# Program Specifications

|  |  |
| --- | --- |
| **Project Name:** | **SimpleLogin** |
| Programmer: | Ming Chi Hsiao |
| Supervisor: | Jawdat Moussa, Software Development Team Leader |
| Purpose of program: | Create a simple way for user to login with their password and account name |
| Programming language: | python |
| Software to be used: | Visual studio code |
| Project brief:  *Simple English description of the program to be created (maximum 50 words)* | **Login** – for users who have previously registered. Username and password to be checked for validity.  **Register** – create a new user account.  **Passwords** – new users given the option to enter their own password or generate one.  **Generated passwords** – user given the option to choose the password character types – numbers, symbols or letters. The default password length should be applied, but users should also be allowed to choose how many characters.  **Save file**– usernames and passwords should be saved to a text file **accounts.txt.**  **Exit** – delay for 2 seconds before exit.  **View accounts** – to display user account information from the accounts.txt file (assuming that only admin have access to this program). |

### Table of Contents

[Program Specifications 1](#_Toc81985354)

[Coding Rules 3](#_Toc81985355)

[Data Library Variables 3](#_Toc81985356)

[Variable Scope 4](#_Toc81985357)

[Program Test Data 5](#_Toc81985358)

[Program Algorithm 6](#_Toc81985359)

[Program Code 7](#_Toc81985360)

[Software Test Report 8](#_Toc81985361)

[Overview 8](#_Toc81985362)

[Changes Required from Feedback 9](#_Toc81985363)

[Test Cases 10](#_Toc81985364)

[Software Evaluation 12](#_Toc81985365)

[Programmer’s Checklist 13](#_Toc81985366)

[Software Final Approval 14](#_Toc81985367)

[When you have completed your report, select the table (above) and press F9 to update the entire table. Once done, delete this text]

Coding Rules

|  |  |
| --- | --- |
| **Rule** | **Brief explanation of rule** |
| 1. safe | The coding should be able to be used and doe not cause harm or damage to others or company. |
| 1. Secure | The coding should be secure, others without authority should not be able to access the information and document involved. |
| 1. Reliable | The coding should be functional all the time and every time the function should be able to be the same. |
| 1. Testable | The coding should be testable and debugged, to make sure all the designed function are working condition. |
| 1. Maintainable | The code should be able to be maintained, so the coding can always to make sure has the best performance as designed. |

Data Library Variables

|  |  |
| --- | --- |
| **Variable name** | **How variable will be used** |
| **Login:** |  |
| Username | Use to storage the user input login username |
| Password | Use to storage the user input login password |
| Accounts\_username | Use to storage the account\_username which we have in our data. |
| Accounts\_password | Use to storage the account\_password which we have in our data |
| ErrorMessage | Use to storage the error message send to user when the information the user provide is wrong. |
| **Register:** |  |
| username | Use to storage the user input login username |
| password | Use to storage the user input login password |
| accounts\_username | Use to storage the account\_username which we have in our data. |
| accounts\_password | Use to storage the account\_password which we have in our data |
| ErrorMessage | Use to storage the error message send to user when the information the user provide is wrong. |
| **Account View:** |  |
| username | Use to storage the user input login username |
| password | Use to storage the user input login password |
| accounts\_username | Use to storage the account\_username which we have in our data. |
| accounts\_password | Use to storage the account\_password which we have in our data |
| ErrorMessage | Use to storage the error message send to user when the information the user operate is wrong. |

