataGridView.CurrentCell.RowIndex].ItemArray[8].ToString();
402                 editingCard = true;
403             }
404             catch (Exception) { MessageBox.Show("Selected empty row."); }
405         }
406
            1 reference
407         private void c_buttonNew_Click(object sender, EventArgs e)
408         {
409             c_textBoxTitle.Text = ""; c_textBoxNameOnCard.Text = ""; c_textBoxCardNo.Text = ""; c_textBoxCardType.Text = "";
410             c_textBoxSecCode.Text = ""; c_textBoxStartDate.Text = ""; c_textBoxEndDate.Text = ""; c_textBoxLink.Text = "";
411             editingCard = false;
412         }
```

```
414         private void c_buttonSave_Click(object sender, EventArgs e)
415         {
416             if (editingCard == true) //if existing item is being overwritten
417             {
418                 string query = "UPDATE Cards SET c_Title=@c_Title, c_NameOnCard=@c_NameOnCard, c_CardNumber=@c_CardNumber, " +
419                     "c_CardType=@c_CardType, c_SecurityCode=@c_SecurityCode, c_StartDate=@c_StartDate, c_EndDate=@c_EndDate, c_Link=@c_Link " +
420                     "WHERE CardID=@CardID";
421 
422                 c_command = new OleDbCommand(query, connection);
423                 c_command.Parameters.AddWithValue("@c_Title", c_textBoxTitle.Text);
424                 c_command.Parameters.AddWithValue("@c_NameOnCard", c_textBoxNameOnCard.Text);
425                 c_command.Parameters.AddWithValue("@c_CardNumber", c_textBoxCardNo.Text);
426                 c_command.Parameters.AddWithValue("@c_CardType", c_textBoxCardType.Text);
427                 c_command.Parameters.AddWithValue("@c_SecurityCode", c_textBoxSecCode.Text);
428                 c_command.Parameters.AddWithValue("@c_StartDate", c_textBoxStartDate.Text);
429                 c_command.Parameters.AddWithValue("@c_EndDate", c_textBoxEndDate.Text);
430                 c_command.Parameters.AddWithValue("@c_Link", c_textBoxLink.Text);
431                 c_command.Parameters.AddWithValue("@CardID", Convert.ToInt32(c_dataTable.Rows[c_dataGridView.CurrentCell.
432                     RowIndex]["CardID"]));
433                 connection.Open();
434                 c_command.ExecuteNonQuery();
435                 connection.Close();
436 
437                 c_dataTable.Clear();
438                 GetCards();
439             }
440             else //if the item is new
441             {
442                 string query = "INSERT INTO Cards (c_Title, c_NameOnCard, c_CardNumber, c_CardType, c_SecurityCode, c_StartDate, " +
443                     "c_EndDate, c_Link) " +
444                     "VALUES (@c_Title, @c_NameOnCard, @c_CardNumber, @c_CardType, @c_SecurityCode, @c_StartDate, @c_EndDate, @c_Link)";
445 
446                 c_command = new OleDbCommand(query, connection);
447                 c_command.Parameters.AddWithValue("@c_Title", c_textBoxTitle.Text);
448                 c_command.Parameters.AddWithValue("@c_NameOnCard", c_textBoxNameOnCard.Text);
449                 c_command.Parameters.AddWithValue("@c_CardNumber", c_textBoxCardNo.Text);
450                 c_command.Parameters.AddWithValue("@c_CardType", c_textBoxCardType.Text);
451                 c_command.Parameters.AddWithValue("@c_SecurityCode", c_textBoxSecCode.Text);
452                 c_command.Parameters.AddWithValue("@c_StartDate", c_textBoxStartDate.Text);
453                 c_command.Parameters.AddWithValue("@c_EndDate", c_textBoxEndDate.Text);
454                 c_command.Parameters.AddWithValue("@c_Link", c_textBoxLink.Text);
455                 connection.Open();
456                 c_command.ExecuteNonQuery();
457                 connection.Close();
458 
459                 c_dataTable.Clear();
460                 GetCards();
461             }
462 
463             c_textBoxTitle.Text = ""; c_textBoxNameOnCard.Text = ""; c_textBoxCardNo.Text = ""; c_textBoxCardType.Text = "";
464             c_textBoxSecCode.Text = ""; c_textBoxStartDate.Text = ""; c_textBoxEndDate.Text = ""; c_textBoxLink.Text = "";
465             editingCard = false;
466         }
```

