Delhi Technological University

Team Making Utility

Aryan(2K19/SE/019)

https://github.com/aryan0154762/Team-Making-Utility

Title

The application **Team Making Utility**, as the name suggests helps an organization to make Teams for many different tasks that require different skill sets, in addition to selecting individuals based on their skills it also tries to select the members having good interpersonal relations with each other and thus will also prevent any delay caused due to incompatibility of the team members.

This utility is a Modified implementation of the concept research paper mentioned in the end of this document, The concept taught by sir Manu Narula greatly helped me in making of this project, I thoroughly referred to the internet and am greatful for the knowledge gained while reading and implementing those resources, all the places referred in making of the project is linked at the end of this document.

Table of contents

Abstract

The application attempts to solve the problem of making groups of people having required skills set and good interpersonal relations with other members of the group to achieve the intended goal harmoniously.

The program aimed to accomplish the following :
1. Store the skills of each member/employee with their details.
2. Maintain their relations(friends or discordants) with other members
3. Make teams for a specific skill set while considering the interrelations of all the team members.
4. All the CRUD(Create, Read, Update, Delete) operations must be supported.

I used the build and fix methodology as the size of our project is small and relatively less complex than most of the enterprise software we repeatedly tested and debugged all of our code every time we added any new functionality.

Problem Statement

This program is the solution of the following problem statement :

It is not uncommon to see many projects get cancelled or delayed during the phase of development just because the team making the project is not able to work with each other harmoniously and productively, this might be because some people are incompatible with some people, and thus even if they have the required skill set their interpersonal relationships with other team members often end up being the problem,
To avoid making a team of members having unhealthy interpersonal relations with one another, the program tries to suggest groups of people having the required skill set and good interpersonal relations with other members to work productively.

I was able to solve this problem by using the concept of graph data structure.
The program uses graph data structure at its core with various features that may be useful to an organization(like dual mode). Dual mode, one for the ‘Admin’ and the other for the ‘user’ helps satisfy the requirement of both(described further in the text).

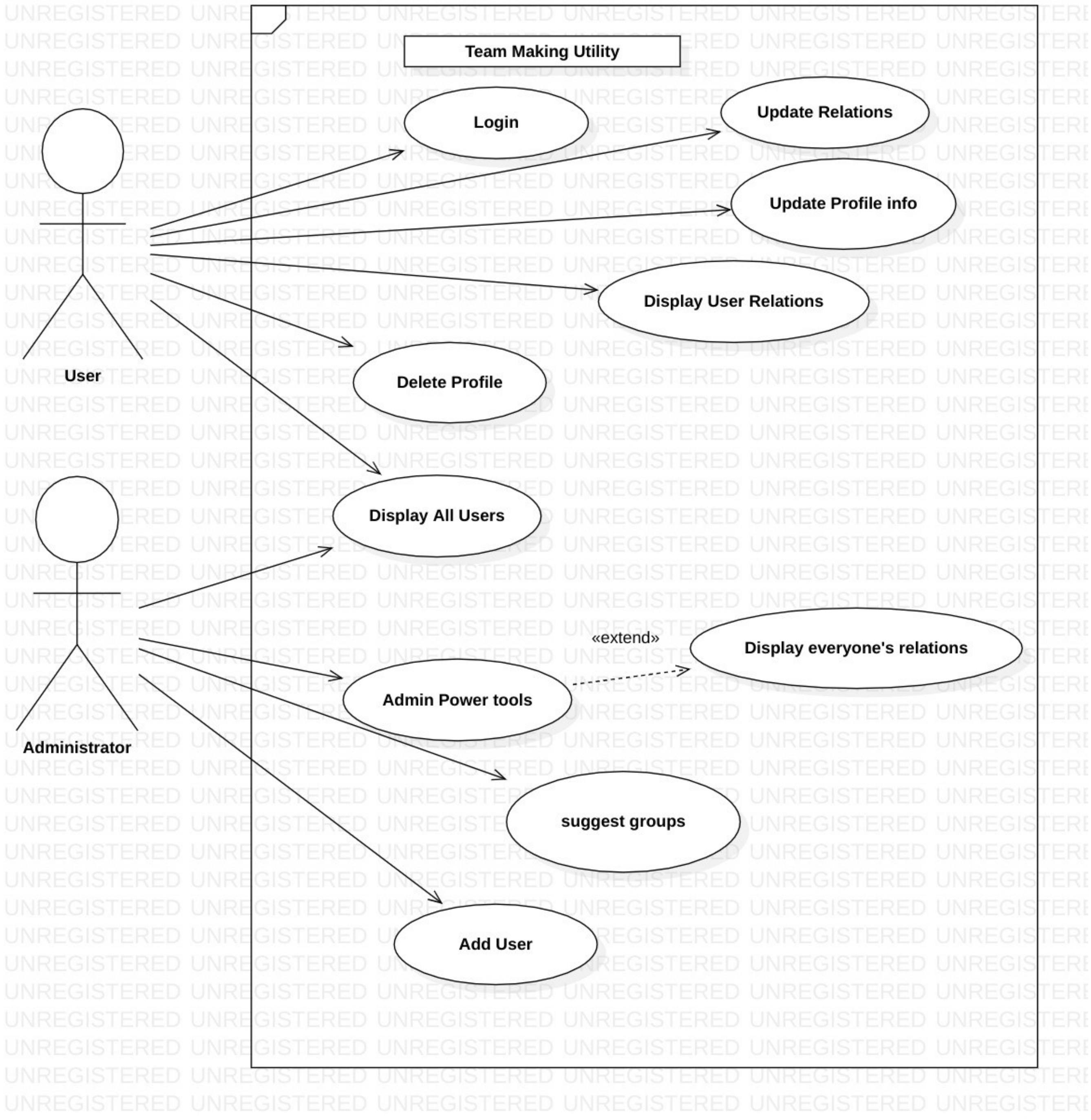
Introduction

This project is a result of a problem that has been bothering employees and members of many groups, i.e peaceful, productive working environment and non-cooperative colleagues. It can be a major waste of time if the people you are working with are not cooperative and do not work as hard as you do and consistently bother you.

I thought that the cause of all these problems is that many a times the authority making these groups is itself unaware of all that goes into achieving a milestone and thus are equally unaware of the manforce required for that task which often leads to delay and civil conflicts within the group if the members are not in harmony with each other.
To minimize this, we need to make teams in such a way that the members of the team are understanding caring and prioritize their wellbeing and quality delivery rather than rushing the expected milestone, this is beneficial for both the consumer and the authority making it as the quality of the product delivered will be leaps ahead of what would have been delivered otherwise.

Diagrams and Flowcharts

Use case diagram :



Use case description of the primary use case :

Introduction : This use case description documents the procedure of suggesting optimal groups to the admin give a set of members using their skill set and relations with one another.

Actors : Admin

Pre-condition : The admin Password must be entered correctly

Post-condition : Show the time taken to perform the operations

Event flow :

basic flow :

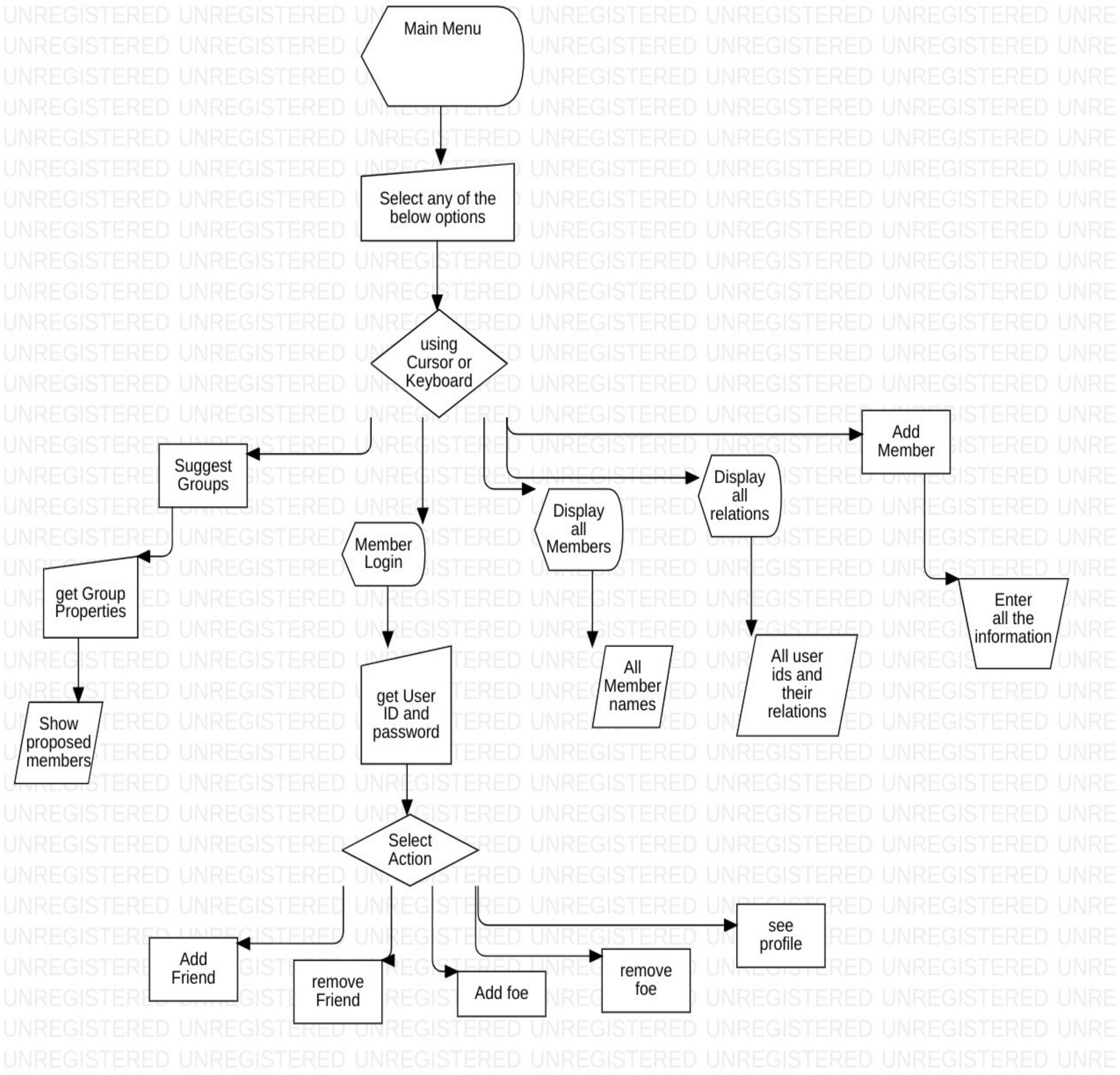
1. The admin selects the suggest groups option.
2. The admin lists the required skill set
3. The admin lists the strength(no. of members) of the group.
4. The program suggests the group members by listing their name and user IDs .

