Integrated College Management System

MC-122 Project

- Nidhi Dodiya (202303009)
- Param Savjani (202303046)

Before starting with the report...

(An aspect that we want to highlight)

In our program, we have made two database files named 'Admin_database' and 'Student_database', which may or may not be present in the system before the code runs. This decision is upto the user, whether he/she want these files to exist or not (before the code runs for the first time). If the admin file existed before the code runs (for the first time), then there will be an option to fetch data from that file. Else, there will be no such option.

If there existed the file and you didn't fetch data, then the option to fetch data will be available till the data has been fetched. But if there were no files at all, then after the code runs (for the first time), then both the files will be created in the run-time itself, which will last forever until you delete these files manually.

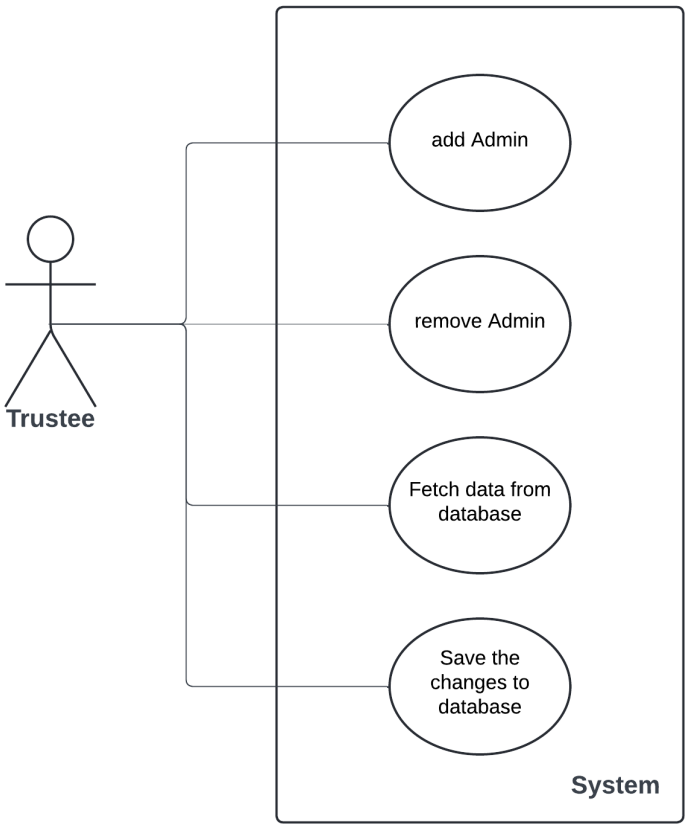
As the trustee of DAIICT, you need to enter the following credentials.
PLEASE NOTE THAT THESE ARE CONFIDENTIALS:
Username : **AnilAmbani**
Password : **DhirubhaiAmbani**

These are only given to the user while running the code. In order to check whether the entered Username or password is correct, we have used an encrypt method that will prevent the leakage of the credentials by directly looking at the code.

