COMP1811 – Python Project Report

Name	Yash Sanjaykumar Patel	Student ID	001249232-9

1. BRIEF STATEMENT OF FEATURES YOU HAVE COMPLETED

1.1 Circle the parts of the coursework you have **fully completed and are fully working**. Please be accurate.	**Features** **F1:** i ☒ ii ☒ **F2:** i ☒ ii ☒ **F3:** i ☒ ii ☒ iii ☒ iv ☒ v ☒
1.2 Circle the parts of the coursework you have **partly completed or are partly working.**	**Features** **F1:** i ☐ ii ☐ **F2:** i ☐ ii ☐ **F3:** i ☐ ii ☐ iii ☐ iv ☐ v ☐
Briefly explain your answer if you circled any parts in 1.2	

2. Concise List of Bugs and Weaknesses

*A concise list of bugs and/or weaknesses in your work (if you don't think there are any, then say so). Bugs that are declared in this list will lose you fewer marks than ones that you don't declare! (**100-200 word**, but word count depends heavily on the number of bugs and weaknesses identified.)*

2.1 Bugs

List each bug plus a brief description

2.2 Weaknesses

List each weakness plus a brief description

3. DESCRIPTION OF THE FEATURES IMPLEMENTED

Describe your implementation of the required features and how well do they work. Provide some exposition of the design decisions made and indicate how the features developed were integrated.

The code implements the following features:

- Reading a social network data file, which has a data of member and its friends.
- Display the social network data.
- Recommend a friend for a user based on mutual friends.
- Display the number of friends a user has.
- Display the users who have the least number of friends or have 0 friends.
- Display the friends of the friends of a given user.

The code which reads the social network data file and stores the information in a dictionary "users", where each key is a user ID and the value is a list of the user's friends. If the file does not exist, the code prints "File not found!!". The code then prompts the user to display the social network data, recommend a friend for a user, display the number of friends a user has, display the users with the least number of friends or 0 friends, and display the friends of the friends of a user.

The recommendation feature works by looping through the friends of a user and checking if their friends are not already friends with the user. If no mutual friends are found, the code also checks for a second-degree connection (friends of friends). If a recommended friend is found, it is displayed, otherwise, the code outputs "There is no recommended friend."

The code to display the number of friends a user has works by using the length of the list of friends stored in the "users" dictionary.

The code to display the users with the least number of friends or 0 friends loops through all the users, checks the length of their list of friends, and appends the user IDs to a list "atleast_friend" if they have at least one friend or to a list "no_friend" if they have 0 friends. The resulting lists are then displayed.

The code to display the friends of the friends of a user works by looping through all the users, checking if they are friends with the input user, and finding the mutual friends between the user and their friend. If any mutual friends are found, they are displayed, otherwise, "None" is displayed.

All of the features were integrated by using the input function to prompt the user for choices and if statements to handle the user's choices.

4. CLASSES AND OOP FEATURES

*List all the classes used in your program and include the attributes and behaviours for each. You may use a class diagram to illustrate these classes. Your narrative for section 3.2 should describe the design decisions you made and the OOP techniques used. List the classes you developed and provide an exposition on the choice of classes, class design and OOP features implemented. (**200-400 words**).*

4.1 CLASSES USED

There are two I have used in this social network application.

1. **User**
2. **Social Network**

4.2 BRIEF EXPLANATION OF CLASS DESIGN AND OOP FEATURES USED

- In the user class I have use the constructor(_ _init_ _) function to pass the value of members' id and their name and the second function is doing the looping to correct value of user. The second class which is the social network in that i passed so many function to get the network implementation features.

5. TESTING

Describe the process you took to test your code and to make sure the program functions as required. Provide the detailed test plan used.

Here is a sample detailed test plan that can be used for the code:

1. Test for successful file read: Input: "sample_file.txt" Expected Output: Display the social network (y/n)? (enter y) File contents displayed
2. Test for invalid file name: Input: "invalid_file.txt" Expected Output: File not found!!
3. Test for recommended friend functionality: Input: Enter a user ID as an integer from 1 to 3: 1 Do you want to recommend friends to another user (y/n)? (enter y) Enter a user ID as an integer from 1 to 3: 2 Expected Output: The recommended friend for 2 is 3
4. Test for displaying the number of friends of a user: Input: Display how many friends a user has (y/n)? (enter y) Enter a user ID as an integer from 1 to 3: 1 Expected Output: User 1 has 2 friends
5. Test for displaying the users with the least number of or have 0 friends: Input: Display the users with the least number of or have 0 friends (y/n)? (enter y) Expected Output: The user ID for the user with least friends is: 3 The user ID for the user with 0 friends is:
6. Test for displaying the friends of the friends of a given user: Input: Display the friends of the friends of a given user (y/n)? (enter y) Enter a user ID as an integer from 1 to 3: 1 Expected Output: The friends of the friends of user 1 are: 3
7. Test for user not found case: Input: Enter a user ID as an integer from 1 to 3: 4 Expected Output: None (As user ID 4 is not found in the input file)