Alternate flow : The required number of people in the group are more than or close to the total number of members in the organization :
In such a case generate a warning that the number of people in the group are close to the total number of people in the org and thus the groups suggested by the programs is indifferent from any other suggestion.

Special Requirements : The admin must login in admin mode

Associated Use Case : Admin tools

Flowchart :



Main Menu
Select any of the below options
using Cursor or Keyboard
Suggest Groups
get Group Properties
Show proposed members
Member Login
get User ID and password
Select Action
Add Friend
remove Friend
Add foe
remove foe
see profile
Display all Members
All Member names
Display all relations
All user ids and their relations
Add Member
Enter all the information

Work Done

Modules used :

A total of **7** modules are there, these are :

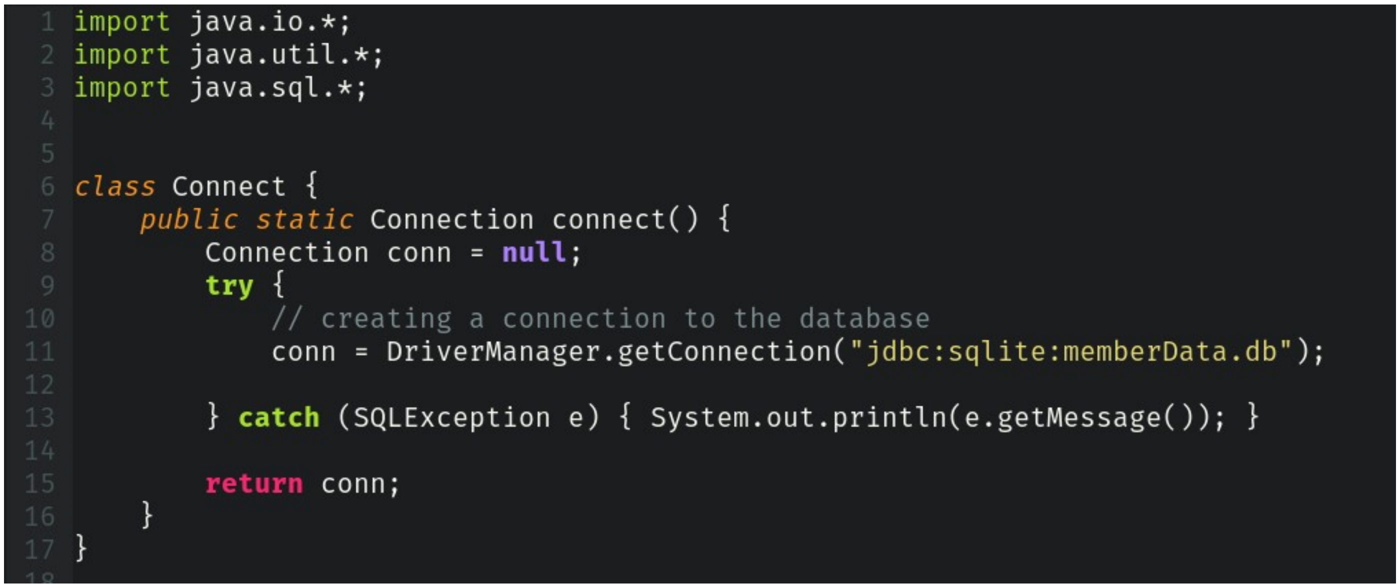
- **Connect.class**
- **DatabaseInterface.class**
- **MemberInterface.class**
- **MemoryInterface.class**
- **RelationsGraph.class**
- **Skill.class**
- **main.sh**

The usage and working of each of these class is as follows :

1. Connect class :

This class is being used for helping in communicating with the SQLite3 database by connecting to it and providing the link to rest of the program, this is essential functionality and thus had to included separtely to adhere to the rules of encapsulation.

Below are some screenshots of the class code :



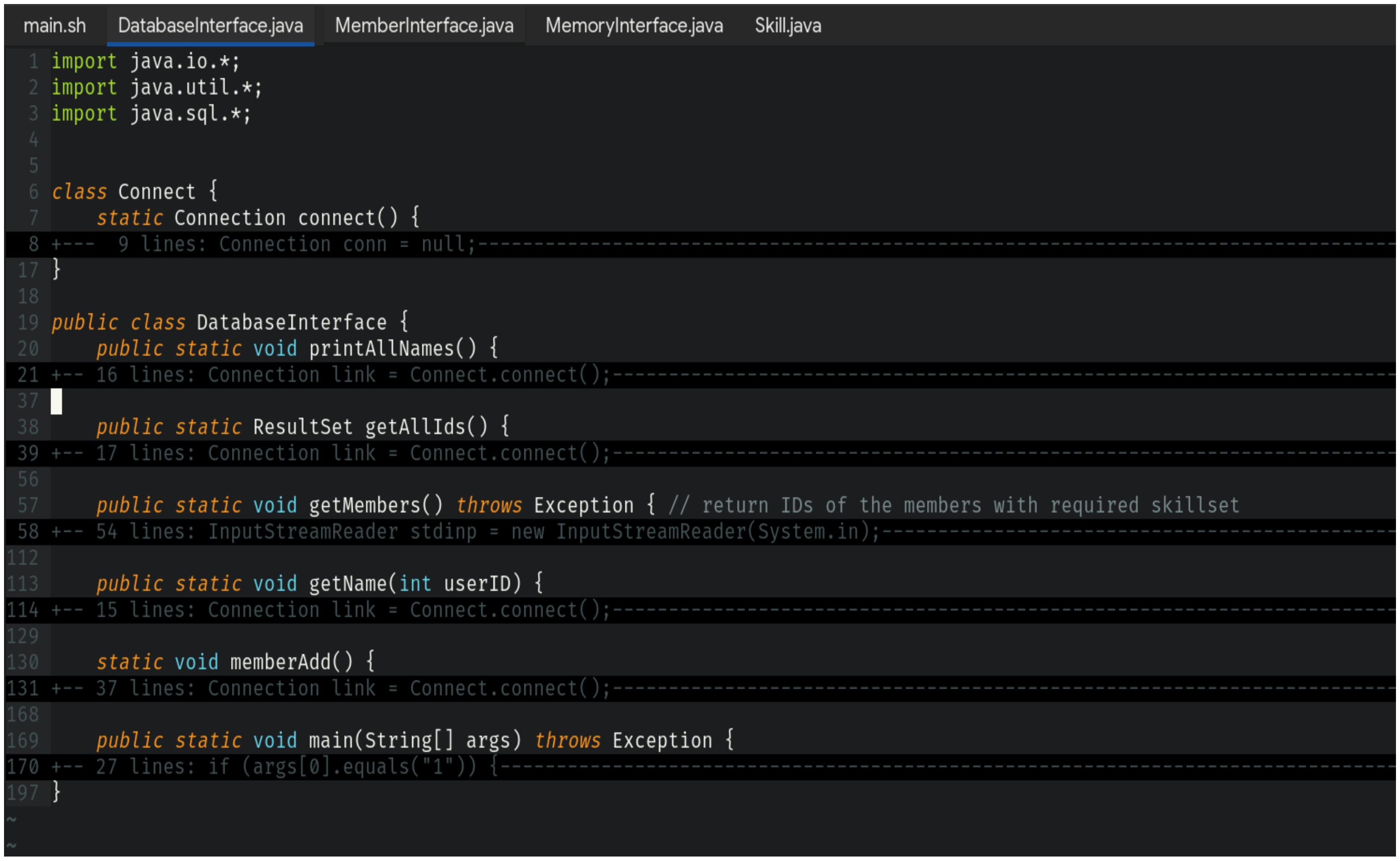
```java
import java.io.*;
import java.util.*;
import java.sql.*;


class Connect {
    public static Connection connect() {
        Connection conn = null;
        try {
            // creating a connection to the database
            conn = DriverManager.getConnection("jdbc:sqlite:memberData.db");

        } catch (SQLException e) { System.out.println(e.getMessage()); }

        return conn;
    }
}
```

2. DatabaseInterface class :

This class is also extremely important as it is responsible for providing other classes genuine data for usage to all the other classes in the file.
It also provides the following services :

- Print the names of all the members in the Team Making utility ordered in alphabetic order to list all the members.
- Get all the members with a specific skillset and use that information to make or suggest groups to the admin.
- Adding a new Member to the Team Making utility's SQLite3 Database.