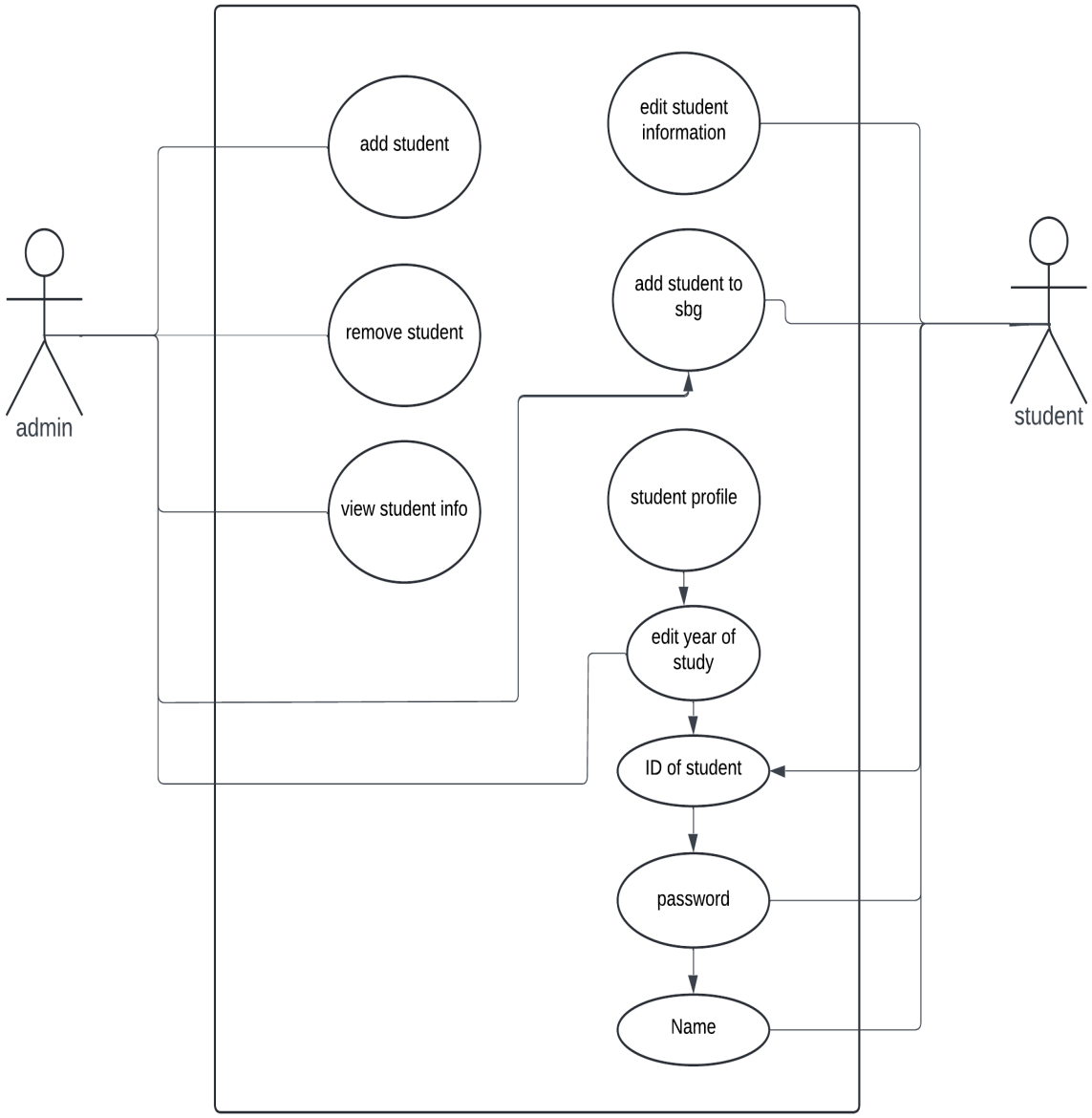
An important and unique logic

We have not edited the changes in the CSV and the TXT files. Instead, we have created new files each time a change is made. It is done using the data structure ‘arraylist’. Arraylist stores the data of admins and students. Any changes made in the arraylist will automatically be reflected in the files because a new file (with the same file name) is created with the data of the updated arraylist.

UML Diagrams of our program



This is the UML diagram which depicts what’s the role of trustee in the college. A trustee can add or remove admin, fetch or save data from or to the database.



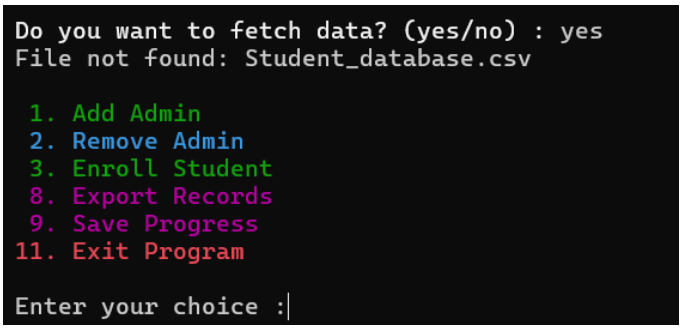
This is the UML diagram which depicts the functioning of both the admins and students. An admin can add or remove student, can view information of all the students, also can edit the year of study of students. Students on the other hand can edit the student profile except the year of study.

The project utilizes object-oriented programming to represent our college. It begins with the user assuming the role of the college trustee. The trustee's password is encrypted within the code, safeguarding it against unauthorized access. The system then encrypts the password input by the trustee and verifies it against the encrypted password stored in the code. Notably, our code lacks a method to decrypt the encrypted password string.