6. Annotated Screenshots Demonstrating Implementation

*Provide screenshots that demonstrate the features implemented. Annotate each screenshot and if necessary, provide a brief description for **each** (**up to 100 words**) to explain the code in action. Make sure the screenshots make clear what you have implemented and achieved.*

```
import os
temp = {}
class User:
    def __init__(self, id, name):
        self.id=id
        self.name=name
    def in_list(self, lst: list):
        for i in lst:
            if i.id == self.id:
                return True
        return False
```

This code imports the 'os' module, which is used for interacting with the operating system to check the directory. It then creates an empty dictionary called 'temp' which can be used to store values. It then creates a class named 'User'. This class takes two parameters, 'id' and 'name', which is used to create instances of the class. It then defines a method called 'in_list' which takes a list as a parameter. The method iterates through the list and checks if any of the elements in the list have an id which matches the id of the current instance. If it does, it returns True, otherwise, it returns False.

```
class SocialNetwork:
    def __init__(self):
        self.users: dict[User, list[User]] = {}
        self.name_mode = None

    def add_user_from_line(self, line):
        data = line.split()
        user_name = data[0].strip()
        if self.name_mode == None:
            if user_name.isdigit():
                self.name_mode = True
            else:
                self.name_mode = False
```

Now I have created a SocialNetwork class. It has an __init__ method which takes no parameters and sets two properties: "users" which is an empty dictionary, and "name_mode" which is set to None. The class also has a method "add_user_from_line" which takes a line as a parameter. This method splits the line into its separate parts, then checks to see if the first part is an integer or a string. If it is an integer, it sets the "name_mode" property to True, otherwise it sets it to False.

```
if user_name not in temp.keys():
    temp[user_name] = user_name if user_name.isdigit() else len(temp.keys())
user_id = temp[user_name]
user = User(user_id, user_name)
if len(data)!=2:
    self.users[user] = []
    return
friend_name = data[1].strip()
if friend_name not in temp.keys():
    temp[friend_name] = friend_name if friend_name.isdigit() else len(temp.keys())
```

```
friend_id = temp[friend_name]
friend = User(friend_id, friend_name)
for u in self.users.keys():
    if u.id == user.id:
        self.users[u].append(friend)
        break
else:
    self.users[user] = [friend]
user, friend = friend, user
for u in self.users.keys():
    if u.id == user.id:
        self.users[u].append(friend)
        break
```

This code is checking if the user name is included in a temporary set of keys and assigns them a user id if it is not. It then creates a new user object with the user's id and name. If the data does not have two elements, it creates an empty list for the user and returns. If it does have two elements, it takes the second one (the friend's name) and checks if it is included in the temporary set of keys. If it is not, it assigns it a friend id. It then creates a new user object with the friend's id and name. It then checks the list of users in the self.users dictionary to see if the user's id is included. If it is, it appends the friend user object to the list of users associated with the user. If the user isn't included in the list of users, creates a new entry in the dictionary with the user as the key and a list containing the friend as the value. Finally, it replaced the user and friend, and checks the list of users in the self , users dictionary again in the same manner as before.

```
else:
        self.users[user] = [friend]

def display(self):
    for u, f in self.users.items():
        print(f"{u.name} -> {', '.join(map(lambda x: x.name, f))}")

def get_user_from_id(self, id: str) -> User:
    for u, f in self.users.items():
        if u.id == id:
            return u
    return None
```

This code is part of a class that is likely used to manage user relationships in an application. The first part of the code is an else statement which states that if the user does not already exist in the self.users dictionary, the user should be added to the dictionary with the given friend. The display function prints out the user's name followed by the names of their friends, separated by commas. The get_user_from_id function takes in an id string and searches through the self.users dictionary to find a User object that has a matching id. If a matching User object is found, it is returned, otherwise the function returns None.

```
def get_user_from_input(self):
    text = "Enter user name: " if not self.name_mode else f"Enter user id as an
integer from 0 to {len(self.users.keys())-1}: "
    ans = input(text)
    for u in self.users.keys():
        if not self.name_mode and u.name.lower() == ans.lower():
            return u
        if u.id == ans:
            return u
```

```
    print("User doesn't exist")
    return self.get_user_from_input()
```