The code snippet of the DatabaseInterface class can be seen below :



```
import java.io.*;
import java.util.*;
import java.sql.*;


class Connect {
    static Connection connect() {
+---  9 lines: Connection conn = null;-----------------------------------------------------------------------------
}

public class DatabaseInterface {
    public static void printAllNames() {
+-- 16 lines: Connection link = Connect.connect();----------------------------------------------------------------

    public static ResultSet getAllIds() {
+-- 17 lines: Connection link = Connect.connect();----------------------------------------------------------------

    public static void getMembers() throws Exception { // return IDs of the members with required skillset
+-- 54 lines: InputStreamReader stdinp = new InputStreamReader(System.in);-----------------------------------------

    public static void getName(int userID) {
+-- 15 lines: Connection link = Connect.connect();----------------------------------------------------------------

    static void memberAdd() {
+-- 37 lines: Connection link = Connect.connect();----------------------------------------------------------------

    public static void main(String[] args) throws Exception {
+-- 27 lines: if (args[0].equals("1")) {--------------------------------------------------------------------------
}
```

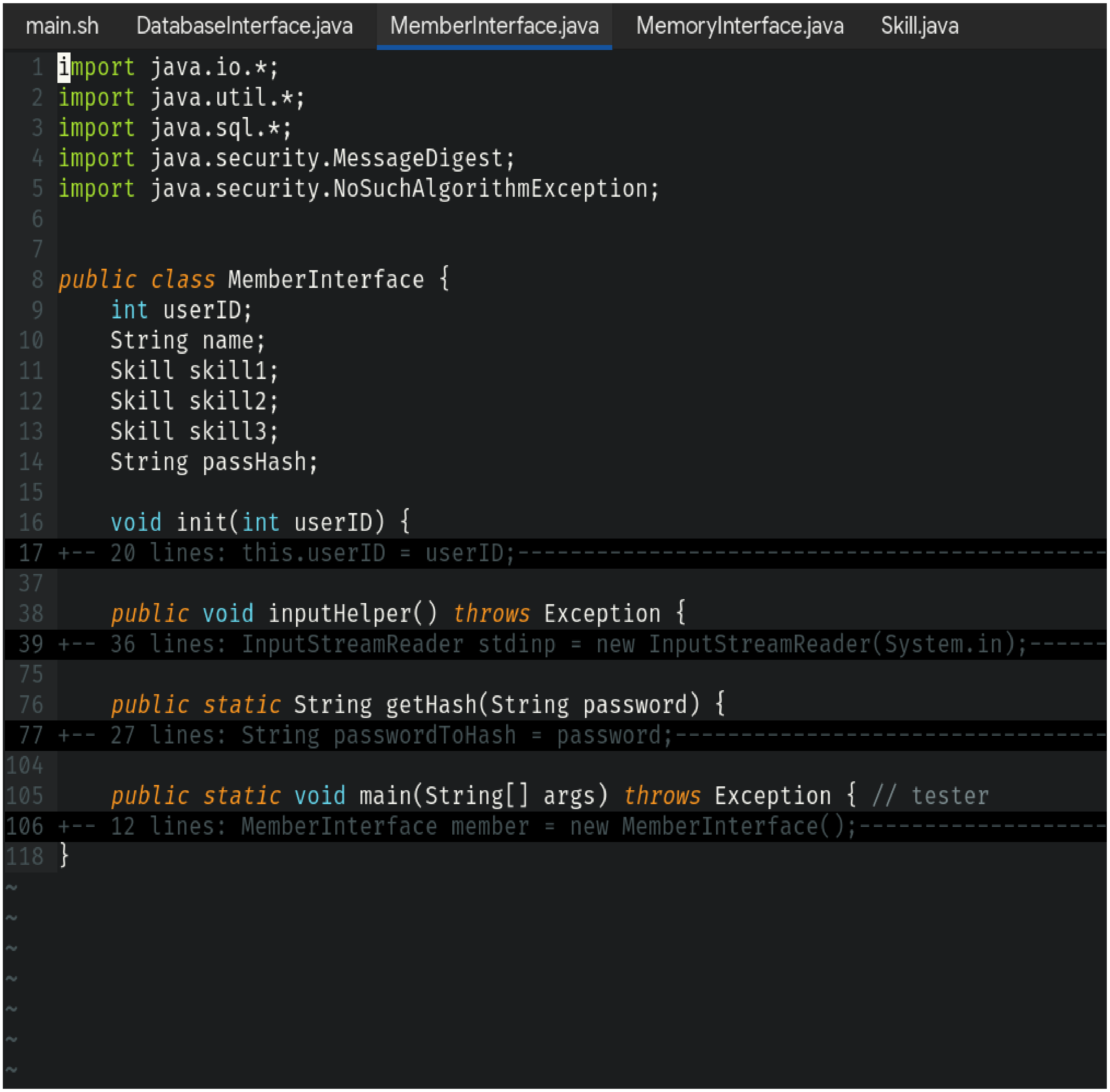
3. MemberInterface class :

This class is responsible for storing all the required information of a Member locally, it is mainly used for storing the user data in the program itself to avoid accessing the database again and again as that operation is costly.

It provides the following main services :

- Get basic member information of a particular member from the SQLite3 Database and save that information in the MemberInterface class object.
- Help in receiving,carring and transporting member data from one class to another class in a coherent way. It also accepts the information entered and processes that information and makes a member object out of it to encapsulate the information.
- The Function also hashes all the passwords so that they are not accessible or visible to anyone. Here I used the MD5 hashing algorithm using the java.security class members for hashing all the member passwords into the database for secure logging.

The code for the MemberInterface class is very large and thus whole code cannot be included, thus only some portion of the code is shown below :



main.sh DatabaseInterface.java MemberInterface.java MemoryInterface.java Skill.java

```
import java.io.*;
import java.util.*;
import java.sql.*;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;


public class MemberInterface {
    int userID;
    String name;
    Skill skill1;
    Skill skill2;
    Skill skill3;
    String passHash;

    void init(int userID) {
+-- 20 lines: this.userID = userID;------------------------------------------

    public void inputHelper() throws Exception {
+-- 36 lines: InputStreamReader stdinp = new InputStreamReader(System.in);------

    public static String getHash(String password) {
+-- 27 lines: String passwordToHash = password;---------------------------------

    public static void main(String[] args) throws Exception { // tester
+-- 12 lines: MemberInterface member = new MemberInterface();-------------------
}
~
~
~
~
~
~
~
```

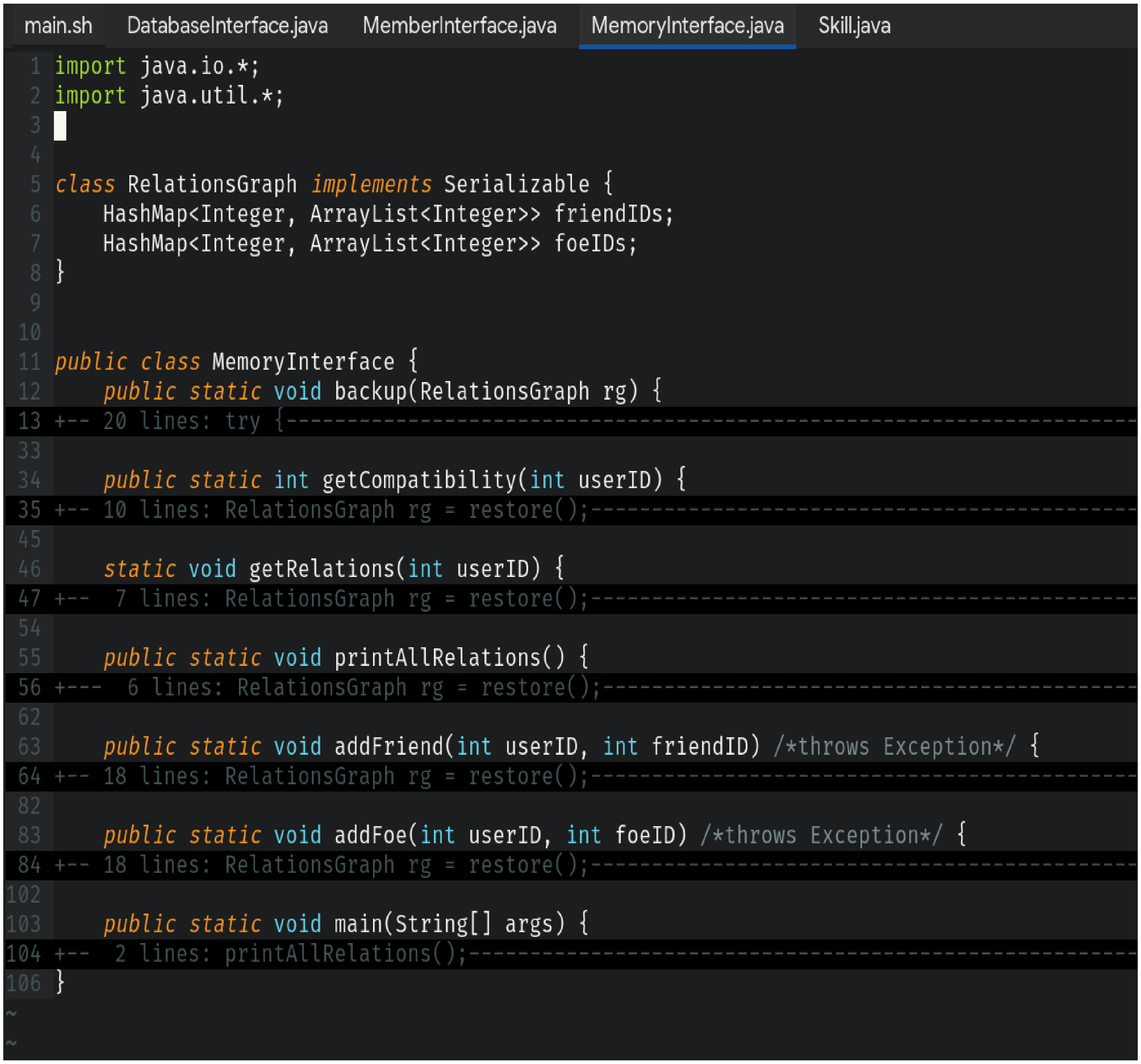
4. MemoryInterface/RelationsGraph class :

This class is essential for maintaining the friend and foe functionality and thus consequently the main functinality of the program i.e Suggesting groups based on the relationships of the members and their skillset, This class maintains all the relationship data in the program and as well as when the program is not running by serializing itself to the disk.