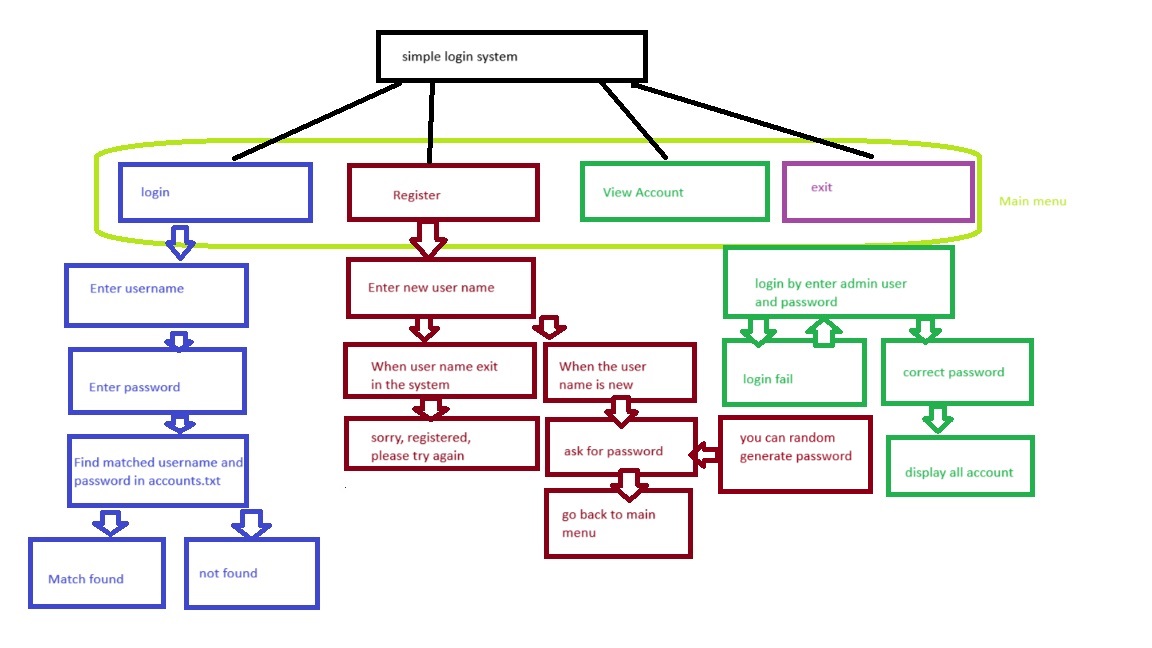
Variable Scope

|  |  |
| --- | --- |
| **Scope** | **Variable Name** |
| Local | Login variable, all local scopes to the function login  Username, password, Accounts\_username, Accounts\_password, Error message |
| Local | Register variable, all local scopes to the function register  Username, password, Accounts\_username, Accounts\_password, Error message |
| Local | AccountsView variable, all local scopes to the function accountsView  Username, password, Accounts\_username, Accounts\_password, Error message |
| Choose an item. |  |
| Choose an item. |  |
| Choose an item. |  |
| Choose an item. |  |
| Choose an item. |  |
| Choose an item. |  |
| Choose an item. |  |
| Choose an item. |  |
| Choose an item. |  |
| Choose an item. |  |
| Choose an item. |  |
| Choose an item. |  |
| Choose an item. |  |

Program Test Data

|  |  |
| --- | --- |
| **Username** | **Password** |
| fredsmart1 | 12345678 |
| jrobertson4 | r@=%8(\_W=1 |
| bob101 | 1234598 |
| marcusw | 3#tr@9dw%4 |
| popeyedd | 1989eidjce |
| junkman00 | p3\*(kd8&ld |
| sbj2021 | $d5e(ep2(d |
| robotman | 7777Spy007 |

Program Algorithm



Program Code

#Ming Chi Hsiao, 27/05/2024, simple login programS

#import packages

import sys ,time   # internal modue

import string, random # internal modules

def generate\_random\_password():

    ## characters to generate password from

    characters = list(string.ascii\_letters + string.digits + "!@#$%^&\*()")

    ## length of password from the user

    length = int(input("Enter password length: "))

    ## shuffling the characters

    random.shuffle(characters)

    ## picking random characters from the list

    password = []

    for i in range(length):

        password.append(random.choice(characters))

    ## shuffling the resultant password

    random.shuffle(password)

    ## converting the list to string

    ## printing the list

    print("".join(password))

    return "".join(password)

# login function

def Login():

    '''

    version1:

    def UserLogin():

    Username=input("please enter your username: ")

