IFY Assessment Cover Sheet

Student Name	Lwin Min Khant
NCUK Student ID	117578
Module	Scenario 3
Coursework 1 / 2 (etc.)	Coursework 3
Lecturer/Tutor	Dr Hnin Lelt Win
Date Submitted	9.10.2024

Academic Misconduct: Student Declaration

All forms of academic misconduct (e.g. plagiarism, collusion, fabrication of results and subcontracting and the use of translation services) are regarded seriously by NCUK and could result in penalties, including a zero mark (failure) and possible disciplinary action. Types of academic misconduct include:

- **Plagiarism** - Copying information, thoughts or ideas from a published or unpublished source without acknowledging (showing in your work) where that information, thoughts or ideas came from
- **Collusion** - Where two or more students work together to produce individual assessments that contain the same ideas and text
- **Fabrication of Results** - Where a student presents a set of results that are not from his/her observations or calculations
- **Subcontracting** - Where a student receives help from someone else with his or her assessment, this may be via a paid for service (also known as contract cheating) or by using friends and family.
- **Translation Services -** where a student uses a person or service (including online tools) to translate - into English - some or all their work from another language. *This type of academic misconduct applies only assessments that contribute to your EAP, EAPPU or RCS grade.*

DECLARATION

I declare that all material in this assessment is my own work and that I have given fully documented references to the work of others.

Signed: _____Lwin_____ Date: ___9.10.2024_____

NCUK INTERNATIONAL FOUNDATION YEAR

Computer Science

Coursework

Scenario 3

New User Account Registration

Contents List

1. DESCRIPTION OF INVESTIGATION

The new user account registration in scenario three is a program that allows users to create their accounts and privacy information, and enables them to create, edit, and delete their accounts as necessary. This program is usually used for managing the registered accounts, such as schools, offices, and some shopping malls. The program has detailed the requirements for the user during creating a new account such as entering the user's date of birth and phone number for creating a new account then other processes.

This program required gathering users' information and privacy, and finding their accounts by using the user's phone number. The requirement for this investigation of the program has been written by Lwin Min Khant.

The primary aim of this system is to collect and manage user data while ensuring that follows the instructions such as validating passwords for security. In this program, the password must meet the following criteria:

- Be at least 8 characters long
- Contain at least one letter
- Contain at least one number
- Contain at least one special character

As the user has created a new user account from a different users, the program allows the users to find their registered account or view it. The program not only allows users to view their user but also to amend their account or delete it if it is unnecessary. To review, amend, and delete accounts,

the program requires the user's "First Name" and "Last Name". These requirements could make registered accounts more secure and safer without interacting with other user accounts and their detailed information.

While amending the user account, "First Name" and "Last Name" of the previously registered account, the program will let the user input their desired input or the place they want to edit or change.

Similar to deleting the user account and information, the program requires the user to enter the "First Name" and "Last Name" of the previous account information that the user wants to delete.

This program follows the Waterfall Model of software development. The code is designed to allow user registration, retrieval, and searching by phone number. Each stage of the Waterfall Model is addressed during development.

2. JUSTIFICATION OF INVESTIGATION

In this investigation, a user registration system is a fundamental functionality found in many applications, such as websites to mobile apps, and developing this system allows for practical application. Example: data storage, validation, and retravel. This coursework offers a great chance to demonstrate skills such as input validation and creating a user-friendly menu interface, making it a suitable and meaningful challenge for the program scenario three.

3. ANALYSIS, DESIGN, AND METHODS USED

Analysis

The program in scenario three requires a program that allows users to be registered with basic details and offers functions such as searching for users and viewing the registered users. For the solution, the system should store users in a list and allow interaction through a simple menu interface.

Functional Requirements

For the functional requirement, this program must ensure that user create new accounts by inputting their first name, last name, date of birth, phone number, password, and address. These password validations should be at least 8 characters long and include letters, digits, and some special characters Users can view all registered users, and the program also allows a user to use their phone number and displays their details if found. The program should offer functionality to edit user details by specifying the user's "First Name" and "Last Name," and users must be able to delete a registered user by their "Full Name."