```
470         // for addresses
            1 reference
471         private void a_buttonDelete_Click(object sender, EventArgs e)
472         {
473             try
474             {
475                 string query = "Delete FROM Addresses WHERE AddressID=@AddressID";
476
477                 a_command = new OleDbCommand(query, connection);
478                 a_command.Parameters.AddWithValue("@AddressID", Convert.ToInt32(a_dataTable.Rows[a_dataGridView.
479                     CurrentCell.RowIndex]["AddressID"]));
480
481                 connection.Open();
482                 a_command.ExecuteNonQuery();
483                 connection.Close();
484
485                 a_dataTable.Clear();
486                 GetAddresses();
487             }
488             catch (Exception) { MessageBox.Show("Select row to delete."); }
489         }
490
            1 reference
491         private void a_dataGridView_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
492         {
493             try
494             {
495                 a_textBoxTitle.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[1].ToString();
496                 a_textBoxAddress1.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[2].ToString();
497                 a_textBoxAddress2.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[3].ToString();
498                 a_textBoxAddress3.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[4].ToString();
499                 a_textBoxCity.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[5].ToString();
500                 a_textBoxCounty.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[6].ToString();
501                 a_textBoxPostcode.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[7].ToString();
502                 a_textBoxLink.Text = a_dataTable.Rows[a_dataGridView.CurrentCell.RowIndex].ItemArray[8].ToString();
503                 editingAddress = true;
504             }
505             catch (Exception) { MessageBox.Show("Selected empty row."); }
506         }
507
            1 reference
508         private void a_buttonNew_Click(object sender, EventArgs e)
509         {
510             a_textBoxTitle.Text = ""; a_textBoxAddress1.Text = ""; a_textBoxAddress2.Text = ""; a_textBoxAddress3.Text = "";
511             a_textBoxCity.Text = ""; a_textBoxCounty.Text = ""; a_textBoxPostcode.Text = ""; a_textBoxLink.Text = "";
512             editingAddress = false;
513         }
```

```
1 reference
private void a_buttonSave_Click(object sender, EventArgs e) //if existing item is being overwritten
{
    if (editingAddress == true)
    {
        string query = "UPDATE Addresses SET a_Title=@a_Title, a_Address1=@a_Address1, " +
            "a_Address2=@a_Address2, a_Address3=@a_Address3, a_CityTown=@a_CityTown, a_CountyState=@a_CountyState, " +
            "a_Postcode=@a_Postcode, a_Link=@a_Link " +
            "WHERE AddressID=@AddressID";

        a_command = new OleDbCommand(query, connection);
        a_command.Parameters.AddWithValue("@a_Title", a_textBoxTitle.Text);
        a_command.Parameters.AddWithValue("@a_Address1", a_textBoxAddress1.Text);
        a_command.Parameters.AddWithValue("@a_Address2", a_textBoxAddress2.Text);
        a_command.Parameters.AddWithValue("@a_Address3", a_textBoxAddress3.Text);
        a_command.Parameters.AddWithValue("@a_CityTown", a_textBoxCity.Text);
        a_command.Parameters.AddWithValue("@a_CountyState", a_textBoxCounty.Text);
        a_command.Parameters.AddWithValue("@a_Postcode", a_textBoxPostcode.Text);
        a_command.Parameters.AddWithValue("@a_Link", a_textBoxLink.Text);
        a_command.Parameters.AddWithValue("@a_AddressID", Convert.ToInt32(a_dataTable.Rows[a_dataGridView.
            CurrentCell.RowIndex]["AddressID"]));
        connection.Open();
        a_command.ExecuteNonQuery();
        connection.Close();

        a_dataTable.Clear();
        GetAddresses();
    }
    else //if the item is new
    {
        string query = "INSERT INTO Addresses (a_Title, a_Address1, a_Address2, a_Address3, a_CityTown, a_CountyState, a_Postcode, a_Link) " +
            "VALUES (@a_Title, @a_Address1, @a_Address2, @a_Address3, @a_CityTown, @a_CountyState, @a_Postcode, @a_Link)";

        a_command = new OleDbCommand(query, connection);
        a_command.Parameters.AddWithValue("@a_Title", a_textBoxTitle.Text);
        a_command.Parameters.AddWithValue("@a_Address1", a_textBoxAddress1.Text);
        a_command.Parameters.AddWithValue("@a_Address2", a_textBoxAddress2.Text);
        a_command.Parameters.AddWithValue("@a_Address3", a_textBoxAddress3.Text);
        a_command.Parameters.AddWithValue("@a_CityTown", a_textBoxCity.Text);
        a_command.Parameters.AddWithValue("@a_CountyState", a_textBoxCounty.Text);
        a_command.Parameters.AddWithValue("@a_Postcode", a_textBoxPostcode.Text);
        a_command.Parameters.AddWithValue("@a_Link", a_textBoxLink.Text);
        connection.Open();
        a_command.ExecuteNonQuery();
        connection.Close();

        a_dataTable.Clear();
        GetAddresses();
    }

    a_textBoxTitle.Text = ""; a_textBoxAddress1.Text = ""; a_textBoxAddress2.Text = ""; a_textBoxAddress3.Text = "";
    a_textBoxCity.Text = ""; a_textBoxCounty.Text = ""; a_textBoxPostcode.Text = ""; a_textBoxLink.Text = "";
    editingAddress = false;
}
```