It Provides the following mean functinality :

- backup all the relationship information to the disk for future retrieval.
- Restore all the relationship information from the disk for usage.
- Edit the relationship information to edit the relations of one or many people.
- Manage all the relations atomically

The screenshots for the code are shown below :



```java
import java.io.*;
import java.util.*;


class RelationsGraph implements Serializable {
    HashMap<Integer, ArrayList<Integer>> friendIDs;
    HashMap<Integer, ArrayList<Integer>> foeIDs;
}


public class MemoryInterface {
    public static void backup(RelationsGraph rg) {
+-- 20 lines: try {--------------------------------------------------------------------

    public static int getCompatibility(int userID) {
+-- 10 lines: RelationsGraph rg = restore();-----------------------------------------------

    static void getRelations(int userID) {
+--  7 lines: RelationsGraph rg = restore();-----------------------------------------------

    public static void printAllRelations() {
+---  6 lines: RelationsGraph rg = restore();----------------------------------------------

    public static void addFriend(int userID, int friendID) /*throws Exception*/ {
+-- 18 lines: RelationsGraph rg = restore();-----------------------------------------------

    public static void addFoe(int userID, int foeID) /*throws Exception*/ {
+-- 18 lines: RelationsGraph rg = restore();-----------------------------------------------

    public static void main(String[] args) {
+--  2 lines: printAllRelations();---------------------------------------------------------
}
~
~
```

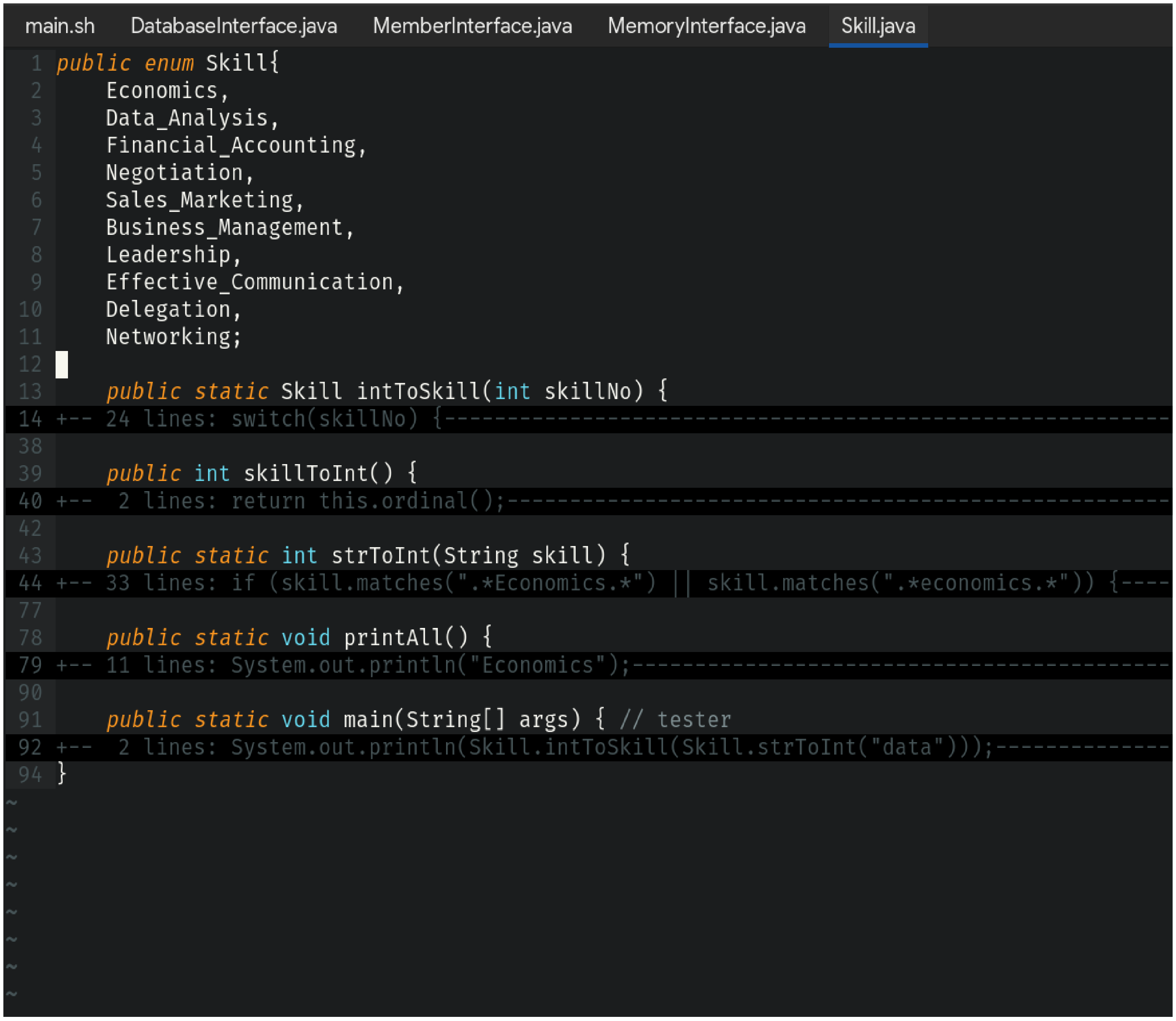
6. Skill class :

This class is an enum and stores all the skills currently supported by the Team Making Utility, This helps to make and encapsulate the skill related data in one entity and thus provide coherency.

We made use of the following skills :

Economics,
Data_Analysis,
Financial_Accounting,
Negotiation,
Sales_Marketing,
Business_Management,
Leadership,
Effective_Communication,
Delegation,
Networking.

In future, more skills can be added to the program simply by appending this list and a few more pieces of code in Skill.java file.

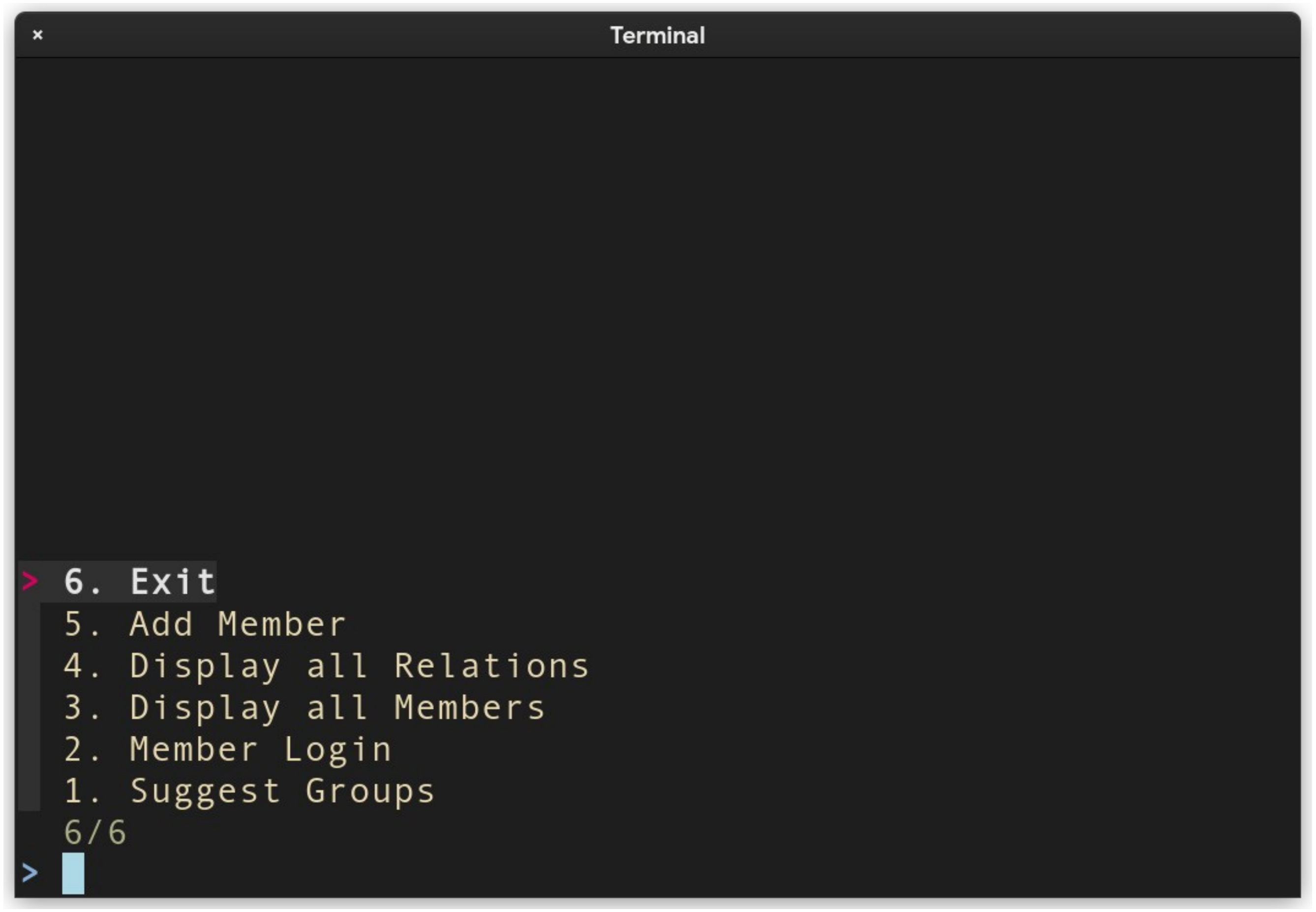

main.sh DatabaseInterface.java MemberInterface.java MemoryInterface.java Skill.java

```
public enum Skill{
    Economics,
    Data_Analysis,
    Financial_Accounting,
    Negotiation,
    Sales_Marketing,
    Business_Management,
    Leadership,
    Effective_Communication,
    Delegation,
    Networking;

    public static Skill intToSkill(int skillNo) {
+-- 24 lines: switch(skillNo) {-----------------------------------------------------

    public int skillToInt() {
+--  2 lines: return this.ordinal();--------------------------------------------------

    public static int strToInt(String skill) {
+-- 33 lines: if (skill.matches(".*Economics.*") || skill.matches(".*economics.*")) {----

    public static void printAll() {
+-- 11 lines: System.out.println("Economics");-----------------------------------------

    public static void main(String[] args) { // tester
+--  2 lines: System.out.println(Skill.intToSkill(Skill.strToInt("data")));-------------
}
~
~
~
~
~
~
~
~
```

7. Main.sh file :

This file is responsible for compiling the whole program and linking all the modules and providing a fuzzy finder based User Interface components to provide one good and useful TUI(terminal user interface)

Below is a screenshot of what it helps to achieve :



```
> 6. Exit
  5. Add Member
  4. Display all Relations
  3. Display all Members
  2. Member Login
  1. Suggest Groups
  6/6
>
```

The status of Proposed milestones :

1.	Account Control	: Complete
2.	Admin Tools	: Complete
3.	Admin Class	: Complete
4.	User Class	: Complete
5.	FileIO	: Complete
6.	Memory IO	: Complete
7.	Suggest groups	: Complete
8.	Initialization	: Complete
9.	Implement fuzzy menu	: Complete

The **User class and the admin class** contains all the data members and member functions directly associated with the Users and the admin respectively.
For e.g The user class has the functions, edit relations and display relations and all the relations data will be saved in the user class object, the information will be saved in memory as well as disk for faster operations(using RAM) and permanence(using disk).
The admin class contains the password(hard coded for security), last login and all the functions to satisfy the use cases included in Admin tools

The **memory I/O aids in dynamic allocation of memory** for storing the user data(Interpersonal relations of everyone in form of a graph) to get high bandwidth at the time or performing the I/O intensive use cases in admin tools.