Non-Functional Requirements

The program needs to be friendly to the user, with a menu that makes it easy for users to navigate and perform tasks. The program must handle errors, such as invalid inputs or missing records, gracefully without crashing and passwords should be securely validated, to make sure that weak passwords are rejected such as user's passwords, should not be stored or displayed in plain text

and the code should be organized into modules for tasks like creating, searching, editing, and deleting, making future updates easier.

4. DECOMPOSITION

Measurable

The features of the system can be measured to ensure that it works as intended.

1. User Creation

In the creation, the program should allow users to create accounts with valid information. Testing will be used to create multiple accounts to prove all of the required fields that are correctly filled and saved to the internal storage.

2. Password Validation

For password validation, the user needs to follow the security requirements of a password such as at least 8 characters, a number, a letter, and a special character. The program will be tested by edit to enter weak passwords to confirm that they are rejected.

3. Search for a User

When searching for a user in a program, it must correctly find users based on the user's first and last names. Entering the valid user names will return the correct user information, while non-existent users will be sparked with an appropriate error message.

4. Editing User Details

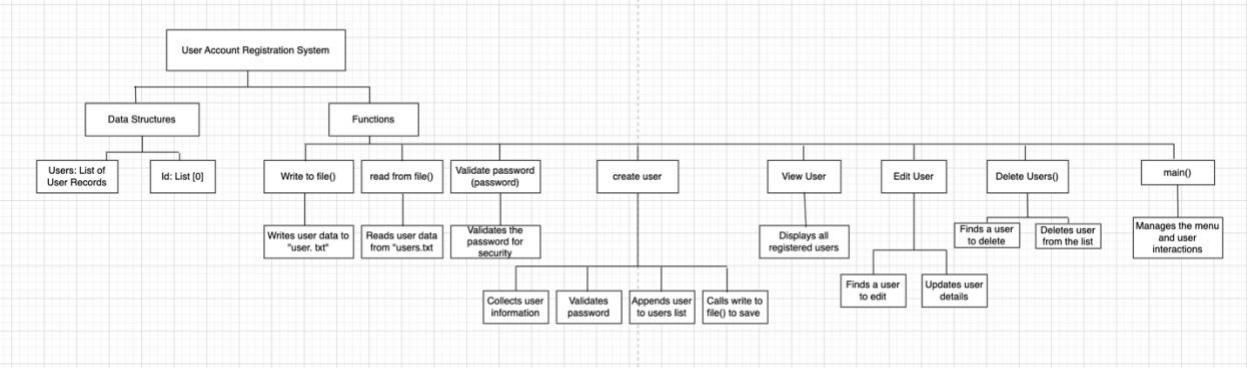
When editing the information the users should be able to update their details like name, phone, or address. The program will be tested by creating the existing user information and checking that the changes are saved.

5. Deleting a User

The program needs to successfully remove users from the database. After deletion, the program will be checked to ensure the user is no longer on the list.

6. Error Handling

The program should manage errors smoothly without causing issues. Users will intentionally create the mistakes of input such as leaving the fields empty or using invalid password formats, to check if the program can handle them correctly.