```
571         // Password Generator
572         Random RandomCharacter = new Random();
573         int currentPasswordLength = 0;
574
            1 reference
575         private void trackBarPasswordLength_Scroll(object sender, EventArgs e)
576         {
577             currentPasswordLength = trackBarPasswordLength.Value;
578             randomPasswordGenerator(currentPasswordLength);
579         }
580
            1 reference
581         private void randomPasswordGenerator(int passwordLength)
582         {
583
584             char[] Characters =
585             {
586                     'Q','W','E','R','T','Y','U','I','O','P','A','S','D','F','G','H','J','K','L','Z',
587                     'X','C','V','B','N','M','q','w','e','r','t','y','u','i','o','p','a','s','d','f',
588                     'g','h','j','k','l','z','x','c','v','b','n','m','1','2','3','4','5','6','7','8',
589                     '9','0','!','£','$','%','^','&','*','(',')','_','-','+','=','{','}','[',']',';',
590                     ':','@','#','~',',','<','>','.','/','?'
591             };
592
593             string randomPassword = "";
594
595             for (int i = 0; i < passwordLength; i++)
596             {
597                 randomPassword += Characters[RandomCharacter.Next(0, 90)];
598             }
599
600             labelPassword.Text = randomPassword;
601         }
602
            1 reference
603         private void buttonCopyPassword_Click(object sender, EventArgs e)
604         {
605             Clipboard.SetText(labelPassword.Text);
606         }
607
```

```
610         // Hashing Algorithm
            1 reference
611         public string toHash256(string passwordInput)
612         {
613             var sha256 = SHA256.Create();
614             byte[] bytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(passwordInput));
615
616             var hashedPassword = new StringBuilder();
617             for (int i = 0; i < bytes.Length; i++)
618             {
619                 hashedPassword.Append(bytes[i].ToString("x2"));
620             }
621             return hashedPassword.ToString();
622         }
623     }
624 }
```

COMPLETE PSEUDOCODE

--- Log In / Sign Up ---

SET currentTab TO tabLogIn

PROCEDURE tabSignUp_Click()

SET currentTab TO tabSignUp

END PROCEDURE

```
PROCEDURE su_buttonSignUp_Click()

        REQUEST DATA IN MasterAccount TABLE IN PassVault DATABASE

        IF su_textBoxMasterUsername.Text IN MasterAccount DO

                IF su_textBoxMasterPassword.Text.Length < 8 OR su_textBoxMasterPassword.Text.Length > 16

                        ADD su_textBoxMasterPassword.Text AND su_textBoxMasterUsername.Text TO MasterAccount TABLE

                ELSE DO

                        RETURN "Password must be 8 - 16 characters." TO DISPLAY

                END IF

        ELSE DO

                RETURN "Username exists already." TO DISPLAY
```

```
        END IF

END PROCEDURE


PROCEDURE tabLogIn_Click()

        SET currentTab TO tabLogIn

END PROCEDURE


PROCEDURE l_buttonLogIn_Click()

        REQUEST DATA IN MasterAccount TABLE IN PassVault DATABASE

        IF l_textBoxMasterUsername.Text IN MasterAccount TABLE DO

                IF l_textBoxMasterPassword.Text FIELD IN l_textBoxMasterUsername.Text FIELD
```

```
                    LOAD l_textBoxMasterUsername.Text PRIMARY KEY TO Vault

                ELSE DO

                    RETURN "Invalid password." TO DISPLAY

                ENDIF

            ELSE DO

                RETURN "Invalid username." TO DISPLAY

            END IF

END PROCEDURE
```

--- Vault ---

```
PROCEDURE tabPassword_Click()

    SET currentTab TO tabPassword

END PROCEDURE

PROCEDURE tabBankAccount_Click()

    SET currentTab TO tabBankAccount

END PROCEDURE

PROCEDURE tabCards_Click()

    SET currentTab TO tabCards

END PROCEDURE
```

```
PROCEDURE tabAddresses_Click()

        SET currentTab TO tabAddresses

END PROCEDURE
```

```
PROCEDURE tabNotes_Click()

        SET currentTab TO tabNotes

END PROCEDURE
```

```
PROCEDURE tabPasswordGenerator_Click()

        SET currentTab TO tabPasswordGenerator
```

```
END PROCEDURE


--- Passwords ---


SET p_dataTable TO NEW DATATABLE

SET editingPassword TO BOOL FALSE


PROCEDURE p_buttonDelete_Click()

        TRY

                DELETE p_dataTable.ROWS[p_dataGridView.CURRENTROW]

        CATCH Exception
```

```
            SEND "Select row to delete." TO DISPLAY

        END TRYCATCH

END PROCEDURE

PROCEDURE p_buttonLoad_Click OR p_dataGridView_DoubleClick()

        TRY

            SET p_textBoxTitle.Text TO p_dataTable.ROWS[p_dataGridView.CURRENTROW].ITEMARRAY[0]