The **suggest groups module** being the main highlight of our program is the most complicated module to right as it needs to be both fast and efficient for it to be able to work for sufficiently large sample sets. The current complexity of the algorithm is n^2 which is terrible considering how large the groups can get, though it should be possible to do it in nlogn time by sorting the data first but the trade offs for that are yet to be managed.

The file I/O is responsible for saving all the data of the member including there relations in a disk for permanent storage to retrieve that information whenever the program starts up again. It uses the serialization interface that can serialize any class and store its state to the disk.

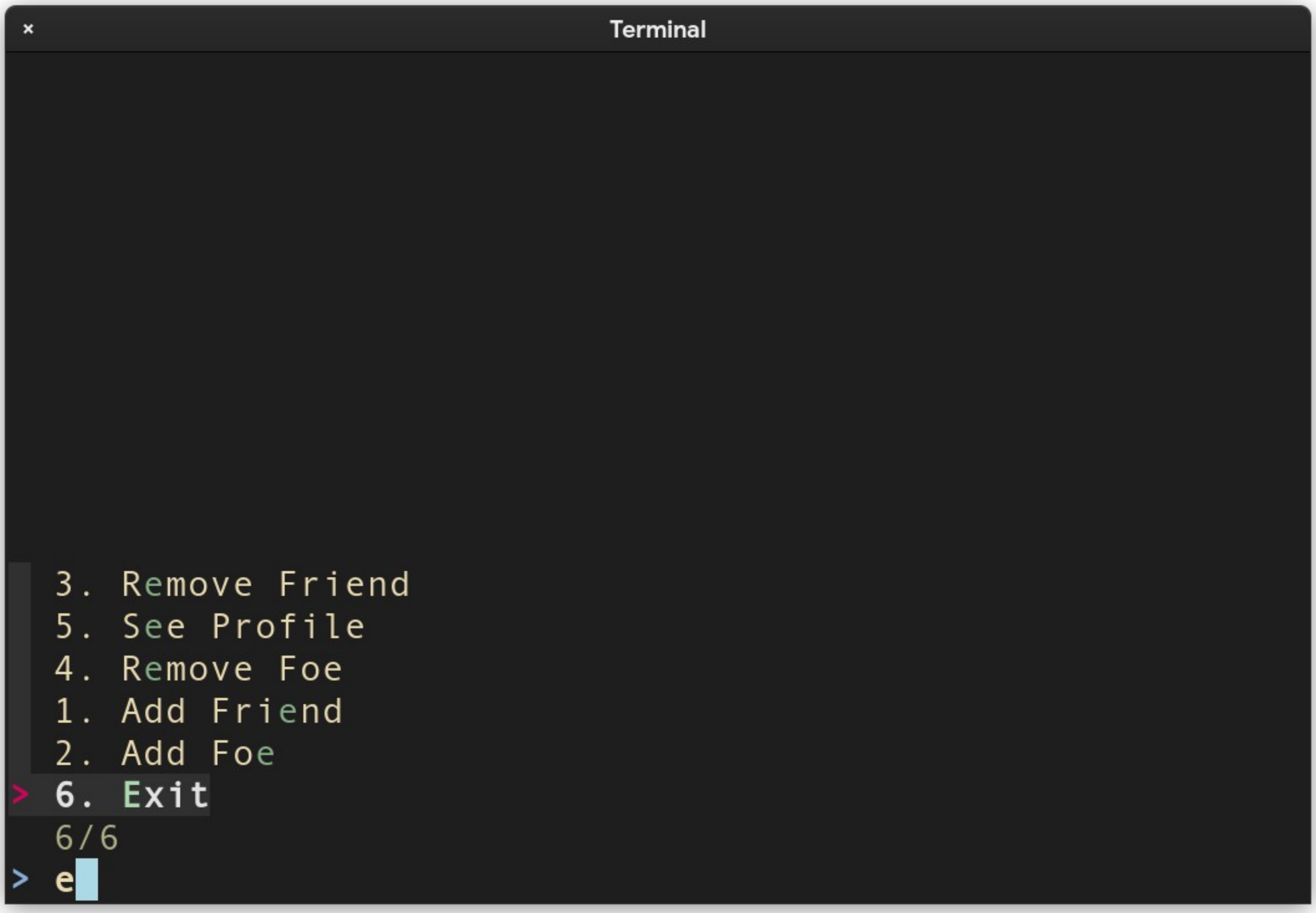
The account control is required for adding a new user and is an extension of admin tools use cases. This is important as the functionality of adding a user should be available to the members having the authorization to add a new member, this module may also be extended

The functions in **Admin tools are Algorithmically complex** and thus require more time. These functions are the main purpose of the program and help the admin to get the desired results quickly. e.g suggest groups, display all links, display all members etc.

Initialization will be used to initialize all the resources required by rest of the application to work. This is heavily reliant on the memory I/O module of the program. This will initialize all the classes with their data and the graph to perform the functions efficiently.

The fuzzy finder is a shell application which can generate good looking menus which are responsive and easier to use, it makes the user experience more friendly as the application deals with a lot of data and thus it can be over whelming to look at such a large volume of data at once thus, the fuzzy finder tool helps us select and see and search through the data effectively.

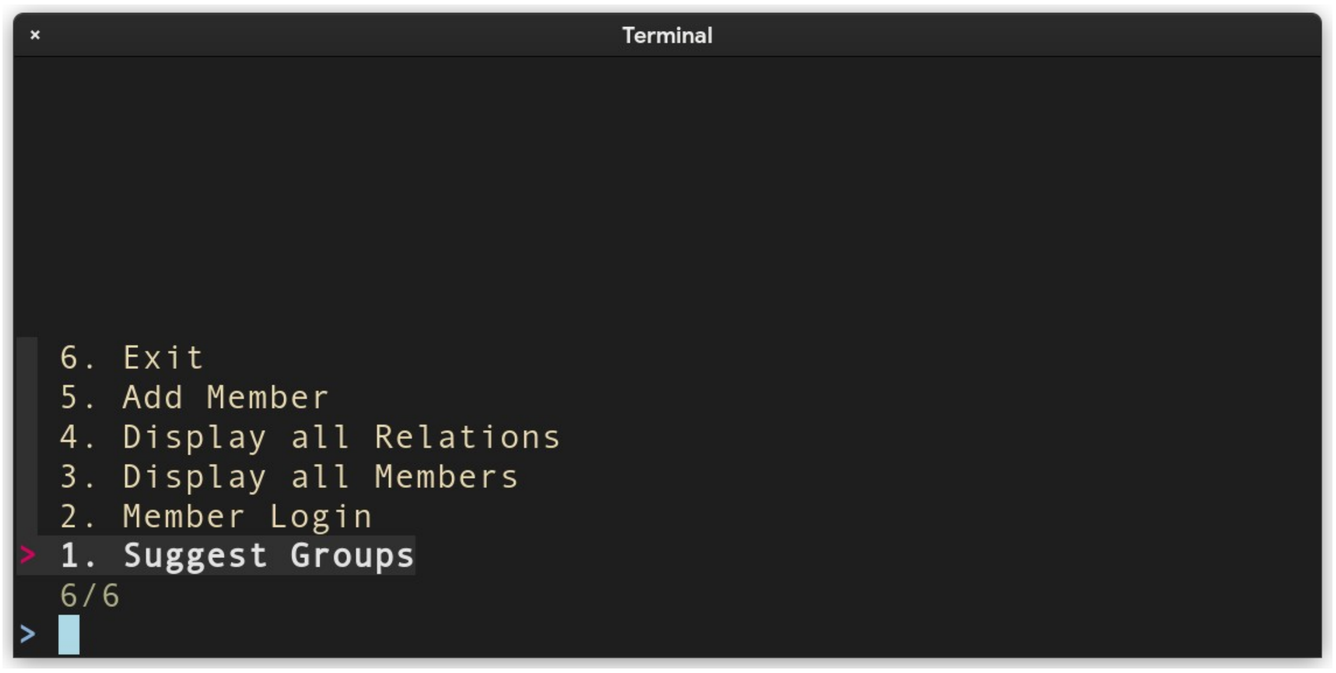
e.g. The image below shows the menu prompt by the fuzzy finder :
*this is a test database, the application is still in development phase and cannot produce such a functional output



```
  3. Remove Friend
  5. See Profile
  4. Remove Foe
  1. Add Friend
  2. Add Foe
> 6. Exit
  6/6
> e
```