    Password=input("please enter your password: ")

    with open("accounts.txt","r") as file:

        content = file.readlines() # generate a list

        for line in content:

            strippedline=line.split(" ")

            print(strippedline)

            getuser=strippedline[0].strip()

            getpassword=strippedline[1].strip()

            print(getuser, getpassword)

            if(Username==getuser and Password==getpassword):

                print(getuser, getpassword)

                print("welcome back")

                return "login successful"

    print("login fail")

    return None

    input username and password:

    username = ask the user for username:

    password = ask the user for password:

    '''

    username = input("Please Enter your Username: ")

    password = input("Please Enter your Password: ")

    # read accounts.txt and search for valid username:

    file = open("accounts.txt", "r")  # Open accounts file for reading

    info = file.read() # read the whole as a string

    # testing reading the accounts.txt file

    #print(">>>"+info+"<<<") # I proved the file is a string

    file.close()

    infolist= info.split()

    # test the contents of the string that we read from accounts

    #print(infolist)

    if username in infolist: # check my username is already in accounts.txt

        index = infolist.index(username) +1 # gent the index of the password in accounts.txt

        account\_password = infolist[index]

        #test if it work by print it out

        #print(account\_password)

        if password == account\_password:

            print("Welcome Back "+ username+ " Login Successful")

            return [True,username]

        else:

            print("Sorry "+ username + ": password invalid")

            return [False,username]

    else:

        print("Sorry  "+ username+ " Login denied")

        return [False,username]

# Register function

def Register():

    username = input("Please Enter a valid username: ")

    file = open("accounts.txt", "r")  # Open accounts file for reading

    info = file.read() # read the whole as a string

    file.close()

    #print(info)

    if username in info:

       return print("Sorry username:  "+ username + " already Exisits")

    else:

        print("============= password handling ===================")

        print("Enter (a) to create your own password, or (e) to generate password")

        p = input("Please choose (a) or (e) : " )

        if p == "a":

            password = input("Enter your own password: ")

        elif p == "e":

            password = generate\_random\_password()

        # now ready to register username and assword:

        with open("accounts.txt", "a") as file:

            #writeline = "\n"+username + " " + str(password)

            #print(writeline)

            file.write("\n"+username + " " + password)

            print("Your account has been registered ")

            print("Please login")

def View():

    Administrators = ["admin", "admin2" ]

    string = " ".join(Administrators)

    #print(string)

    print("------------------------------------\n")

    print("Entering View Accounts() procedure  \n")

    print("------------------------------------\n")

    #open the file in read mode and read line by line and display on

    #screen. This will display all the account details from the file.

    validlogin, username = Login()

    #print(validlogin, username)

    if validlogin:

        print("List of accounts (usernames and Passwords):\n")

        f = open("accounts.txt","r")

        for line in f:

            if username in Administrators:

                userlist, password =line.split(" ")

                #print(userlist,"==", password)

                #print(userlist)

                if userlist not in Administrators:

                    print(line.strip())

            if username not in Administrators:

                print("Sorry Only Adminstrators can view accounts")

                return

        f.close

    else:

        print("Sorry Only Adminstrators can view accounts")

def logoff(i):

    #option c exits the program after a delay of 2 seconds

    print("\n Signal "+str(i) + " Exiting program...")

    time.sleep(2)

    sys.exit()

def UserProcess():

    username = ""

    password = ""

    print("\n\nWelcome to Login and Registration system...")

    print("\nWould you like to:")

    #print the main menu

    print("\nL) Login")

    print("R) Register new account")

    print("V) View accounts")

    print("Q) Quit")

    #enter the menu choice

    i = input("\nChoose [l/r/v/q]: ")

# Login option

    if (i == "l" or i == "L"):

        Login()

    elif (i == "r" or i == "R"):

        Register()

    elif (i == "v" or i == "V"):

        View()

    elif (i == "q" or i == "Q"):

        logoff(i)

while True:

    UserProcess()

# Software Test Report

Overview

This document is the software test report of the testing phase of the **[project name]** software development project. It contains the results of tests, which were executed during the testing phase.