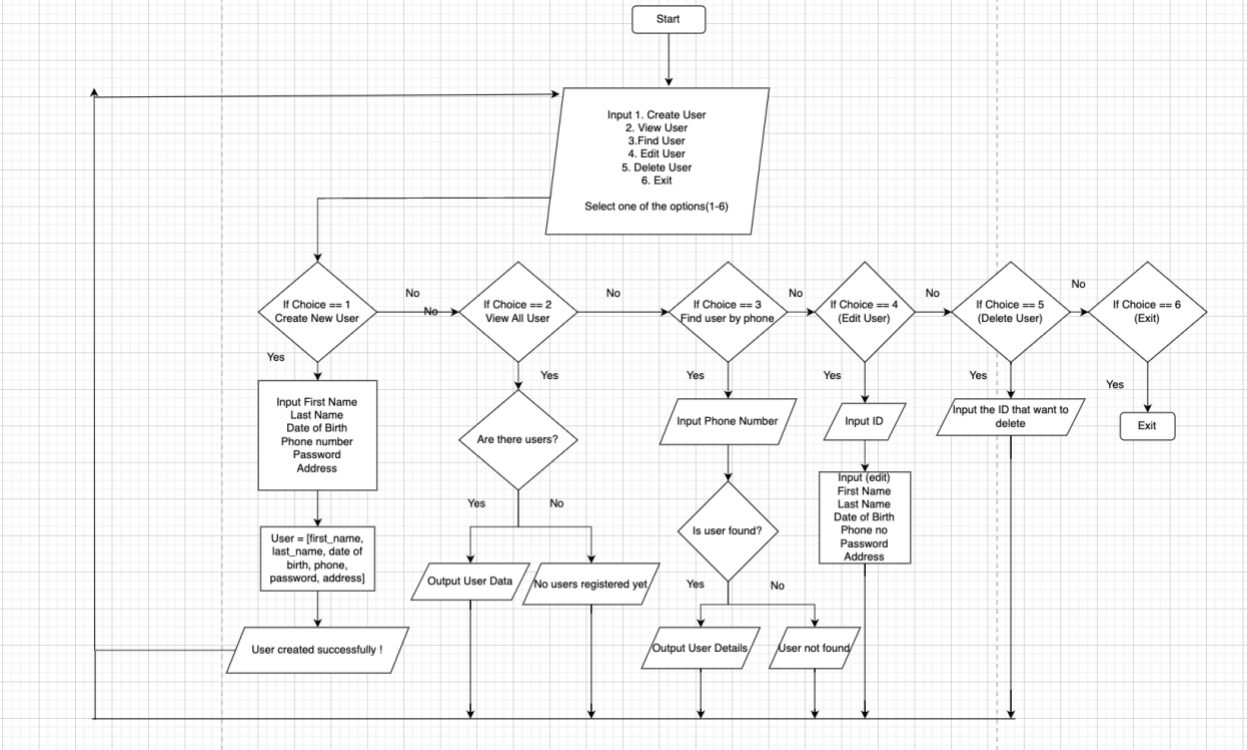


(Structural diagram designed by Lwin Min Khant using “draw io” tools)

5. DESIGN

Flowchart

Each main functions in the program are arranged with each of its steps and each function and process with different shapes. Then connected with arrows in the flowchart below.



(Flow chart designed by Lwin Min Khant using “draw io” tools)

6. CODING

This program will be using the Python Community Edition 2023. The flowchart procedures are all converted into Python code and are combined to create a “User Registration System”.

The code:

```python
import os

# List to store user information during registering
users = []
Id = [0]


# Function to write users to a file
def write_to_file():
    with open("users.txt", "w") as file:
        for user in users:
            # Save user data separated by commas
            file.write(",".join(user) + "\n")

```

Figure 4.0

```python

# Function to read users from a file
def read_from_file():
    # Check if the file exists
    if os.path.exists("users.txt"):
        with open("users.txt", "r") as file:
            for line in file:
                # Read each line and split it into a list
                user = line.strip().split(",")
                users.append(user)
            # Update Id counter to reflect the number of users
            if users:
                Id[0] = int(users[-1][6]) + 1