In the ‘get_user_from_input’ function the first line of code sets a variable called 'text' which contains a string. If the 'name_mode' parameter is not set, then the 'text' will contain a message asking the user to enter their user name. If the 'name_mode' parameter is set, then the 'text' will contain a message asking the user to input the user id as an integer. The second line of code creates a variable called 'ans' which stores the user input. The third line of code is a for loop which iterates through all the keys of the 'users' dictionary. The fourth line of code checks if the 'name_mode' parameter is not set and if the user name stored in 'ans' is the same as the user name stored in the 'users' dictionary. If the two conditions are met, then the user is returned. The fifth line of code checks if the user id stored in 'ans' is the same as the user id stored in the 'users' dictionary. If the two conditions are met, then the user is returned. The sixth line of code prints a message if the user doesn’t.

```
if input("Display how many friends a user has (y/n)?").lower() in ("y","yes"):
    user = social_nw.get_user_from_input()
    existing_friend = social_nw.users(social_nw.get_user_from_id(user.id))
    if existing_friend:
        print(f"User {user.name} has {len(existing_friend)} friends")
if input("Display the users with the least number of or have 0 friends
(y/n)?").lower() in ("y","yes"):
    atleast_friend = []
    no_friend=  []
    for user, friend in social_nw.users.items():
        if len(friend) == 1 and not user.in_list(atleast_friend):
            atleast_friend.append(user)
        elif not friend and not user.in_list(no_friend):
            no_friend.append(user)
        for i in friend:
            usr_data = social_nw.users[social_nw.get_user_from_id(i.id)]
            if not usr_data and not i.in_list(no_friend):
```

This code checks if the user wants to display the friends of the friends of a given user. If the user inputs "y" or "yes", the code will prompt the user to enter a user ID. Then it retrieves the friends of the given user and for each friend, it retrieves their friends as well. The code then displays the list of the friends of the friends of the given user. This information is presented to the user in the form of a list.

```
if atleast_friend:
            print(f"The user ID for the user with least friends is: {',
'.join(map(lambda x: x.name, atleast_friend))}")
        if no_friend:
            print(f"The user ID for the user with 0 friends is: {',
'.join(map(lambda x: x.name, no_friend))}")
    if input("Display the friends of the friends of a given user (y/n)?").lower()
in ("y","yes"):
        user = social_nw.get_user_from_input()
        friends = social_nw.users[user]
        for u, friend in social_nw.users.items():
            if u.id != user.id and u.in_list(friends):
                common_list = set(friends).intersection(friend)
                mutual_frd = "None"
                if common_list:
                    mutual_frd = ",".join(map(lambda x: x.name, common_list))
                print("%s -> %s"%(u.name,mutual_frd))
    if input("Do you want to try another social network (y/n)?").lower() in
("y","yes"):
```

```
        return main()
main()
```

7. EVALUATION

*Give a reflective, critical self-evaluation of your experience developing the project and discuss what you would do if you had more time to work on the project. Answer the following questions for the reflection and write **350-400 words overall**. Please include an actual word count for this section.*

8.1 EVALUATE HOW WELL YOUR DESIGN AND IMPLEMENTATION MEET THE REQUIREMENTS

valuation of Design and Implementation against Requirements: The code was written to fulfill the requirements of reading in a social network from a file, recommending a friend, and displaying friend count and users with least friends. The code successfully reads in the file and creates a dictionary of users and their friends. The code also provides a friend recommendation by looking at the friends of the friends of a user and checking if they are not already friends. The code also displays the count of friends a user has and the users with the least number of friends.

8.2 EVALUATE YOU OWN PERFORMANCE

8.2.1 WHAT WENT WELL?

The code successfully reads in the social network from a file.

The code successfully provides friend recommendations and displays friend count.
The code successfully displays users with the least number of friends.

8.2.2 WHAT WENT LESS WELL?

The code only recommends one friend, and the recommendations could be improved by providing multiple recommendations.

The code could use some refactoring to make it more readable and efficient.

8.2.3 WHAT WAS LEARNT?

The importance of checking for edge cases when reading in data from a file.

The importance of refactoring code for readability and efficiency.

8.2.4 HOW WOULD A SIMILAR TASK BE COMPLETED DIFFERENTLY?

A similar task would be completed differently by using a graph data structure to represent the social network, which would make it easier to traverse the network and make recommendations.

8.2.5 HOW COULD THE MODULE BE IMPROVED?

The code could be improved by providing multiple friend recommendations.

The code could be refactored to use a graph data structure to represent the social network.