            SET p_textBoxUsername.Text TO p_dataTable.ROWS[p_dataGridView.CURRENTROW].ITEMARRAY[1]

            SET p_textBoxPassword.Text TO p_dataTable.ROWS[p_dataGridView.CURRENTROW].ITEMARRAY[2]

            SET p_textBoxLink.Text TO p_dataTable.ROWS[p_dataGridView.CURRENTROW].ITEMARRAY[3]

            SET editingPassword TO BOOL TRUE

        CATCH Exception

            SEND "Selected empty row." TO DISPLAY
```

```
        END TRYCATCH

END PROCEDURE

PROCEDURE p_buttonNew_Click()

        SET p_textBoxTitle.Text TO ""

        SET p_textBoxUsername.Text TO ""

        SET p_textBoxPassword.Text TO ""

        SET p_textBoxLink.Text TO ""

END PROCEDURE

PROCEDURE p_buttonSave_Click()

        REQUEST DATA IN Passwords TABLE IN PassVault DATABASE
```

```
IF editingPassword == TRUE DO

        SET p_dataTable.ROWS[p_dataGridView.CURRENTROW]["Title"] TO p_textBoxTitle.Text

        ADD toHash256(p_textBoxTitle.Text ) TO Passwords TABLE

        SET p_dataTable.ROWS[p_dataGridView.CURRENTROW]["Username"] TO p_textBoxUsername.Text

        ADD toHash256( p_textBoxUsername.Text) TO Passwords TABLE

        SET p_dataTable.ROWS[p_dataGridView.CURRENTROW]["Password"] TO p_textBoxPassword.Text

        ADD toHash256(p_textBoxPassword.Text ) TO Passwords TABLE

        SET p_dataTable.ROWS[p_dataGridView.CURRENTROW]["Link"] TO p_textBoxLink.Text

        ADD toHash256(p_textBoxLink.Text ) TO Passwords TABLE

ELSE DO

        SET p_dataTable.ROWS("Title") TO p_textBoxTitle.Text

        OVERWRITE toHash256( p_textBoxTitle.Text) IN Passwords TABLE

        SET p_dataTable.ROWS("Username") TO p_textBoxUsername.Text

        OVERWRITE toHash256( p_textBoxUsername.Text) IN Passwords TABLE
```

```
                SET p_dataTable.ROWS("Password") TO p_textBoxPassword.Text

                OVERWRITE toHash256( p_textBoxPassword.Text) IN Passwords TABLE


                SET p_dataTable.ROWS("Link") TO p_textBoxLink.Text

                OVERWRITE toHash256( p_textBoxLink.Text) IN Passwords TABLE


        END IF


        SET p_textBoxTitle.Text TO ""

        SET p_textBoxUsername.Text TO ""

        SET p_textBoxPassword.Text TO ""

        SET p_textBoxLink.Text TO ""

        SET editingPassword TO False


END PROCEDURE



--- Bank Accounts ---
```

```
SET ba_dataTable TO NEW DATATABLE

SET editingBankAccount TO BOOL FALSE



PROCEDURE ba_buttonDelete_Click()

        TRY

                DELETE ba_dataTable.ROWS[ba_dataGridView.CURRENTROW]

        CATCH Exception

                SEND "Select row to delete." TO DISPLAY

        END TRYCATCH

END PROCEDURE
```

```
PROCEDURE ba_buttonLoad_Click OR ba_dataGridView_DoubleClick()

        TRY

                SET ba_textBoxTitle.Text TO ba_dataTable.ROWS[ba_dataGridView.CURRENTROW].ITEMARRAY[0]

                SET ba_textBoxUsername.Text TO ba_dataTable.ROWS[ba_dataGridView.CURRENTROW].ITEMARRAY[1]

                SET ba_textBoxPassword.Text TO ba_dataTable.ROWS[ba_dataGridView.CURRENTROW].ITEMARRAY[2]

                SET ba_textBoxAccNo.Text TO ba_dataTable.ROWS[ba_dataGridView.CURRENTROW].ITEMARRAY[3]

                SET ba_textBoxSortCode.Text TO ba_dataTable.ROWS[ba_dataGridView.CURRENTROW].ITEMARRAY[4]

                SET ba_textBoxLink.Text TO ba_dataTable.ROWS[ba_dataGridView.CURRENTROW].ITEMARRAY[5]

                SET editingBankAccount TO BOOL TRUE

        CATCH Exception

                SEND "Selected empty row." TO DISPLAY

        END TRYCATCH

END PROCEDURE
```

```
PROCEDURE ba_buttonNew_Click()

        SET ba_textBoxTitle.Text TO ""

        SET ba_textBoxUsername.Text TO ""

        SET ba_textBoxPassword.Text TO ""

        SET ba_textBoxAccNo.Text TO ""

        SET ba_textBoxSortCode.Text TO ""