```

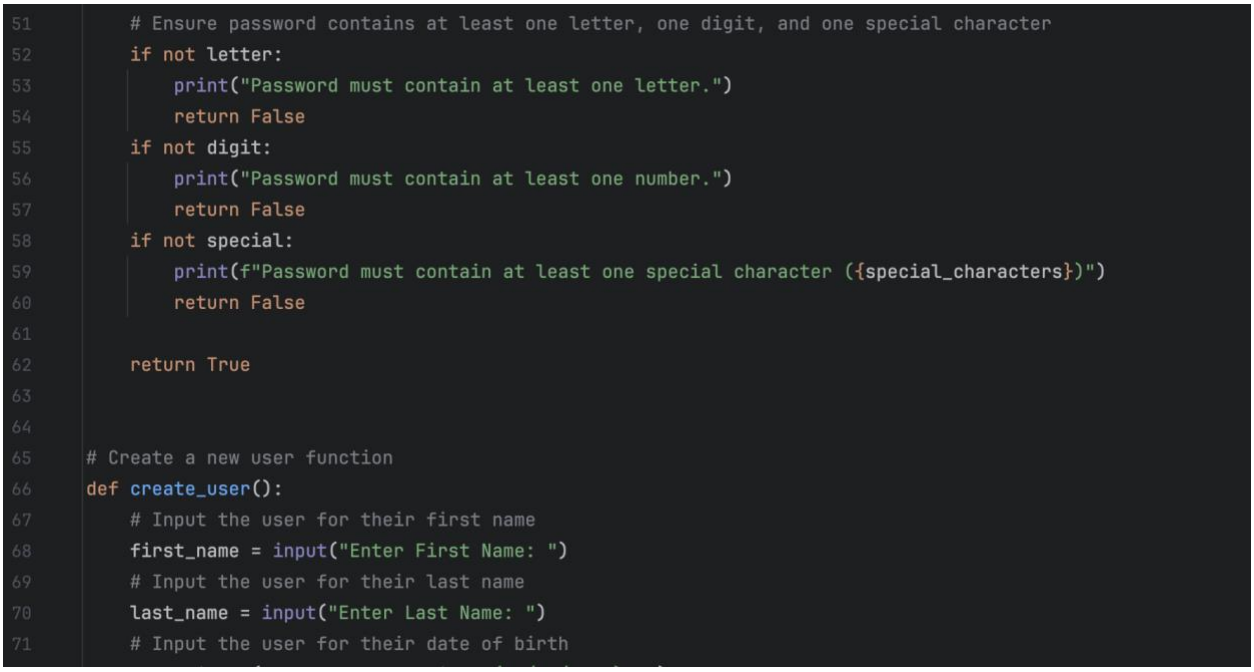
Figure 4.1

```
# Function of validation password
def validate_password(password):
    # Check if the password is at least 8 characters long
    if len(password) < 8:
        print("Password must be at least 8 characters long.")
        return False

    letter = False
    digit = False
    special = False
    special_characters = "!@#$%^&*()"

    # Check each character manually
    for char in password:
        if char.isalpha():
            letter = True
        elif char.isdigit():
            digit = True
        elif char in special_characters:
            special = True
```

Figure 4.2



```
    # Ensure password contains at least one letter, one digit, and one special character
    if not letter:
        print("Password must contain at least one letter.")
        return False
    if not digit:
        print("Password must contain at least one number.")
        return False
    if not special:
        print(f"Password must contain at least one special character ({special_characters})")
        return False

    return True


# Create a new user function
def create_user():
    # Input the user for their first name
    first_name = input("Enter First Name: ")
    # Input the user for their last name
    last_name = input("Enter Last Name: ")
    # Input the user for their date of birth
```

Figure 4.3

```
72      dob = input("Enter Date of Birth (DD/MM/YYYY): ")
73      # Input the user for their phone number
74      phone = input("Enter Phone Number: ")
75
76      # Validate the password
77      while True:
78          password = input("Enter Password: ")
79          if validate_password(password):
80              break  # If the password is valid, break the loop
81
82      address = input("Enter Address: ")
83
84      # Store the user information in a list
85      user = [first_name, last_name, dob, phone, password, address, str(Id[0])]
86
87      # Append the new user to the users list
88      users.append(user)
89      # Save to the file after creating the user
90      write_to_file()
91      # Notify that the user has been created successfully
92      print("User created successfully! with an ID of:", Id[0])
```

Figure 4.4

Figure 4.5

```
97  # Function of registered users
98  def view_users():
99      # Check if there are any registered users
100     if len(users) == 0:
101         # Notify if no users are registered
102         print("No users registered yet.\n")
103     else:
104         # Loop through the users list and print their details
105         for user in users:
106             print(f"Name: {user[0]} {user[1]}, Phone: {user[3]}, ID: {user[6]}")
107         print()
108
109
110 # Function of finding a user by first name and last name
111 def find_user():
112     first_name = input("Enter First Name to find: ")
113     last_name = input("Enter Last Name to find (optional, press Enter to skip): ")
114
115     found = False
116     for user in users:
117         if user[0].lower() == first_name.lower() and (last_name == "" or user[1].lower() == last_name.lower()):
118             print("\n--- User Details ---")
```