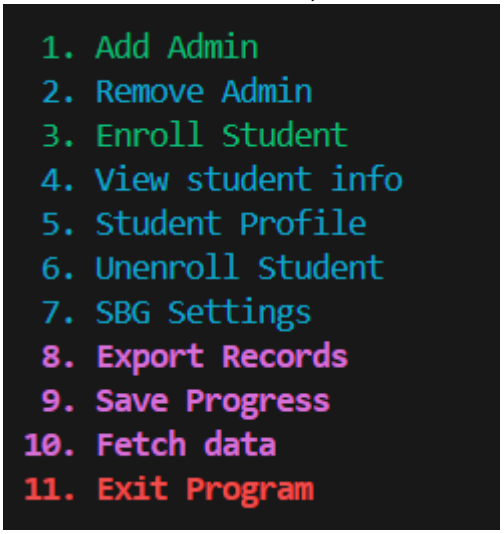
- **Upon correct password entry within three attempts, the following occurs:**
- **The system manages two databases: Student_database.csv and Admin_database.csv. These databases can either be generated during runtime or pre-existing.**
- **At the program's initiation, two scenarios are possible:**

1) If at least the admin database is present, a prompt will appear, requesting data retrieval from the database.

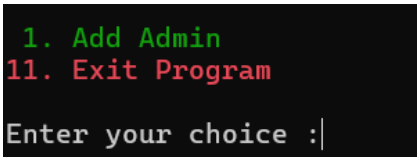
   a) If only the admin database is available, a total of — menus will be displayed.



   b) If both admin and student databases are available, 11 menus will be presented.



2) If neither database exists, a total of 2 menus will be available in the terminal.



In the project, once the trustee's password is initially verified, they can add or remove administrators without further password prompts. Subsequently, administrators have the capability to enrol students. Each student possesses various attributes and may also be a part of the student body government (SBG).

Class General :

- In the general class there is a method that capitalizes the first letter of the string and there are many final static strings that depict Ansi escape code of different colours (example : RED, CYAN, GREEN) and RESET and BOLD.
- This general class is made with an intent to contain methods and attribute that are general and common to most of the other classes.

Abstract class Person :

This class has a contain basic attributes and method that should be extended by a class related to a person.

Attributes present in this class :

- Protected string of username and password.
- Protected array of string that contains 3 elements (Fist name , middle name, last name)
- Protected static final scanner for take data from terminal.

Constructer of this class :

- This constructor takes a **username** as input. Throughout the code, it compares each person using their username. If we need to check whether a student with a specific username exists, we can use this constructor.

Methods present in the class :

- **getName :** Returns the first name of a person.
- **getPassword :** Retrieves the person's password.
- **getNameFromUser :** Prompts the user to enter a full name, ensuring it contains only alphabets and spaces before returning it as a string. If an invalid name is entered, it requests re-entry until a correct name is provided.

Class Admin :

ATTRIBUTE PRESENT IN ADMIN CLASS :

the arraylist named totaladmins they stores object of admin class.

This arraylist is engaged in the work where the data needs to be fetched and export of the data.

CONSTRUCTORS OF THE ADMIN CLASS

These constructors call the corresponding methods of the constructors of the person abstract class using super keyword.

METHODS IN THE ADMIN CLASS:

- A getter and setter are made in this class for username.
- We override the equals method that considers two admin objects two be equal if and only if their usernames are equal.
- We overrided the hashcode method that will generate the key of the admins on the basis of their usernames.
- isEmpty method that will check whether the totalAdmins arraylist is empty or not in Boolean.
- **loginAdmin:**
  - There will be max allowed attempts=3.
  - It will ask you to enter the username and then will check whether an admin of that username exist or not in the arraylist.
  - If it exists, then it will proceed and ask you to enter the password.
  - If you entered the password incorrectly, then it will not ask you to re-enter the username that you entered correctly previously.
  - If the entered password or username was incorrect than both will be held in account for the number of chances remaining in the login process.
  - If in 3 or less attempts, the username or password was entered correctly then the correct index of that admin will be returned. Else it will return -1.