        SET ba_textBoxLink.Text TO ""

END PROCEDURE

PROCEDURE ba_buttonSave_Click()

        REQUEST DATA IN BankAccounts TABLE IN PassVault DATABASE

        IF editingBankAccount == TRUE DO

                SET ba_dataTable.ROWS[ba_dataGridView.CURRENTROW]["Title"] TO ba_textBoxTitle.Text

                ADD toHash256(ba_textBoxTitle.Text ) TO BankAccounts TABLE
```

SET ba_dataTable.ROWS[ba_dataGridView.CURRENTROW]["Username"] TO ba_textBoxUsername.Text

ADD toHash256(ba_textBoxUsername.Text) TO BankAccounts TABLE

SET ba_dataTable.ROWS[ba_dataGridView.CURRENTROW]["Password"] TO ba_textBoxPassword.Text

ADD toHash256(ba_textBoxPassword.Text) TO BankAccounts TABLE

SET ba_dataTable.ROWS[ba_dataGridView.CURRENTROW]["Account No."] TO ba_textBoxAccNo.Text

ADD toHash256(ba_textBoxAccNo.Text) TO BankAccounts TABLE

SET ba_dataTable.ROWS[ba_dataGridView.CURRENTROW]["Sort Code"] TO ba_textBoxSortCode.Text

ADD toHash256(ba_textBoxSortCode.Text) TO BankAccounts TABLE

SET ba_dataTable.ROWS[ba_dataGridView.CURRENTROW]["Link"] TO ba_textBoxLink.Text

ADD toHash256(ba_textBoxLink.Text) TO BankAccounts TABLE

ELSE DO

SET ba_dataTable.ROWS("Title") TO ba_textBoxTitle.Text

OVERWRITE toHash256(ba_textBoxTitle.Text) IN BankAccounts TABLE

SET ba_dataTable.ROWS("Username") TO ba_textBoxUsername.Text

```
            OVERWRITE toHash256( ba_textBoxUsername.Text) IN BankAccounts TABLE

            SET ba_dataTable.ROWS("Password") TO ba_textBoxPassword.Text

            OVERWRITE toHash256( ba_textBoxPassword.Text) IN BankAccounts TABLE

            SET ba_dataTable.ROWS("Account No.") TO ba_textBoxAccNo.Text

            OVERWRITE toHash256(ba_textBoxAccNo.Text ) IN BankAccounts TABLE

            SET ba_dataTable.ROWS("Sort Code") TO ba_textBoxSortCode.Text

            OVERWRITE toHash256(ba_textBoxSortCode.Text ) IN BankAccounts TABLE

            SET ba_dataTable.ROWS("Link") TO ba_textBoxLink.Text

            OVERWRITE toHash256( ba_textBoxLink.Text) IN BankAccounts TABLE

        END IF

        SET ba_textBoxTitle.Text TO ""

        SET ba_textBoxUsername.Text TO ""

        SET ba_textBoxPassword.Text TO ""

        SET ba_textBoxAccNo.Text TO ""

        SET ba_textBoxSortCode.Text TO ""

        SET ba_textBoxLink.Text TO ""
```

```
        SET editingBankAccount TO False

END PROCEDURE

--- Cards ---

SET c_dataTable TO NEW DATATABLE

SET editingCards TO BOOL FALSE

PROCEDURE c_buttonDelete_Click()

        TRY

                DELETE c_dataTable.ROWS[c_dataGridView.CURRENTROW]
```

```
        CATCH Exception

            SEND "Select row to delete." TO DISPLAY

        END TRYCATCH

END PROCEDURE

PROCEDURE c_buttonLoad_Click OR c_dataGridView_DoubleClick()

        TRY

            SET c_textBoxTitle.Text TO c_dataTable.ROWS[c_dataGridView.CURRENTROW].ITEMARRAY[0]

            SET c_textBoxNameOnCard.Text TO c_dataTable.ROWS[c_dataGridView.CURRENTROW].ITEMARRAY[1]

            SET c_textBoxCardNo.Text TO c_dataTable.ROWS[c_dataGridView.CURRENTROW].ITEMARRAY[2]

            SET c_textBoxCardType.Text TO c_dataTable.ROWS[c_dataGridView.CURRENTROW].ITEMARRAY[3]

            SET c_textBoxSecCode.Text TO c_dataTable.ROWS[c_dataGridView.CURRENTROW].ITEMARRAY[4]

            SET c_textBoxStartDate.Text TO c_dataTable.ROWS[c_dataGridView.CURRENTROW].ITEMARRAY[5]

            SET c_textBoxEndDate.Text TO c_dataTable.ROWS[c_dataGridView.CURRENTROW].ITEMARRAY[6]
```

```
                SET c_textBoxLink.Text TO c_dataTable.ROWS[c_dataGridView.CURRENTROW].ITEMARRAY[7]

