Contents

[Acknowledgement 3](#_Toc534129386)

[Introduction 4](#_Toc534129387)

[Features 5](#_Toc534129388)

[GUI Description 6](#_Toc534129389)

[Supported Platforms 10](#_Toc534129390)

[Issues 12](#_Toc534129391)

[Desired Improvements 12](#_Toc534129392)

[Documentation of Code 13](#_Toc534129393)

[Instructions To Run Program 14](#_Toc534129394)

# Acknowledgement

I would like to express my great appreciation to Mr. Marius Varga for his guidance to the module. He give his time so generously by visiting Sri Lanka and lecturing our batch. Also he did many lab sessions to get familiar with C++ and QT Creator both. Especially because his guidance at the lab sessions I inspired to explore more about QT, C++, Database connected applications. From that experience I build this application which I named ‘PassVault’. Again I thanks Mr. Marius Varga for his great help.

# Introduction

I named my C++ application as ‘PassVault’. Using PassVault users can store their different kind of user account and passwords including social accounts. For example if you have Facebook, Gmail, yahoo and more accounts, you can store them with their passwords in the PassVault. It also help you to build strong password which will lengthy eight characters, contains capitals simples and symbols.

When using PassVault you do not need to worry about remember your passwords. You’re free to use strength passwords. PassVault will hold your details. Whenever you want you can simply logging to PassVault and check it.

Also there are some situations like multiple users are using same PC. At that time also users can create account for PassVault login. Then they can store their passwords. Likewise different users can use PassVault. PassVault will show password for real owners using its login functionality.

Anyone can watch running video of PassVault which is available on my YouTube channel from this link.

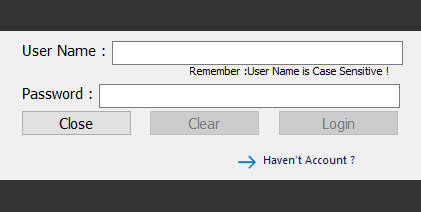
At the end of this documentation I provided instructions to run this program.

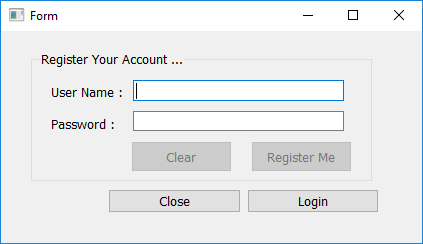
# Features

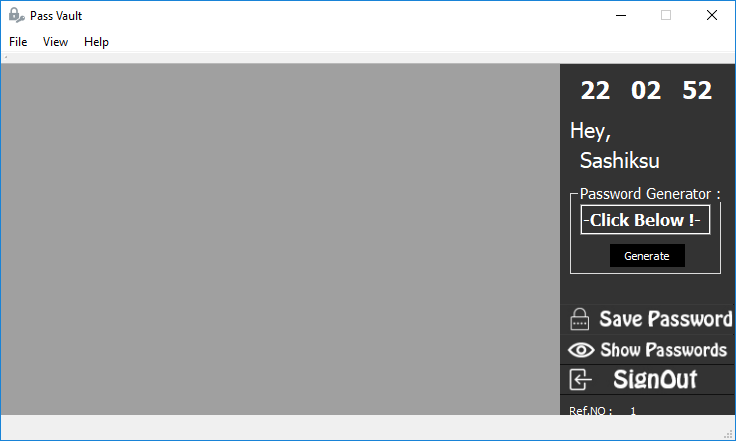
* Users can login to PassVault by creating user accounts.
* Storing Passwords with account’s username.
* Can retrieve all accounts and password details particular current logged user.
* Generating strong passwords.
* Let users delete their password details
* Let users update their password details.

# GUI Description

Splash screen will firstly show up and it will give user application is opening.

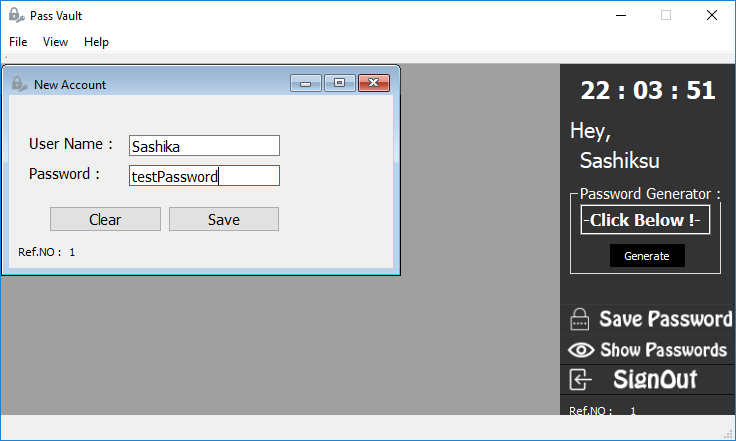
Login form contains two line edits and three buttons. By default two buttons disabled. When user enter their username and password both Clear and Login buttons will be enable. Also when user empty either one of line edit again two buttons will be disabled. If user is new to PassVault he/she can make new user account by click on ‘Haven’t Account’ command line link button. It will open new form as below.