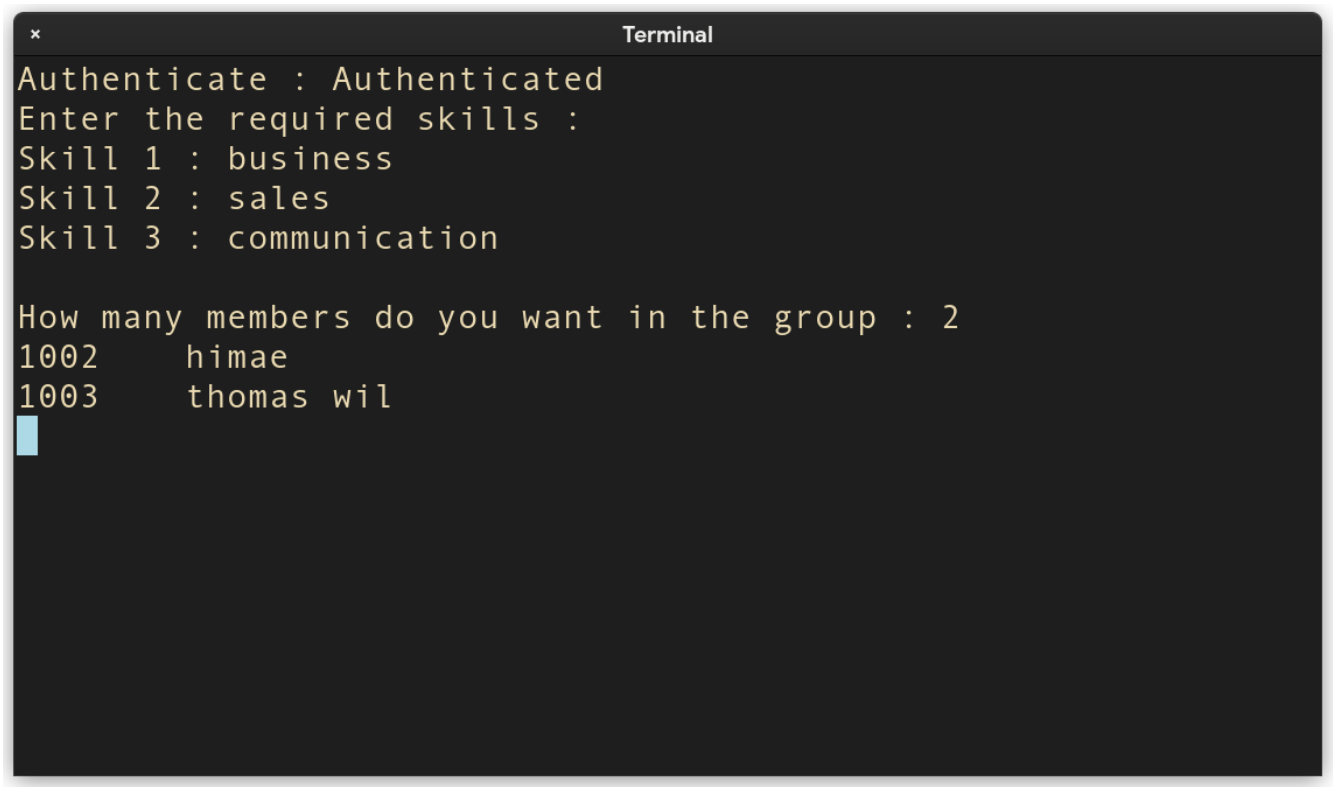
Screenshots

Main menu :



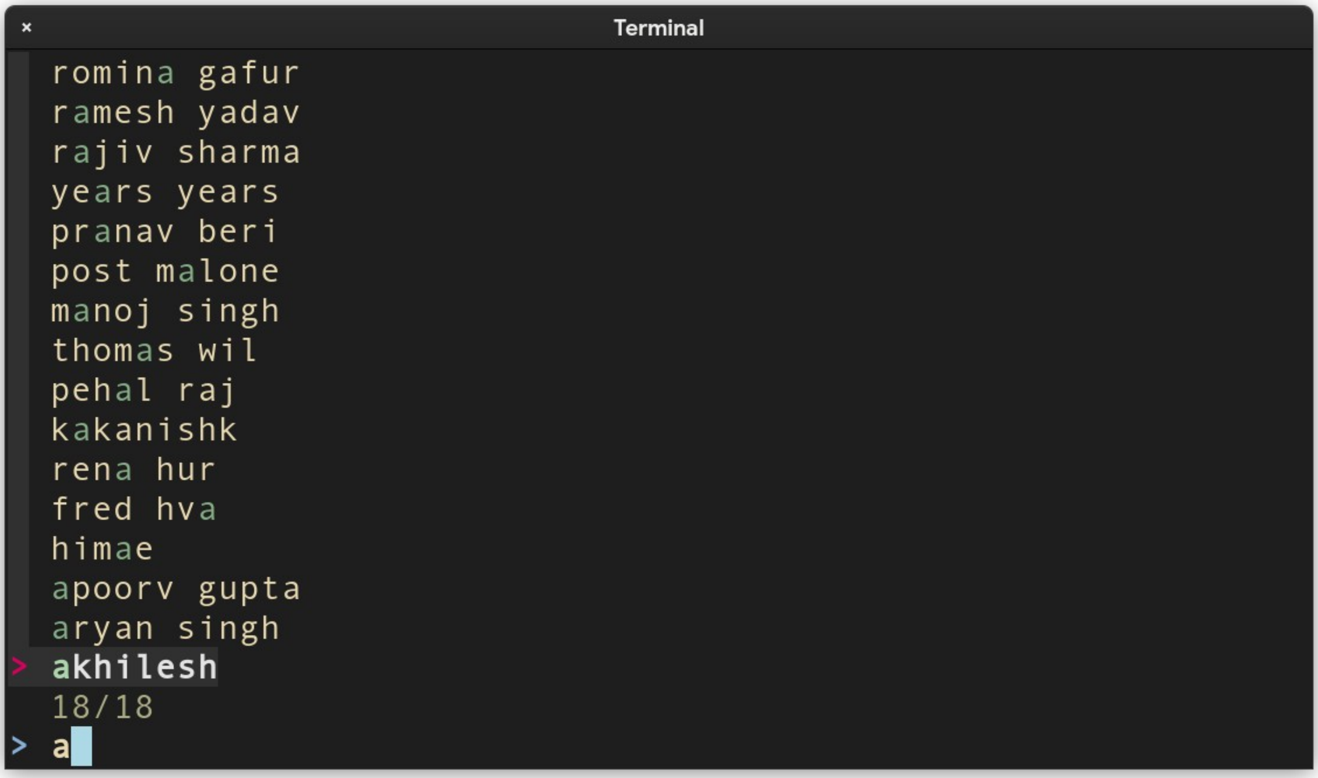
```
  6. Exit
  5. Add Member
  4. Display all Relations
  3. Display all Members
  2. Member Login
> 1. Suggest Groups
  6/6
>
```

Suggest groups :



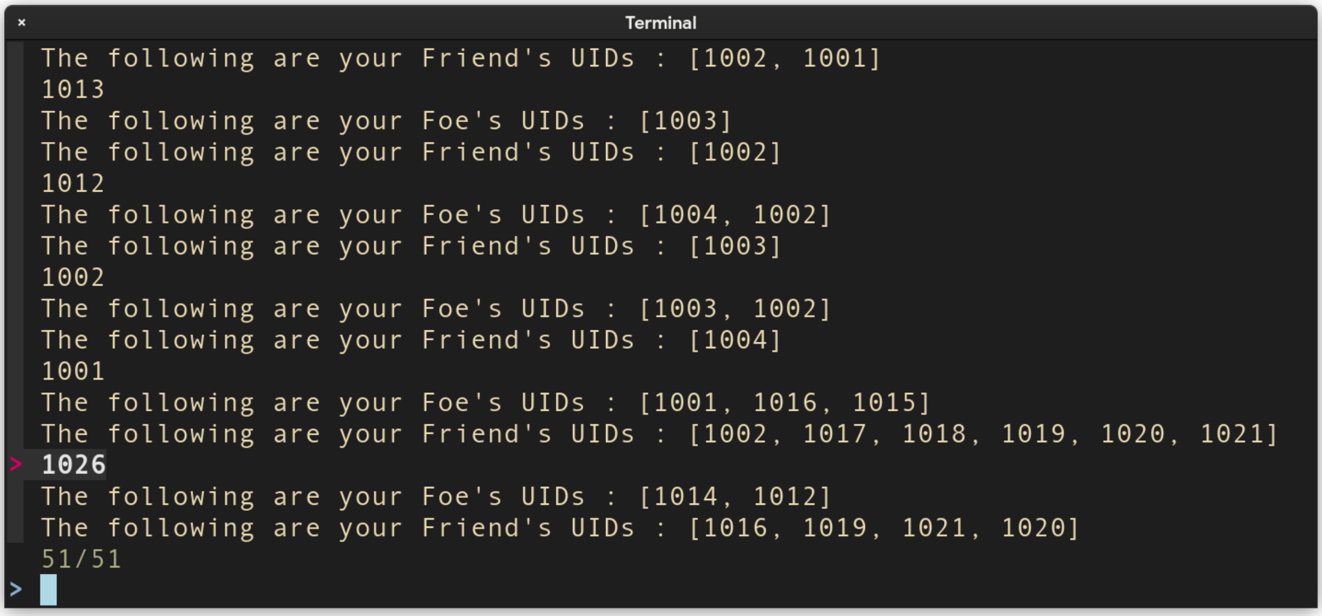
```
Authenticate : Authenticated
Enter the required skills :
Skill 1 : business
Skill 2 : sales
Skill 3 : communication

How many members do you want in the group : 2
1002    himae
1003    thomas wil
```

Display all Members :