Figure 4.6

```
            print(f"First Name: {user[0]}")
            print(f"Last Name: {user[1]}")
            print(f"Date of Birth: {user[2]}")
            print(f"Phone: {user[3]}")
            print(f"Address: {user[5]}")
            found = True
            break

    if not found:
        print("User not found.\n")


# Function to edit a user by first name and last name
def edit_users():
    first_name = input("Enter First Name to find: ")
    last_name = input("Enter Last Name to find (optional, press Enter to skip): ")

    found_user = None

    for user in users:
        if user[0].lower() == first_name.lower() and (last_name == "" or user[1].lower() == last_name.lower()):
            found_user = user
```

Figure 4.7

```
    if found_user is None:
        print("User not found.\n")
        return

    print("\n--- Editing User ---")
    new_first_name = input(f"Enter New First Name (current: {found_user[0]}): ") or found_user[0]
    new_last_name = input(f"Enter New Last Name (current: {found_user[1]}): ") or found_user[1]
    new_dob = input(f"Enter New Date of Birth (current: {found_user[2]}): ") or found_user[2]
    new_phone = input(f"Enter New Phone Number (current: {found_user[3]}): ") or found_user[3]
    new_address = input(f"Enter New Address (current: {found_user[5]}): ") or found_user[5]
    new_password = input(f"Enter New Password (current: {found_user[4]}): ") or found_user[4]

    found_user[0] = new_first_name
    found_user[1] = new_last_name
    found_user[2] = new_dob
    found_user[3] = new_phone
    found_user[4] = new_password
    found_user[5] = new_address

    # Update the file after editing user details
    write_to_file()
    print("User details updated successfully.\n")
```

Figure 4.8

```
# Function to delete a user by first name and last name
def delete_users():
    first_name = input("Enter First Name to find for deletion: ")
    last_name = input("Enter Last Name to find for deletion (optional, press Enter to skip): ")

    found_user = None

    for i, user in enumerate(users):
        if user[0].lower() == first_name.lower() and (last_name == "" or user[1].lower() == last_name.lower()):
            found_user = i
            break

    if found_user is None:
        print("User not found.\n")
        return
    # Delete the found user
    del users[found_user]
    # Update the file after deletion
    write_to_file()
    print("User deleted successfully.\n")
```

Figure 4.9

```
# Function of menu system
def main():
    # Load users from the file when the program starts
    read_from_file()

    # Start an infinite loop to keep the program running
    # Display menu options
    while True:
        print("1. Create New User")
        print("2. View All Users")
        print("3. Find User by Name")
        print("4. Edit User")
        print("5. Delete User")
        print("6. Exit")

        # Input the user for their choice
        choice = input("Choose an option (1-6): ")

        # Handle the user's choice
        if choice == "1":
            # Call the function to create a new user
            create_user()
        elif choice == "2":
```

Figure 5.0

```
        # Call the function to view all registered users
        view_users()
    elif choice == "3":
        # Call the function to find a user by name
        find_user()
    elif choice == "4":
        # Call the function to edit a user by name
        edit_users()
    elif choice == "5":
        # Call the function to delete a user by name
        delete_users()
    elif choice == "6":
        # Notify that the program is exiting
        print("Exit")
        # Exit the loop and end the program
        break
    else:
        # Notify if the user input is invalid
        print("Invalid option, please enter a number between 1 and 6.\n")
```

Figure 5.1

```
231
232
233 # Run the program
234 if __name__ == "__main__":
235     main()
236
```

Figure 5.2

7. METHOD USED

Input validation

To make sure the code and user input are captured and stored accurately, with basic validation such as checking for valid numbers and strings.