                SET editingCards TO BOOL TRUE


            CATCH Exception


                SEND "Selected empty row." TO DISPLAY


            END TRYCATCH

END PROCEDURE



PROCEDURE c_buttonNew_Click()


            SET c_textBoxTitle.Text TO ""

            SET c_textBoxNameOnCard.Text TO ""

            SET c_textBoxCardNo.Text TO ""

            SET c_textBoxCardType.Text TO ""

            SET c_textBoxSecCode.Text TO ""

            SET c_textBoxStartDate.Text TO ""

            SET c_textBoxEndDate.Text TO ""
```

```
        SET c_textBoxLink.Text TO ""

END PROCEDURE



PROCEDURE c_buttonSave_Click()

        REQUEST DATA IN Cards TABLE IN PassVault DATABASE

        IF editingCards == TRUE DO

                SET c_dataTable.ROWS[c_dataGridView.CURRENTROW]["Title"] TO c_textBoxTitle.Text

                ADD toHash256(c_textBoxTitle.Text ) TO Cards TABLE


                SET c_dataTable.ROWS[c_dataGridView.CURRENTROW]["Name on Card"] TO c_textBoxNameOnCard.Text

                ADD toHash256(c_textBoxNameOnCard.Text) TO Cards TABLE


                SET c_dataTable.ROWS[c_dataGridView.CURRENTROW]["Card No."] TO c_textBoxCardNo.Text

                ADD toHash256( c_textBoxCardNo.Text) TO Cards TABLE


                SET c_dataTable.ROWS[c_dataGridView.CURRENTROW]["Card Type"] TO c_textBoxCardType.Text
```

```
ADD toHash256(c_textBoxCardType.Text ) TO Cards TABLE

SET c_dataTable.ROWS[c_dataGridView.CURRENTROW]["Sec. Code"] TO c_textBoxSecCode.Text

ADD toHash256(c_textBoxSecCode.Text ) TO Cards TABLE

SET c_dataTable.ROWS[c_dataGridView.CURRENTROW]["Start Date"] TO c_textBoxStartDate.Text

ADD toHash256(c_textBoxStartDate.Text ) TO Cards TABLE

SET c_dataTable.ROWS[c_dataGridView.CURRENTROW]["End Dade"] TO c_textBoxEndDate.Text

ADD toHash256(c_textBoxEndDate.Text ) TO Cards TABLE

SET c_dataTable.ROWS[c_dataGridView.CURRENTROW]["Link"] TO c_textBoxLink.Text

ADD toHash256(c_textBoxLink.Text.Text ) TO Cards TABLE


ELSE DO

SET c_dataTable.ROWS("Title") TO c_textBoxTitle.Text

OVERWRITE toHash256( c_textBoxTitle.Text) IN Cards TABLE

SET c_dataTable.ROWS("Name on Card") TO c_textBoxNameOnCard.Text

OVERWRITE toHash256(c_textBoxNameOnCard.Text ) IN Cards TABLE
```

```
            SET c_dataTable.ROWS("Card No.") TO c_textBoxCardNo.Text

            OVERWRITE toHash256( c_textBoxCardNo.Text) IN Cards TABLE


            SET c_dataTable.ROWS("Card Type") TO c_textBoxCardType.Text

            OVERWRITE toHash256(c_textBoxCardType.Text ) IN Cards TABLE


            SET c_dataTable.ROWS("Sec. Code") TO c_textBoxSecCode.Text

            OVERWRITE toHash256(c_textBoxSecCode.Text ) IN Cards TABLE


            SET c_dataTable.ROWS("Start Date") TO c_textBoxStartDate.Text

            OVERWRITE toHash256(c_textBoxStartDate.Text ) IN Cards TABLE


            SET c_dataTable.ROWS("End Date") TO c_textBoxEndDate.Text

            OVERWRITE toHash256(c_textBoxEndDate.Text ) IN Cards TABLE


            SET c_dataTable.ROWS("Link") TO c_textBoxLink.Text

            OVERWRITE toHash256(c_textBoxLink.Text ) IN Cards TABLE


        END IF


        SET a_textBoxTitle.Text TO ""
```

```
        SET c_textBoxNameOnCard.Text TO ""
        SET c_textBoxCardNo.Text TO ""
        SET c_textBoxCardType.Text TO ""
        SET c_textBoxSecCode.Text TO ""
        SET c_textBoxStartDate.Text TO ""
        SET c_textBoxEndDate.Text TO ""
        SET c_textBoxLink.Text TO ""
        SET editingCards TO False

END PROCEDURE

--- Addresses ---

SET a_dataTable TO NEW DATATABLE

SET editingAddress TO BOOL FALSE
```

```
PROCEDURE a_buttonDelete_Click()

        TRY

                DELETE a_dataTable.ROWS[a_dataGridView.CURRENTROW]

        CATCH Exception

                SEND "Select row to delete." TO DISPLAY

        END TRYCATCH

END PROCEDURE


PROCEDURE a_buttonLoad_Click OR a_dataGridView_DoubleClick()