As login form in this form also buttons are disabled until user inputs values to line edits. When user click on Register Me button it will create new user. Then user can go back to the Login form and enter those registering details and logging to PassVault. For the correct user credentials it will redirect to PassVault main MDI parent form.

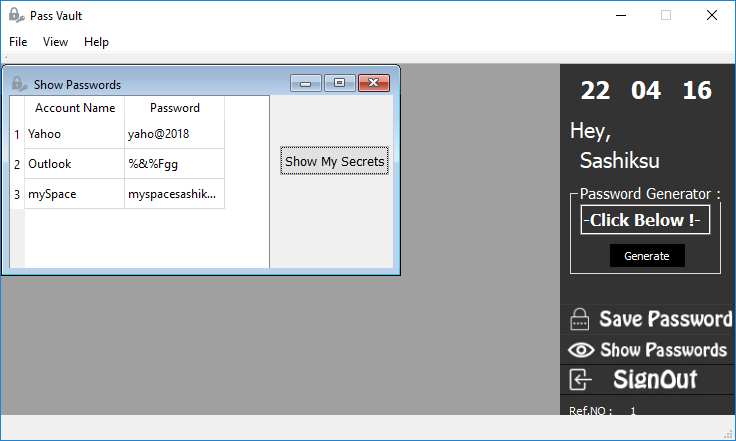
This is the main form of application. It has two main components. One is MDIArea and other is Frame. MDIArea will used to load all child windows inside it while frame contains digital clock, shortcuts and password generator.

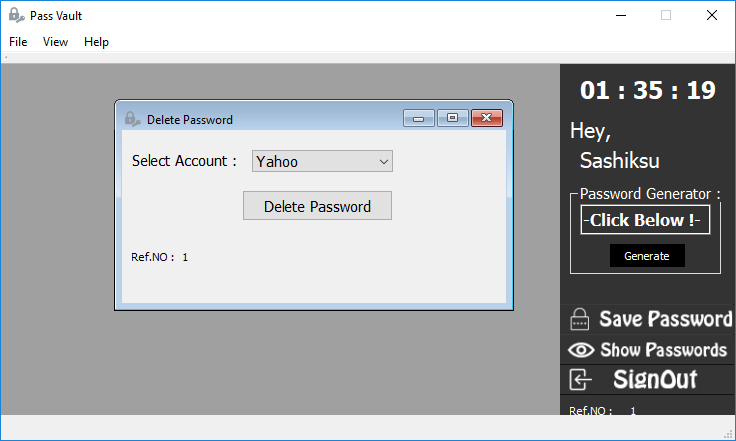
By clicking Generate button it will generate strong password containing simples, capitals, special characters, number. It is shows on line edit and user can easily copy and paste wherever they want to use it.

Shortcuts provide easy access to menu bar’s menu items.

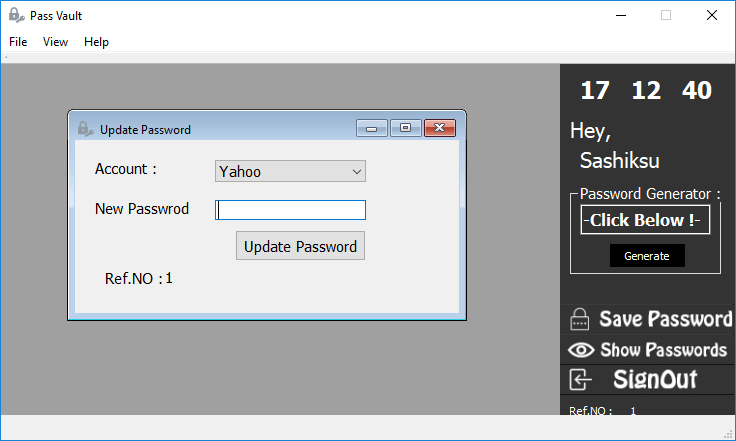


New Account form allow user to insert their usernames and passwords to PassVault. This form also let user to save when they provide both username and password.