|  |  |
| --- | --- |
| Name: | Ming Chi Hsiao |
| Date: | 27/5/2024 |
| Software being tested: | 27/5/2024 |

Changes Required from Feedback

|  |  |  |
| --- | --- | --- |
| **Object to be changed** | **Changes required** | **Completed** |
| *Sample*  *Spelling error in main menu “optoins” should be “options”* | *Correct the spelling error* |  |
| *Wrong Variable used* | *Correct the variable to correct name* |  |
| Repeat message appear | Add return to the loop |  |
| The view account function show the admin information as well | Add statement for avoid show admin information |  |
| Register function overwrite the whole accounts.txt | Open file to “a” type |  |
| Register function don’t change line after append new account name and password | Put \n in the front of file.write() attribute |  |
| View account won’t show the false message when normal user login | Add if statement into the view account |  |
| System wont log off | Import sys |  |

Test Cases

| **Test case ID** | **Test case name** | **Summary** | **Expected results** | **Screenshot filename** | **Actual result** |
| --- | --- | --- | --- | --- | --- |
| **Test01** | **Test the login options in main menu** | 1. **Enter username** 2. **Enter password** 3. **Login failed** 4. **Login successful** | **Login screen is displayed** | **Loginphoto.jpg** | **Login screen is displayed** |
| Test02 | **Registration** | 1. Unique username 2. Password type 3. Password length | The accounts.txt file will have a new username and a password with minimum 10 characters | Registerphoto.jpg | The accounts.txt file has a line of new username and password with minimum 10 characters. |
| Test03 | View accounts | 1. Admin login only view accounts 2. Accounts listed if admin is valid 3. Admin can’t be listed | The user can login in only via admin username and password. The program will show the account list when the admin username and password is provided | Viewaccountphoto.jpg | The program provides the list of accounts |
| Test04 | Logoff | Sign off with sleep of 2 seconds | The program will login out system | Loginoutphoto.jpg | The program login out system |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Software Evaluation

|  |  |
| --- | --- |
| **Software Specification**  (As listed in the software specifications)  (No more than 40 words per response) | **Evaluation**  (How well does your software meet this specification?)  (No more than 20 words per response) |
| ***Sample***  *Menu displays when program starts* | *Menu displays when program started with no spelling or grammatical errors*  ***Describe how your login system comply with the software specifications.*** |
| Design simple login system using python | We design the software using python as per user requirements. |
| Login to accounts for registered users | The user can login the account only by entering the username and password in the accounts file, this is complied with the requirements. |
| Register a accounts with unique name, with password options either plain text of encrypted of length of ten chars minimum | The user can register themselves by enter a username different from the user name in the accounts file and type their own password with minimum ten in length, and they also can choose to generate a password and assign the length of password they want. This complied with the requirement. |
| View accounts procedures to allow the administrators to list users accounts. Only admin can view accounts | The user can login to view the account list by enter the admin username and password, this will provide a list of all username and password besides the admin username and password. This is complied with the requirements. |
| The login system must have a sign off module | The user can sign off the program |
|  |  |

# Programmer’s Checklist

Check that each of these items have been completed and **have been recorded in the Gelos Software Design Document** in the appropriate place.

|  |  |
| --- | --- |
| **Item to be checked** | **Completed** |
| Gelos Software Design Document fully completed with all details |  |
| Coding rules for chosen language included |  |
| Data library variables defined |  |
| Variable scope defined |  |
| Library of functions defined |  |
| Program test data file created |  |
| Program algorithm designed |  |
| Program code created |  |
| Program code uses good design principles (white space, indents, etc) |  |
| Program code is commented throughout |  |
| Test cases completed |  |
| Screenshots correctly named and saved and zipped |  |
| Software evaluation completed |  |
| Changes required from feedback completed |  |
| Self-Evaluation completed |  |

# Software Final Approval

***Supervisors use only***

|  |  |
| --- | --- |
| **Software Approval** |  |
| Name: |  |
| Role: |  |
| Program meets design specifications: |  |
| Signature: |  |
| Date: |  |