- **addAdmin:**
  - It will ask how many admins do you want to add, if you entered non positive number then it will alert you to re-enter. Else if, you enter in a form which is not numbers then it will show “invalid input”. The number entered will be stored in ‘n’. Now the following steps will occur in loop ‘n’ times:
    - The method getNameFromUser is called and the string returned is splitted into three parts and stored as three strings and stored in an array of strings ‘temporaryName’.
    - Then it will ask for username. If the username contains spaces or is already in use then it will ask you to re-enter the username until a valid username has been inputted.
    - Afterwords, it will ask you to enter the password. If you enter the password with spaces, then it will ask you to re-enter the password and will ask you until a valid password is inputted.
    - Now, if the username and the password has been correctly given, then it will form a temporary admin. And this admin will be added to the arraylist ‘totalAdmins’.
  - There is no bound in the number of chances. You will be asked until you fulfil every condition.
- **removeAdmins:**
  - This method will ask for the username of the admin to be removed and will check whether an admin of such username exists or not.
  - If exists, then it will remove that admin from the main arraylist of admins and end the method.
  - Otherwise there will be one less chance for you to enter a correct username and remove that admin.

Class Student

ATTRIBUTES IN THE STUDENT CLASS

There is an arraylist “totalStudents” that stores objects of class Student.

Other attributes are idOfStudent, yearofSudent, isSBGMember.

CONSTRUCTORS IN THE STUDENT CLASS:

- Default constructor
- Constructor that takes only id as the parameter.
- Constructor that takes id, password, yearofStudy and the full name of the student

METHODS IN THE STUDENT CLASS:

- **isEmpty method :**

  Tells whether the totalStudents arraylist is empty or not in Boolean.

➢ Equals method :

We override the equals method that considers two student objects two be equal if and only if their ids are equal.

➢ Hashcode method

We overrided the hashcode method that will generate the key of the students on the basis of their ids.

➢ addStudents, removeStudents, loginStudent :

These methods are mostly the same as that in the admin class. In the student class, there are some extra attributes, yearOfStudy and isSBGmember. In place of username, the student has id.

NOTE : We cannot make a generic method inside the abstract class person, that will function to add or remove or login in both the classes-admin and student. It is because there are many point of differences in these two classes that will not allow us to end up with a common method.

➢ showAllRegisteredStudents :

This method will be shown in the menu if and only if there is atleast one student, hence atleast one admin. This method will call the method loginAdmin. As we know that the loginAdmin method returns the index of that admin in the 'totalAdmins' arraylist. Hence, here if the returned value is not -1, then it will print all the informations of all the students.

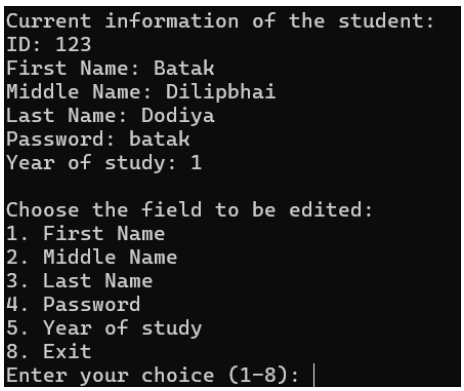
Total Students at DA-IICT: 11

ID	Name	Password	Year of Study	SBG Member
123	Batak	batak	1	No
1	Jai	pass1	2	Yes
2	Oviya	pass2	1	Yes
3	Eshan	pass3	4	Yes
4	Ema	pass4	3	No
5	Avni	pass5	1	No
6	Nayan	pass6	2	No
7	Ishita	pass7	3	No
8	Laksh	pass8	4	No
9	Saanvi	pass9	1	No
10	Manav	pass10	2	No

Here is a small example, where this method will show the info of each and every student.

editStudentInformation:

This method will firstly call the loginStudent method. As we know that this method will return the index of the student in the 'totalStudents' arraylist. Hence if the returned integer is not -1, then it will firstly display the current informations of that particular student. Followed by this, it will display a menu consisting of 8 options as depicted below:



This is the menu with the 8 choices. After any of these operations, the current information will be displayed. This will continue until you enter option 8.

Class SBG

ATTRIBUTES PRESENT IN THE SBG CLASS:

An arraylist 'sbgMembers' that stores the objects of class student (those students who are SBG members).

METHODS PRESENT IN THE SBG CLASS:

findStudentbyID :

- Parameter input-ID
- Return type-Student

- This method will make an iterator of type Student and will return if a student with the entered ID (if such a student do exists).
- If student with the entered ID do not exist, then it will return null.

addMember :

- This method will add a student to the SBG.