Menu-driven interface

The program runs in a loop, presenting the user with different choices and responding based on the user input.

8. EVALUATION

Effectiveness

1. User Account Creation

   The program allows users to register with valid details, and password validation effectively ensures security and ignores weak passwords then all user data is accurately stored. However, for future improvements, storing passwords in an encrypted format would improve security.

2. Password Validation

   In password validation, the user needs to ensure that only strong passwords are accepted and weak passwords are consistently rejected, which helps to maintain accounts more securely. In the future improvements could include giving more detailed feedback on why a password is not available.

3. Account Retrieval

   For user retrieval, the user can easily search their account by full name, and the program provides the specific information in details that display user data when valid input is provided. When there are error messages the output will be “user not found”.

4. Editing User Details

   When amending the user the program allows to change the reflected instantly in the stored data. It applied without any issues and the system prevents errors from the invalid entries. But more advanced validation for certain fields could be added in future improvements

5. Deleting a User

   In the deletion, the function works efficiently by removing the unnecessary account without affecting the remaining data. Deleting the user from the program still needs to get more improvement to add some useful messages before deleting the user to delete the wrong account.

6. Error Handling

   The program does a great job of handling mistakes without errors and gives helpful messages. The error messages are clear and help users fix their errors. In the future, adding more specific advice could make the experience even better.

9. TESTING

The system was tested using various test cases to ensure it behaved as expected. Below are the tests that were conducted:

Test Number	Input Data	Purpose of Test	Expected Output	Actual Output
Test 1: Create New User	First Name: Lwin Last Name: Min Khant DOB: 25/10/2006 Phone: 09788407578 Password: Pass@123	Test creating a new user	User created successfully! with an ID of : 0	User created successfully! with an ID of: 0

	Address: MDY			
Test 2: View All Users	N/A	Check if users list displays correctly	Name: Lwin Min Khant Phone: 09788407578 ID: 0	Name: Lwin Min Khant Phone: 09788407578 ID: 0
Test 3: Find User by Nmae	First Name: Lwin Last Name: Khant	Test finding a user by First Name and Last Name	First Name: Lwin Last Name: Min Khant,…..	First Name: Lwin Last Name: Min Khant,…..
Test 4: Create Another User	First Name: Aung Last Name: Min Phyo DOB: 10/5/2005 Phone: 0978432345 Password: Test@456 Address: MYN	Test creating a second user	User created successfully! with an ID of : 1	User created successfully! with an ID of : 1

Test 5: Edit User Details	ID:0 First Name: Lwin Last Name: Min Khant DOB: 26/10/2006 Phone: 097383243 Password: Pass@123 Address MDY	Test editing a user's details by ID	Successfully edited	Successfully edited
Test 6: View All Users (After Edit)	N/A	Check if the user's edited details display	Name: Lwin Min Khant Phone:09788407578 ID:0	Name: Lwin Min Khant Phone:09788407578 ID:0
Test 7: Find Edited User by Phone	User Name: Lwin Mi Khant	Test finding the edited user	First Name: Lwin Last Name: Min Khant,….	First Name: Lwin Last Name: Min Khant,….
Test 8: Delete User	Enter delete user name:	Test deleting a user	No specific output, user ID 1 is deleted	No specific output, user ID 1 is deleted