        TRY

                SET a_textBoxTitle.Text TO a_dataTable.ROWS[a_dataGridView.CURRENTROW].ITEMARRAY[0]
```

```
                SET a_textBoxAddress1.Text TO a_dataTable.ROWS[a_dataGridView.CURRENTROW].ITEMARRAY[1]

                SET a_textBoxAddress2.Text TO a_dataTable.ROWS[a_dataGridView.CURRENTROW].ITEMARRAY[2]

                SET a_textBoxAddress3.Text TO a_dataTable.ROWS[a_dataGridView.CURRENTROW].ITEMARRAY[3]

                SET a_textBoxCity.Text TO a_dataTable.ROWS[a_dataGridView.CURRENTROW].ITEMARRAY[4]

                SET a_textBoxCounty.Text TO a_dataTable.ROWS[a_dataGridView.CURRENTROW].ITEMARRAY[5]

                SET a_textBoxPostCode.Text TO a_dataTable.ROWS[a_dataGridView.CURRENTROW].ITEMARRAY[6]

                SET a_textBoxLink.Text TO a_dataTable.ROWS[a_dataGridView.CURRENTROW].ITEMARRAY[7]

                SET editingAddress TO BOOL TRUE


            CATCH Exception

                SEND "Selected empty row." TO DISPLAY


            END TRYCATCH

END PROCEDURE



PROCEDURE a_buttonNew_Click()

            SET a_textBoxTitle.Text TO ""
```

```
        SET a_textBoxAddress1.Text TO ""

        SET a_textBoxAddress2.Text TO ""

        SET a_textBoxAddress3.Text TO ""

        SET a_textBoxCity.Text TO ""

        SET a_textBoxCounty.Text TO ""

        SET a_textBoxPostCode.Text TO ""

        SET a_textBoxLink.Text TO ""


END PROCEDURE




PROCEDURE a_buttonSave_Click()

        REQUEST DATA IN Addresses TABLE IN PassVault DATABASE


        IF editingAddress == TRUE DO

                SET a_dataTable.ROWS[a_dataGridView.CURRENTROW]["Title"] TO a_textBoxTitle.Text

                ADD toHash256( a_textBoxTitle.Text) TO Addresses TABLE


                SET a_dataTable.ROWS[a_dataGridView.CURRENTROW]["Address1"] TO a_textBoxAddress1.Text
```

```
ADD toHash256(a_textBoxAddress1.Text ) TO Addresses TABLE

SET a_dataTable.ROWS[a_dataGridView.CURRENTROW]["Address2"] TO a_textBoxAddress2.Text

ADD toHash256( a_textBoxAddress2.Text) TO Addresses TABLE

SET a_dataTable.ROWS[a_dataGridView.CURRENTROW]["Address3"] TO a_textBoxAddress3.Text

ADD toHash256(  a_textBoxAddress3.Text) TO Addresses TABLE

SET a_dataTable.ROWS[a_dataGridView.CURRENTROW]["City/Town"] TO a_textBoxCity.Text

ADD toHash256(a_textBoxCity.Text ) TO Addresses TABLE

SET a_dataTable.ROWS[a_dataGridView.CURRENTROW]["County/State"] TO a_textBoxCounty.Text

ADD toHash256( a_textBoxCounty.Text) TO Addresses TABLE

SET a_dataTable.ROWS[a_dataGridView.CURRENTROW]["Postal Code"] TO a_textBoxPostCode.Text

ADD toHash256( a_textBoxPostCode.Text) TO Addresses TABLE

SET a_dataTable.ROWS[a_dataGridView.CURRENTROW]["Link"] TO a_textBoxLink.Text

ADD toHash256( a_textBoxLink.Text) TO Addresses TABLE

ELSE DO
```

```
SET a_dataTable.ROWS("Title") TO a_textBoxTitle.Text

OVERWRITE toHash256( a_textBoxTitle.Text) IN Addresses TABLE


SET a_dataTable.ROWS("Address1") TO a_textBoxAddress1.Text

OVERWRITE toHash256( a_textBoxAddress1.Text) IN Addresses TABLE


SET a_dataTable.ROWS("Address2") TO a_textBoxAddress2.Text

OVERWRITE toHash256( a_textBoxAddress2.Text) IN Addresses TABLE


SET a_dataTable.ROWS("Address3") TO a_textBoxAddress3.Text

OVERWRITE toHash256( a_textBoxAddress3.Text) IN Addresses TABLE


SET a_dataTable.ROWS("City/Town") TO a_textBoxCity.Text

OVERWRITE toHash256( a_textBoxCity.Text ) IN Addresses TABLE


SET a_dataTable.ROWS("County/State") TO a_textBoxCounty.Text

OVERWRITE toHash256( a_textBoxCounty.Text) IN Addresses TABLE


SET a_dataTable.ROWS("Postal Code") TO a_textBoxPostCode.Text

OVERWRITE toHash256( a_textBoxPostCode.Text) IN Addresses TABLE


SET a_dataTable.ROWS("Link") TO a_textBoxLink.Text
```

```
                OVERWRITE toHash256( a_textBoxLink.Text) IN Addresses TABLE


            END IF


            SET a_textBoxTitle.Text TO ""