- First of all, it will ask you to enter the id of the student to be added to the SBG.
- Then it will call the above method 'findStudentbyID'. If this method returns null, then it means that no student with such an ID exists, and you will have one lesser chance than existing chances.
- If the method returned a student,
  - If the attribute 'isSBGMember' of that student is 1, then that student is already a member of SBG.
  - Else, it will then add that student to the arraylist of SBG 'sbgMembers', and also will make the attribute 'isSBGMember' =1.
- There is a bound in this method, if you entered a non existing ID. The maximum number of attempts is 3.
- Once a correct student id has been entered in the given number of chances, the method will add that student to SBG and will return.
- Else, it will say "You have exceeded the maximum limit of attempts!"

removeMember:

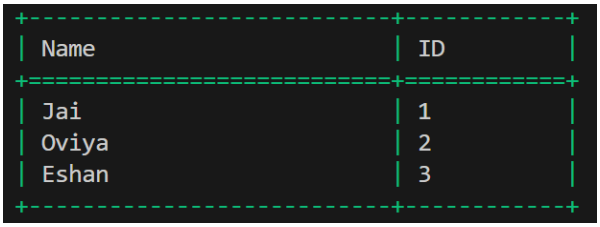
- This method will firstly check if the SBG arraylist is empty or not. If it is not empty then it will then ask for the id of the SBG member to be removed.
- Next, it will verify whether the student with that id is an SBG member or not.
- Then it will remove that student from the arraylists of SBG 'sbgMembers', and also will make the attribute 'isSBGMember' =1.
- There is a bound in this method, if you entered a non-existing ID. The maximum number of attempts is 3.
- Once a correct student id has been entered in the given number of chances, the method will remove that student from SBG and will return.
- Else, it will say "You have exceeded the maximum limit of attempts!"

replaceMember:

- It will firstly check if the arraylist of SBG members is empty or not.
- Also it will check whether the sizes of arraylist of SBG and the Students is equal or not. If they are equal, it means that all the students are in SBG, and hence no student is left to be an SBG member. Hence, in this case, it will return and say,"All students are already the members of SBG."
- After the above two conditions are seen, the code (if possible) will ask for the id of the member to be replaced.
- Now the method 'findStudentbyID' will be called and it will return the Student(if exists with that id) or will return null. The returned student will be stored in 'oldStudent'.
- Now, the current method will check if the returned value is not null or the student is already a member of SBG.
- Then, it will ask for the id of the student that will replace the SBG member.
- Again the method 'findStudentbyID' will be called and it will return the Student(if exists with that id) or will return null. The returned student will be stored in 'newStudent'.
- If student exists, then it will remove the oldStudent and add newStudent in SBG. Consecutively it will make the attribute 'isSBGmember' of oldStudent=0 and newStudent=1.

viewMembers:

- It will first check whether the arraylist of SBG is empty or not.
- If the arraylist is non-empty, then it will display the list of the students in SBG.



Name	ID
Jai	1
Oviya	2
Eshan	3

Class Datasaver

ATTRIBUTES PRESENT IN THE SBG CLASS:

There are two string attributes used to store the path of two CSV files named:

- Admin_database.csv
- Student_database.csv

METHODS PRESENT IN DATASAVER CLASS:

readCSV

- Return type-List<String[]> list whose elements are array of strings.
- 'data' of type List<String[]> is declared.
- Now each line of the CSV file is read and it will be split into an array of strings. These arrays of strings are added to data.
- At last data is returned, which stores each line as its element, where each line is an array of string.

fetchAdminData

- This method creates a list 'adminData' that has elements of type array of strings. This list will contain the returned list after the readCSV method is called with the file path of the Admin_database.csv file.
- Now as we know that each element of this list is an array of strings. These strings will be the username, password and the full name of the admin.
- Now as we know that each line correspond to the info of one admin, this method will form a new admin with the strings in each element of the list and then add this admin to the 'totalAdmins' arraylist.

fetchStudentData

- This method is similar to the fetchAdminData method and will similarly add all the students from the csv file to the 'totalStudents' arraylist.
- Additionally, those students who are SBG members will be added to the arraylist of the SBG members and hence will edit the arraylist.

processYes

- This method is a helper method that will call both the methods 'fetchAdminData' and 'fetchStudentData' and will return true (in all cases whether any or both among these files exist or not). And will return false if there occurred an error in fetching the data.