The code could be improved by adding error handling for cases where the file does not exist or the user does not input a valid user ID.

8.3 Self-assessment

Please assess yourself objectively for each section shown below and then enter the total mark you expect to get. Marks for each assessment criteria are indicated between parentheses.

Code development (70)

Features Implemented [30]

Feature 1 (up to 6)

Sub-features have not been implemented – 0
Attempted, not complete or very buggy – 1 or 2
Implemented and functioning without errors but not integrated – 3
Implemented and fully integrated but buggy – 4
Implemented, fully integrated and functioning without errors – 5 or 6

Feature 2 (up to 12)

Sub-features have not been implemented – 0
Attempted, not complete or very buggy – 1 or 3
Implemented and functioning without errors but not integrated – 4 to 6
Implemented and fully integrated but buggy – 7 to 9
Implemented, fully integrated and functioning without errors – 10 to 12

Feature 3 (up to 12)

Sub-features have not been implemented – 0
Attempted, not complete or very buggy – 1 to 3
Implemented and functioning without errors but not integrated – 4 to 6
Implemented and fully integrated but buggy – 7 to 9
Implemented, fully integrated and functioning without errors – 10 to 12

For this criterion I think I got: 20 out of 30

Use of OOP techniques [25]

Abstraction (up to 9)

No classes have been created – 0
Classes have been created superficially and not instantiated or used – 1 or 2
Classes have been created but only some have been instantiated and used – 3 or 4
Useful classes and objects have been created and used correctly – 5 to 7
The use of classes and objects exceeds the specification – 8 or 9

Encapsulation (up to 9)

No encapsulation has been used – 0
Class variables and methods have been encapsulated superficially – 1 to 3
Class variables and methods have been encapsulated correctly – 4 to 6
The use of encapsulation exceeds the specification – 7 to 9

Inheritance (up to 7)

No inheritance has been used – 0
Classes have been inherited superficially – 1 or 2
Classes have been inherited correctly – 3 to 5
The use of inheritance exceeds the specification – 6 or 7

Bonus marks will be awarded for the appropriate use of polymorphism (bonus marks up to 5)

For this criterion I think I got: 21 out of 25

Quality of Code [15]

Code Duplication (up to 8)

Code contains too many unnecessary code repetition – 0
Regular occurrences of duplicate code – 1 to 3
Occasional duplicate code – 4 to 5
Very little duplicate code – 6 to 7
No duplicate code – 8

PEP8 Conventions and naming of variables, methods and classes (up to 4)

PEP8 and naming convention has not been used – 0
PEP8 and naming convention has been used occasionally – 1
PEP8 and naming convention has been used, but not regularly – 2
PEP8 and naming convention has been used regularly – 3
PEP8 convention used professionally and all items have been named correctly – 4

In-code Comments (up to 3)

No in-code comments – 0
Code contains occasional in-code comments – 1
Code contains useful and regular in-code comments – 2
Thoroughly commented, good use of docstrings, and header comments describing.py files – 3

For this criterion I think I got: 10 out of 15

DOCUMENTATION (20)

Design (up to 10) clear exposition about the design and decisions for OOP use

The documentation cannot be understood on first reading or mostly incomplete – 0
The documentation is readable, but a section(s) are missing – 1 to 3
The documentation is complete – 4 to 6
The documentation is complete and of a high standard – 7 to 10

Testing (5)

Testing has not been demonstrated in the documentation – 0
Little white box testing has been documented – 1 or 2
White box testing has been documented for all the coursework – 3 or 4
White box testing has been documented for the whole system – 5

Evaluation (5)

No evaluation was shown in the documentation – 0
The evaluation shows a lack of thought – 1 or 2
The evaluation shows thought – 3 or 4
The evaluation shows clear introspection, demonstrates increased awareness – 5

For this criterion I think I got: 17 out of 20

ACCEPTANCE TESTS - DEMONSTRATIONS (10)

Final Demo (up to 10)

Not attended or no work demonstrated – 0
Work demonstrated was not up to the standard expected – 1 to 3
Work demonstrated was up to the standard expected – 4 to 7

Work demonstrated exceeded the standard expected – 8 to 10

For this criterion I think I got: 7 out of 10

I think my overall mark would be: 75 out of 100

APPENDIX A: CODE LISTING

*Provide a complete listing of all the *.py files in your PyCharm project. Make sure your code is well commented and applies professional Python convention (refer to PEP 8 for details). The code listed here must match that uploaded to Moodle. Please copy and paste the actual code – no screenshots please! You will lose marks if screenshots are provided instead of code.*