            SET a_textBoxAddress1.Text TO ""

            SET a_textBoxAddress2.Text TO ""

            SET a_textBoxAddress3.Text TO ""

            SET a_textBoxCity.Text TO ""

            SET a_textBoxCounty.Text TO ""

            SET a_textBoxPostCode.Text TO ""

            SET a_textBoxLink.Text TO ""

            SET editingAddress TO False


END PROCEDURE
```

--- Notes ---

```
SET n_dataTable TO NEW DATATABLE

SET editingNote TO BOOL FALSE



PROCEDURE n_buttonDelete_Click()

        TRY

                DELETE n_dataTable.ROWS[n_dataGridView.CURRENTROW]

        CATCH Exception

                SEND "Select row to delete." TO DISPLAY

        END TRYCATCH

END PROCEDURE
```

```
PROCEDURE n_buttonLoad_Click OR n_dataGridView_DoubleClick()

        TRY

                SET n_textBoxTitle.Text TO n_dataTable.ROWS[n_dataGridView.CURRENTROW].ITEMARRAY[0]

                SET n_textBoxNotes.Text TO n_dataTable.ROWS[n_dataGridView.CURRENTROW].ITEMARRAY[1]

                SET editingNote TO BOOL TRUE

        CATCH Exception

                SEND "Selected empty row." TO DISPLAY

        END TRYCATCH

END PROCEDURE

PROCEDURE n_buttonNew_Click()

        SET n_textBoxTitle.Text TO ""

        SET n_textBoxNotes.Text TO ""
```

```
END PROCEDURE



PROCEDURE n_buttonSave_Click()

        REQUEST DATA IN Notes TABLE IN PassVault DATABASE

        IF editingNote == TRUE DO

                SET n_dataTable.ROWS[n_dataGridView.CURRENTROW]["Title"] TO n_textBoxTitle.Text

                ADD toHash256(n_textBoxTitle.Text) TO Notes TABLE

                SET n_dataTable.ROWS[n_dataGridView.CURRENTROW]["Notes"] TO n_textBoxNotes.Text

                ADD toHash256(n_textBoxNotes.Text) TO Notes TABLE

        ELSE DO

                SET n_dataTable.ROWS("Title") TO n_textBoxTitle.Text

                OVERWRITE toHash256(n_textBoxTitle.Text) IN Notes TABLE
```

```
                SET n_dataTable.ROWS("Notes") TO n_textBoxNotes.Text

                OVERWRITE toHash256(n_textBoxNotes.Text) IN Notes TABLE


        END IF


        SET n_textBoxTitle.Text TO ""

        SET n_textBoxNotes.Text TO ""

        SET editingNote TO False


END PROCEDURE




--- Password Generator ---




SET RandomCharacter TO NEW RANDOM


SET currentPasswordLength TO INT 0


PROCEDURE trackBarPasswordLength_Scroll()
```

```
        SET currentPasswordLength TO trackBarPasswordLength

        LOAD PROCEDURE randomPasswordGenerator(currentPasswordLength)

END PROCEDURE

PROCEDURE randomPasswordGenerator(int passwordLength)

        SET Characters TO CHAR[]

                Q','W','E','R','T','Y','U','I','O','P','A','S','D','F','G','H','J','K','L','Z',
'X','C','V','B','N','M','q','w','e','r','t','y','u','i','o','p','a','s','d','f',
                'g','h','j','k','l','z','x','c','v','b','n','m','1','2','3','4','5','6','7','8',
'9','0','!','£','$','%','^','&','*','(',')','_','-','+','=','{','}','[',']',';',
':','@','#','~',',','<','>','.','/','?'

        SET randomPassword TO STRING ""

        SET i TO INT 1
```

```
        for i < passwordLength DO

                SET randomPassword TO CHAR Characters[RandomCharacter.Next(0,90)]

                SET i TO INT i + 1

        SET labelPassword TO STRING randomPassword;

END PROCEDURE

PROCEDURE buttonCopyPassword_Click

        SET CLIPBOARD TO STRING labelPassword

END PROCEDURE
```

--- Hasher ---

```
FUNCTION toHash256(string passwordInput)

        SET sha256 TO SHA256.Create();

        SET byte[] bytes TO sha256.GetBytes(passwordInput)

        SET hashedPassword TO NEW StringBuilder();

        SET i TO 0

        FOR i < bytes.Length DO

                hashedPassword.Append(bytes[i])

        RETURN hashedPassword.ToString()

        END FOR

END FUNCTION
```