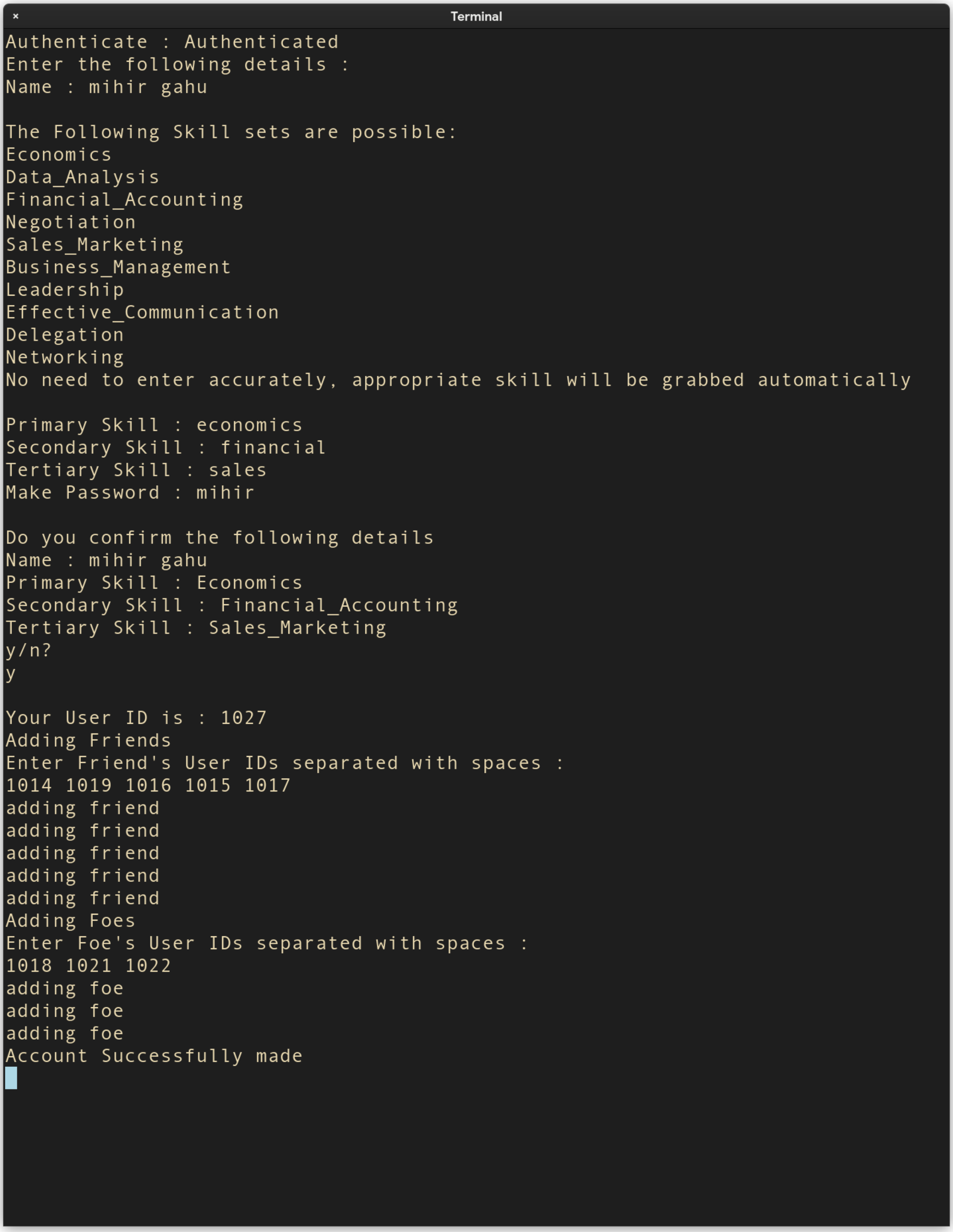
```
  romina gafur
  ramesh yadav
  rajiv sharma
  years years
  pranav beri
  post malone
  manoj singh
  thomas wil
  pehal raj
  kakanishk
  rena hur
  fred hva
  himae
  apoorv gupta
  aryan singh
> akhilesh
  18/18
> a
```

Display all relations :

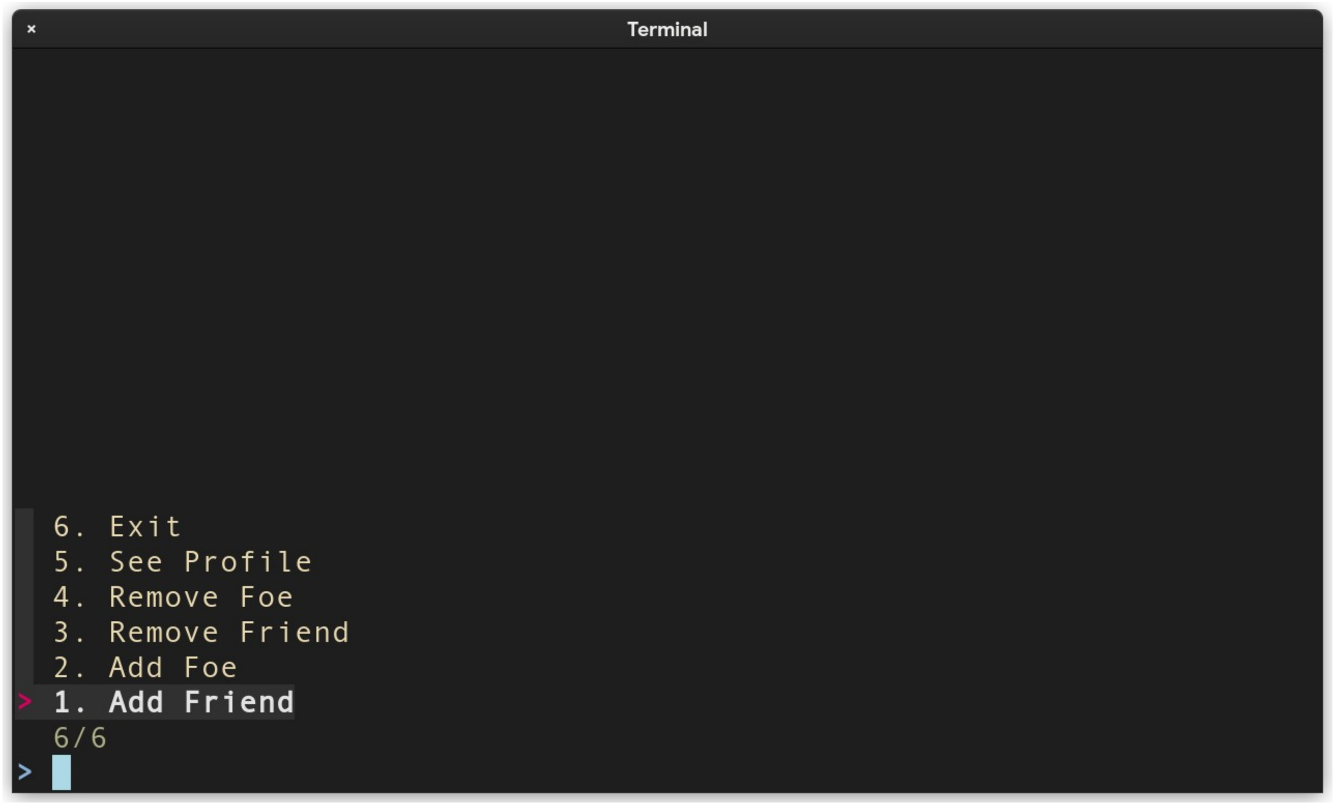


```
  The following are your Friend's UIDs : [1002, 1001]
  1013
  The following are your Foe's UIDs : [1003]
  The following are your Friend's UIDs : [1002]
  1012
  The following are your Foe's UIDs : [1004, 1002]
  The following are your Friend's UIDs : [1003]
  1002
  The following are your Foe's UIDs : [1003, 1002]
  The following are your Friend's UIDs : [1004]
  1001
  The following are your Foe's UIDs : [1001, 1016, 1015]
  The following are your Friend's UIDs : [1002, 1017, 1018, 1019, 1020, 1021]
> 1026
  The following are your Foe's UIDs : [1014, 1012]
  The following are your Friend's UIDs : [1016, 1019, 1021, 1020]
  51/51
>
```

Add Member :