fetchData

- Return type- boolean
- This method takes a string input and then checks if the input string is 'yes' or 'no'. If it is 'yes', then the method processYes is called else it will return false.
- If the input string is neither 'yes' nor 'no', then it will enter a while loop which will ask you to enter a string until you enter yes or no.
- The same procedure is made when you enter yes and no.

writeToCSV_Admin

- This method will take data from the 'totalAdmins' arraylist and will copy the data of each admin in a new line in the CSV file of admins.
- The code will traverse through each admin in the arraylist and will join the username, password and the name with comma symbol and will print the line in CSV file of admin.

writeToCSV_Student

- This method is similar to the above method.
- It writes the data of each student in the CSV file of student.

formatAdminData

- This method takes an input object of class Admin.
- It returns a formatted string containing Username, password and Name of the specific admin.

formatStudentData

- This method to similar to the above method.

writeAdminDataToTxtFile

- This method will iterate through each admin of the admin arraylist and will call the method formatAdminData.

- The returned formatted string is written into the TXT file.

writeStudentDataToTxtFile

- This method to similar to the above method.

saveDataToTxtFile

- This method calls both the methods writeAdminDataToTxtFile and writeAdminStudentToTxtFile.
- Hence it will combinedly save the data of admins and students into the txt file.

Class MainClass

Attributes present in the MainClass:

- String USERNAME: This is an encrypted string, which will be used to compare with the entered username by the user. The input username will be firstly encrypted by an 'encrypt' method and then will be compared with this pre-defined encrypted USERNAME.
- String PASSWORD: This is an encrypted string, which will be used to compare with the entered password by the user. The input password will be firstly encrypted by an 'encrypt' method and then will be compared with this pre-defined encrypted PASSWORD.
- String KEY: This is a string that stores 'secret', which is used in the method 'encrypt' which will be explained in the methods section of this class.

Methods present in the MainClass:

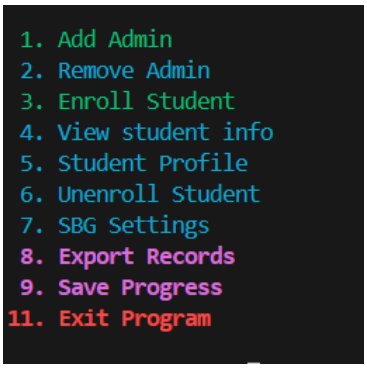
Encrypt

- Input type-String
- Return type-String (encrypted)
- In this method the input is converted to an array of corresponding bytes and stored in inputBytes array.
- keyBytes array stores the bytes of String KEY.
- outputBytes array was predefined to store the encrypted bytes after the following operation.
- ```
  for (int i = 0; i < inputBytes.length; i++) {
      outputBytes[i] = (byte) (inputBytes[i] ^ keyBytes[i % keyBytes.length]);
  }
  ```
- In this the outputBytes array will contain the bytes calculated by the exor of the bytes of the inputBytes and keyBytes.
  ```

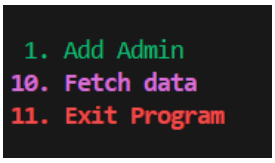
- Here, in any case if the length of inputBytes exceeds keyBytes, hence here we have used modulo function to have the index to be cycled again starting from 0 in keyBytes.
- After this loop operation, the outputBytes is encoded to string and then returned.

➢ displayMenu

- Return type-integer array.
- This method will display the menu which is available according to the present scenario (for e.g. the choice 'removeStudent' cannot be available in the menu unless and until atleast one student has been added in the college).
- This method has an arraylist named 'validChoices', which will store all those choices of the menu which are right now displayed in the console.
- Following are all possible choices in the menu:



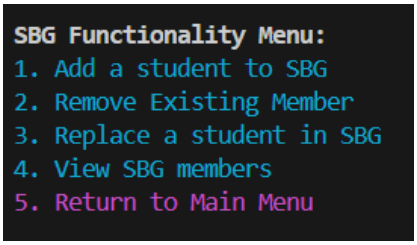
- Also, there is a Boolean variable 'data' which will act like a flag and will be true if and only if fetch data operation has been called. Once it has been called at any instance of time, the variable 'data' will be set to true.
- At first, choice 1(Add Admin) and 11(Exit Program) will be the default choices among all.
- After at least one admin is added, choices 2,3,8,9 will be activated.
- Also, if the variable 'data' is false then choice 10(Fetch data) will be visible. It will be visible until the variable data is set true.



- After at least one student is added, choices 4,5,6,7 will be activated.
- All these valid options available are added to the arraylist 'validChoices'.
- At last, an array 'choices' of length same as that of the arraylist 'validChoices' is made, and then each element of the arraylist is copied to the array.
- The array 'choices' is returned.

➢ MenuForSBG

- There is an integer variable 'sbgChoice' which is initialized as 0.
- Then the code will enter a do-while loop which will run until 'sbgChoice' is set to 5(Return to Main Menu option).



- In the menu there will be 5 choices. There are several switch cases corresponding to these choices that will call the specific methods from the SBG class, and will exit if the choice made is 5.

➢ startingDisplay

- This is a simple method which we have made to welcome you in our program. It will guide you, what to enter as the trustee of the college.

➢ checkTrustee

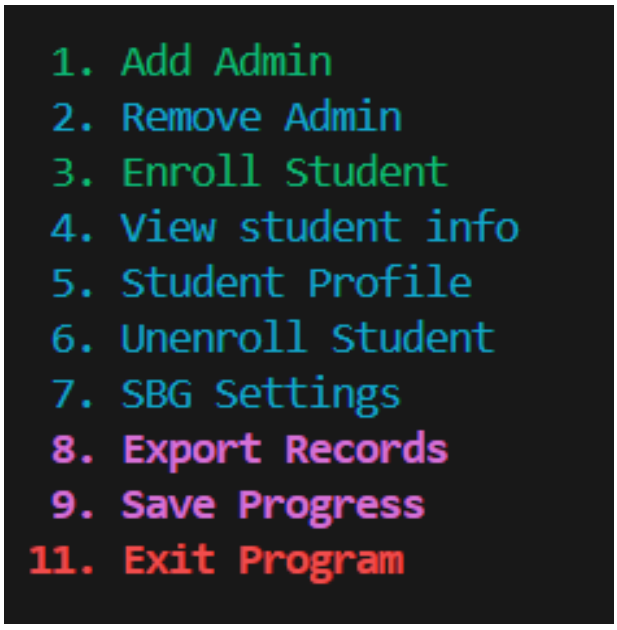
- This method is made to check whether the user using the program is the trustee of the college or not.
- There are three variables declared in this method
    1. Boolean verified (initially false)
    2. Boolean usernameCorrect (initially false)
    3. Int loginAttempts (initially 0)
- Then the code enters a do-while loop which will end only when variable 'verified' is true or loginAttempts exceed the maximum attempts allowed.
- Inside the loop, it will first ask you to enter your username. Then the entered username will be encrypted using the 'encrypt' method and will compare with the pre-defined variable 'USERNAME'. If both matches, then the variable 'usernameCorrect' is set to true.
- If the entered password matches the actual password (after encryption) then the variable 'verified' is set to true, and hence will break the do-while loop.
- At last, if 'verified' is false the variable 'loginAttempts' is incremented and the remaining chances are displayed.

➢ exitFunction

- This method is used to gracefully exit the program.
- Input – integer variable 'old'
- Here the variable 'old' is defined in the main method of the program which stores the last choice made by the user.
- Here if the last choice entered i.e. 'old' is not -1,4, and 9 then it will ask you whether you want to :
    i. Save the file and exit
    ii. Exit without saving
    iii. Do not exit
- Else if 'old' was -1, then it will say, "There were no changes to be saved."

➢ main method

- The main method will call both the methods of the MainClass-startingDisplay and checkTrustee.
- Firstly, it will check whether the Admin_database.csv file exists or not. If that do exists then it will ask you whether you want to fetch data from this file or not.
- If you entered ‘yes’ then it will fetch the data and will set the ‘data’ to true. If file does not exist then also it will set ‘data’ to true.
- As discussed above, an integer variable ‘old’ is defined, including ‘choice’ which is initially set to -1.
- The code then enters a do-while loop, which will run until the choice made is not 11.
- Inside the loop, an array ‘validChoices’ is declared which will store the array returned by the function ‘displayMenu’. Hence, this array will contain the array of all the available valid choices among all 11 choices of the menu. Side by side, the ‘displayMenu’ function would print all these available choices in the terminal.
- Then it will ask you to enter a choice. This choice is then compared with every integer inside the ‘validChoices’ array and if found it will further go inside. Else, it will ask you to enter a valid choice.
- There will be switch cases where each choice will have corresponding method called. Once you enter 11, you will exit the program.




Thank You