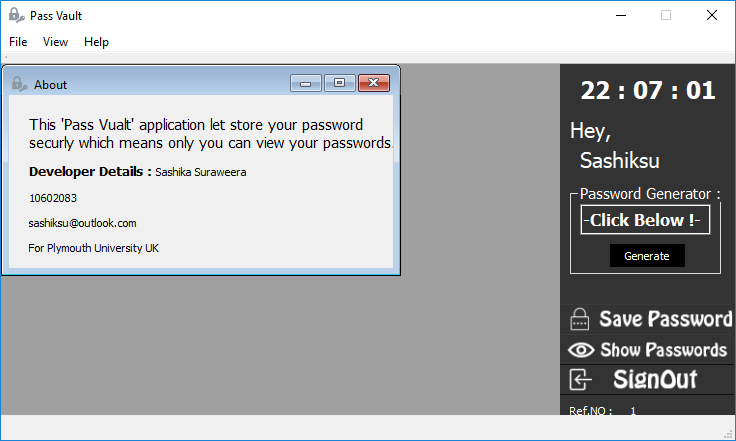
Show Password form contains tableView and button. By default tableView is empty. After click on ‘Show My Secrets’ button it will show Account Names and Passwords according to logged user.



In Delete Password form user can select which account to be delete and click on Delete Password. It will remove user’s account details. (Not PassVault user details).



In Update Password form user can update their accounts password and commit changes. It will update database.

About Form show developers details.

# Supported Platforms

PassVault developed using QT Creator which is running on Windows OS. Then I run it on Ubuntu to make sure it runs without any issues on cross platforms. Here are my evidences.

|  |  |  |
| --- | --- | --- |
| GUI | On Windows | On Ubuntu |
| Splash |  | |
| Login |  |  |
| Create user account for PassVault |  |  |
| MDI window of PassVault |  |  |
| New Password |  |  |
| Show Passwords |  |  |
| Delete Password |  |  |
| Update Password |  |  |
| About |  |  |

# Issues

* When child form maximized it extends to whole MDI Parent form and it’ll hide right side bar.
* Child forms could be resized when dragging it’s form boarders.
* Each child forms are initialized with same form size.

# Desired Improvements

I have many ideas to extend PassVault features. Below I’d mention some of them,

* Store PassVault login details with encryption.
* Embedded SQLite database to application so there won’t be any db path issues.
* Update existing username’s password.
* Reveal username’s password one by one using separate checkboxes which is in column of table.

Also fix issues mention above in issue topic.

# Documentation of Code

To make PassVault successful project I used Qt SQL module.

Qt SQL is an essential module which provides support for SQL databases. Since I used SQLITE for my PassVault project it help me lot. I used lot of methods inside the Qt SQL module. Those help to execute, retrieve, open and close connections, with the interaction to database.

In this section I mention only special codes in whole project. Other codes could be find in project source files.

* Database connection created and open in login.cpp source file. Because it is the first form interact with database. Then whenever wanted to perform SQL queries it uses already opened connection.

myDB=QSqlDatabase::addDatabase("QSQLITE");

myDB.setDatabaseName(Path\_to\_DB);

QFileInfo checkFile(Path\_to\_DB);

if(checkFile.isFile()){

if(myDB.open()){

//database connected code

myDB.open();

}

else{

//qDebug()<<"Not Connected !";

QMessageBox messageBox;

messageBox.warning(0,"Information","Database Error !");

}

}

* I used below method to generate strong password.

void PassVault::**GetRandomString**()

{

const QString possibleCharacters("`~!@#$%^&\*()-\_=+<,>.?/[]{}ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789");

const int randomStringLength = 8; // assuming you want random strings of 8 characters

QString randomString;

for(int i=0; i<randomStringLength; ++i)

{

int index = qrand() % possibleCharacters.length();

QChar nextChar = possibleCharacters.at(index);

randomString.append(nextChar);

}

ui->lineEdit->setText(randomString);

}

This code snip based on [this stackoverflow](https://stackoverflow.com/questions/18862963/qt-c-random-string-generation) thread. I modified it by adding special charters to the possibleCharacters variable and reduce randomStringLength to variable to 8 to produce strong password. This is method and its call inside Generate button click event.

These two are the special code review points of PassVault project.

# Instructions To Run Program

1. Copy the PassVault Project with SQLITE database file to the computer.
2. Get the correct location of database file
3. Open PassVault with Qt Creator.
4. Open Login.cpp file and find following line in upper of code lines.

#define Path\_to\_DB "C:/Users/User/Documents/my.sqlite" //this is for windows

1. Replace copied db file location in between “ “ marks.
2. Check again to confirm you provided the right location with extension.