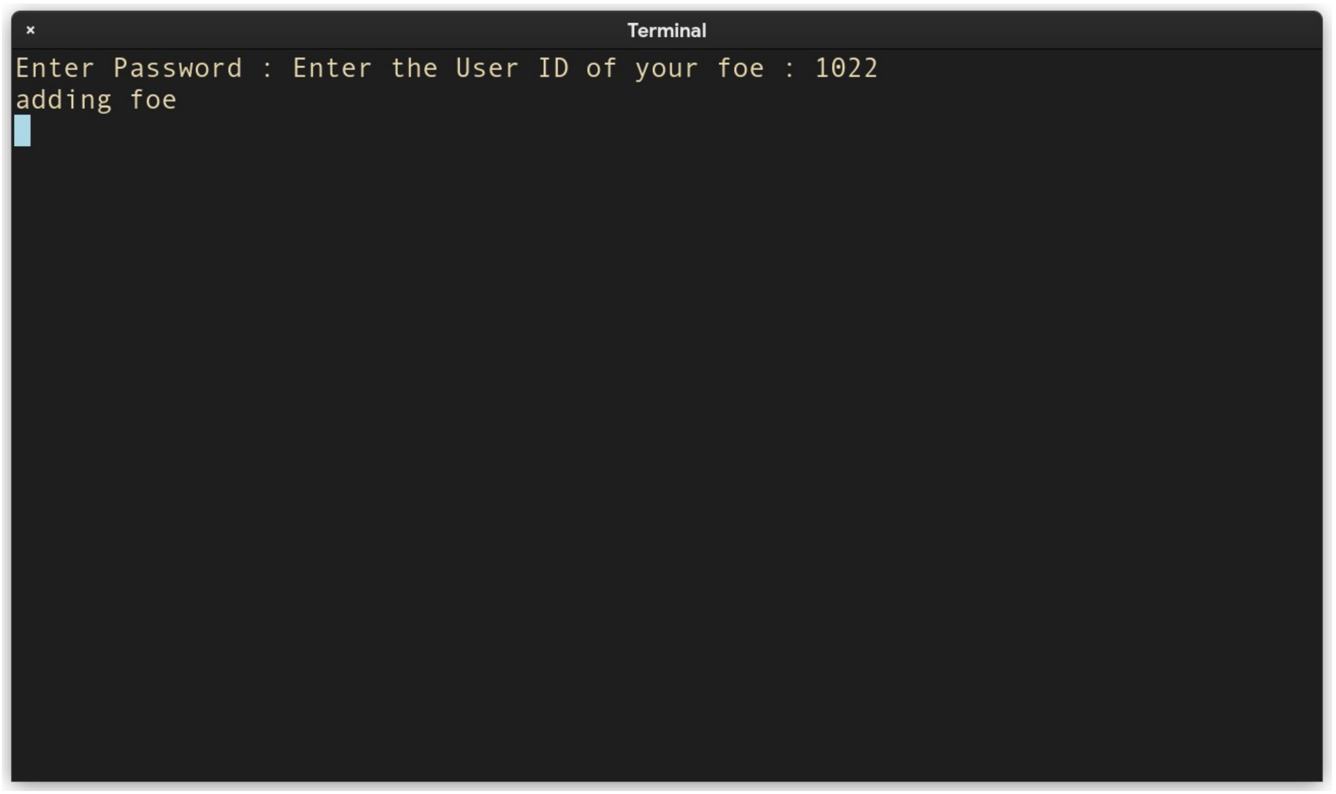

```
Authenticate : Authenticated
Enter the following details :
Name : mihir gahu

The Following Skill sets are possible:
Economics
Data_Analysis
Financial_Accounting
Negotiation
Sales_Marketing
Business_Management
Leadership
Effective_Communication
Delegation
Networking
No need to enter accurately, appropriate skill will be grabbed automatically

Primary Skill : economics
Secondary Skill : financial
Tertiary Skill : sales
Make Password : mihir

Do you confirm the following details
Name : mihir gahu
Primary Skill : Economics
Secondary Skill : Financial_Accounting
Tertiary Skill : Sales_Marketing
y/n?
y

Your User ID is : 1027
Adding Friends
Enter Friend's User IDs separated with spaces :
1014 1019 1016 1015 1017
adding friend
adding friend
adding friend
adding friend
adding friend
Adding Foes
Enter Foe's User IDs separated with spaces :
1018 1021 1022
adding foe
adding foe
adding foe
Account Successfully made
```

Member Login :



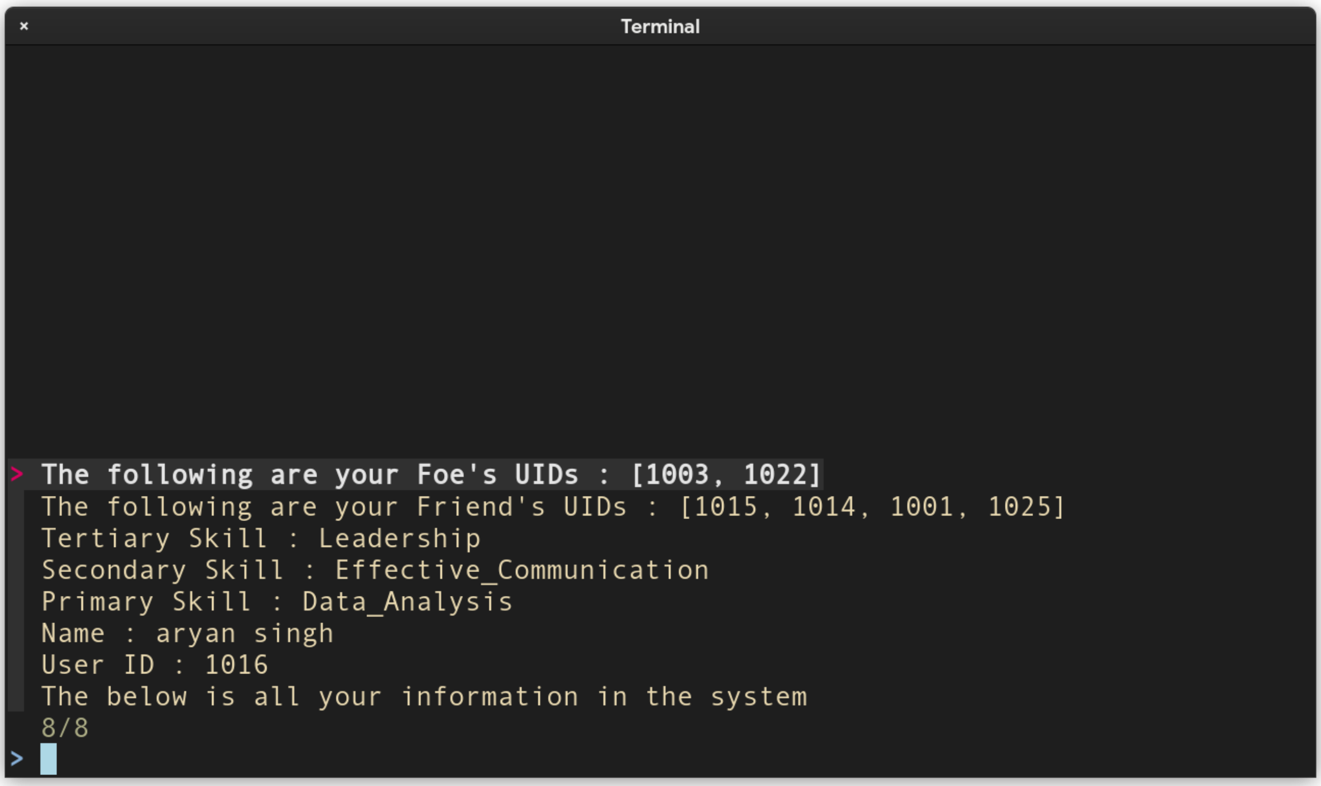
```
 6. Exit
 5. See Profile
 4. Remove Foe
 3. Remove Friend
 2. Add Foe
> 1. Add Friend
 6/6
>
```

Adding/Removing a Friend/Foe :



```
Enter Password : Enter the User ID of your foe : 1022
adding foe
```

See profile after logging in :



```
> The following are your Foe's UIDs : [1003, 1022]
  The following are your Friend's UIDs : [1015, 1014, 1001, 1025]
  Tertiary Skill : Leadership
  Secondary Skill : Effective_Communication
  Primary Skill : Data_Analysis
  Name : aryan singh
  User ID : 1016
  The below is all your information in the system
  8/8
>
```

Advantages of TMU(Team Making Utility)

The application offers some niche features which can be very powerful :

- **Suggesting groups using the skill and relationship** data of all the members.
- Member login, so that any **member can log in and change their profile** Information(for now only changing relationship information is supported).
- **Displays the names of all the members** in the database for searching.
- SQLite3 Database Implementation for **reliable and scalable database Interface**.
- **Passwords are hidden while typing** for improving security.
- **No passwords are stored on the disk**, rather **only their hashes**(using MD5 algorithm) **are stored** in the database table.
- Ability to add and delete friends and foes(people who are not good friends) from the database.
- The ID are never displayed with the names to any member(but are frequently visible to the admin).

Shortcomings of the TMU(Team Making Utility)

- **Over-prioritizes relationships over skill priority** while suggesting skill based group.
- **Not very privacy focused** as the skills are stored in a ser file accessible to the system admin.
- Also the **system admin can access the complete database** and read the skill information of any member.
- Is **slow while accessing the external database** as the interface in the middle limits the throughput.
- Has a c**onsole interface**, thus is **not very user friendly**.

Future Scope

I may Extend the functionality of the program to also accommodate the like and similar social media like features if the development remains consistent.

As if we have such an index then people will be able to see their rating and how much people want to work with them, with this information they can improve themselves to be socially better and more interactive in a good way.

I may also add a functinality to edit personal details like skills, name and password for a better user experience.

Currently there is now way to see how many people like you, so adding that would also be possible as a module already exists in the code but due to time limitations this feature could not be implemented. This would make this software totally usable in the real world solving real world problems, but for now to track the progress of the project you can follow the github link below :

https://github.com/aryan0154762/Team-Making-Utility

Conclusion

The application attempts to solve the problem of making groups of people having required skills set and good interpersonal relations with other members of the group to achieve the intended goal harmoniously.

The program aimed to accomplish the following :
1. Store the skills of each member/employee with their details.
2. Maintain their relations(friends or discordants) with other members
3. Make teams for a specific skill set while considering the interrelations of all the team members.
4. All the CRUD(Create, Read, Update, Delete) operations must be supported.

I used the build and fix methodology as the size of our project is small and relatively less complex than most of the enterprise software we repeatedly tested and debugged all of our code every time we added any new functionality.

Special features :
- **Implement Memory I/O for fast operations through memory,** This will help in performing seek heavy algorithms quickly as instead of accessing the disk(as the records are saved to the disk as well for permanent storage) every time there is a query for making group will make program very slow for sufficiently large sample set.
- Passwords are hidden and hashed and stored no password is present in the database.
- Uses fuzzy finder to make a easy to use console user interface.

Primary use case description : Suggest groups :
This module is the main highlight of the program as it tells us about all the users who can work in a group together having similar skill set and most compatibility with each other(as compatibility is subjective thus it needs to be calculated every time the group creation is required).

The function works by initially taking the input for skill set(requirement) and runs the loop based on the number of the parameters provided:
depending on the parameter provided the function only considers that and chooses the user on the basis of that.

Below is the progress of all the proposed milestones :

1. Account Control : Complete
2. Admin Tools : Complete
3. Admin Class : Complete
4. User Class : Complete
5. FileIO : Complete
6. Memory IO : Complete
7. Suggest groups : Complete
8. Initialization : Complete
9. Implement fuzzy menu : Complete

Helpful references :

https://www.sciencedirect.com/science/article/pii/S1877042815049149
https://www.stackoverflow.com/
https://www.google.com/
https://www.geeksforgeeks.com/
https://www.javatpoint.com/
https://www.tutorialspoint.com/
https://docs.oracle.com/en/java/javase/17/docs/api/
https://www.w3schools.com/sql/