Test 9: View All Users (After Deletion)	N/A	Check if the user list displays correctly	Only the remaining user should be shown	Only the remaining user should be shown
Test 10: Invalid Password	Password: short	Test password validation for short passwords	Password must be at least 8 characters long.	Password must be at least 8 characters long.
Test 11: Invalid Password(No Special Character)	Password: Pass 1234	Test password validation for missing special characters	Password must contain at least one special character (!@#$%^&*())	Password must contain at least one special character (!@#$%^&*())

```
1. Create New User
2. View All Users
3. Find User by Name
4. Edit User
5. Delete User
6. Exit
Choose an option (1-6): 1
Enter First Name: Lwin
Enter Last Name: Min Khant
Enter Date of Birth (DD/MM/YYYY): 25/10/2006
Enter Phone Number: 09788407578
Enter Password: Pass@123
Enter Address: MDY
User created successfully! with an ID of: 3
```

```
1. Create New User
2. View All Users
3. Find User by Name
4. Edit User
5. Delete User
6. Exit
Choose an option (1-6): 1
Enter First Name: Aung
Enter Last Name: Min Phyo
Enter Date of Birth (DD/MM/YYYY): 10/5/2005
Enter Phone Number: 0978432345
Enter Password: 1234
Password must be at least 8 characters long.
```

```
Choose an option (1-6): 1
Enter First Name: Aung
Enter Last Name: Min Phyo
Enter Date of Birth (DD/MM/YYYY): 10/5/2005
Enter Phone Number: 0978432345
Enter Password: 1234
Password must be at least 8 characters long.
Enter Password: Test@456
Enter Address: MYN
```

```
Password must be at least 8 characters long.
Enter Password: Test@456
Enter Address: MYN
User created successfully! with an ID of: 4
1. Create New User
2. View All Users
3. Find User by Name
4. Edit User
5. Delete User
6. Exit
Choose an option (1-6):
Invalid option, please enter a number between 1 and 6.
```

```
Choose an option (1-6): 2
Name: Lwin Min, Phone: 0934324, ID: 0
Name: Lwin Min, Phone: 0934234423, ID: 1
Name: Lwin  Khant, Phone: 09342345, ID: 2
Name: Lwin Min Khant, Phone: 09788407578, ID: 3
Name: Aung Min Phyo, Phone: 0978432345, ID: 4

1. Create New User
2. View All Users
3. Find User by Name
4. Edit User
5. Delete User
6. Exit
Choose an option (1-6):
```

```
4. Edit User
5. Delete User
6. Exit
Choose an option (1-6): 3
Enter First Name to find: Aung
Enter Last Name to find (optional, press Enter to skip): Min Phyo

--- User Details ---
First Name: Aung
Last Name: Min Phyo
Date of Birth: 10/5/2005
Phone: 0978432345
Address: MYN
```

```
6. Exit
Choose an option (1-6): 4
Enter First Name to find: Aung
Enter Last Name to find (optional, press Enter to skip): Min Phyo

--- Editing User ---
Enter New First Name (current: Aung): Aung
Enter New Last Name (current: Min Phyo): Phyo
Enter New Date of Birth (current: 10/5/2005): 10/4/2004
Enter New Phone Number (current: 0978432345): 09234576
Enter New Address (current: MYN): MYN
Enter New Password (current: Test@456): Test@456
```

```
6. Exit
Choose an option (1-6): 5
Enter First Name to find for deletion: Lwin
Enter Last Name to find for deletion (optional, press Enter to skip): Min Khant
User deleted successfully.

1. Create New User
2. View All Users
3. Find User by Name
4. Edit User
5. Delete User
6. Exit
Choose an option (1-6):
```

In the code, the program is successfully running which meets all the requirements. Users can successfully create edit or delete the user's account and the password validation also worked perfectly, to make a secure user account and password.

Improvements

Recently, the program used an array to store user data in memory. In the future, a database or file system could be built to make the program strong and consistent.

Extra error handling could be added to improve the user experience, such as improving invalid inputs in a better way.

10.CONCLUSION

In conclusion, the project for the New User Account Registration System successfully displayed the abilities of computer science such as user input validation and data management. To create strong and safe passwords, the menu allows the users to have multiple choices when creating an account. For the functions, the user can create, view, edit, and delete accounts. Then the future improvements can include implementing data storage and error handling to improve user experience. Finally, this project has improved my understanding of Python programming and practical application development.

11.BIBLIOGRAPHY

- Python Software Foundation, 2023. *Python Documentation*. Available at: https://docs.python.org/3/ [Accessed 7 October 2024].
- W3Schools, 2024. *Python Functions*. Available at: https://www.w3schools.com/python/python_functions.asp [Accessed 30 September 2024].

- TutorialsPoint, 2024. *User Input in Python*. Available at: https://www.tutorialspoint.com/python/python_user_input.htm [Accessed 1 October 2024].