PSG COLLEGE OF TECHNOLOGY

OOPS PROJECT REPORT

STUDENT PERFORMANCE TRACKER

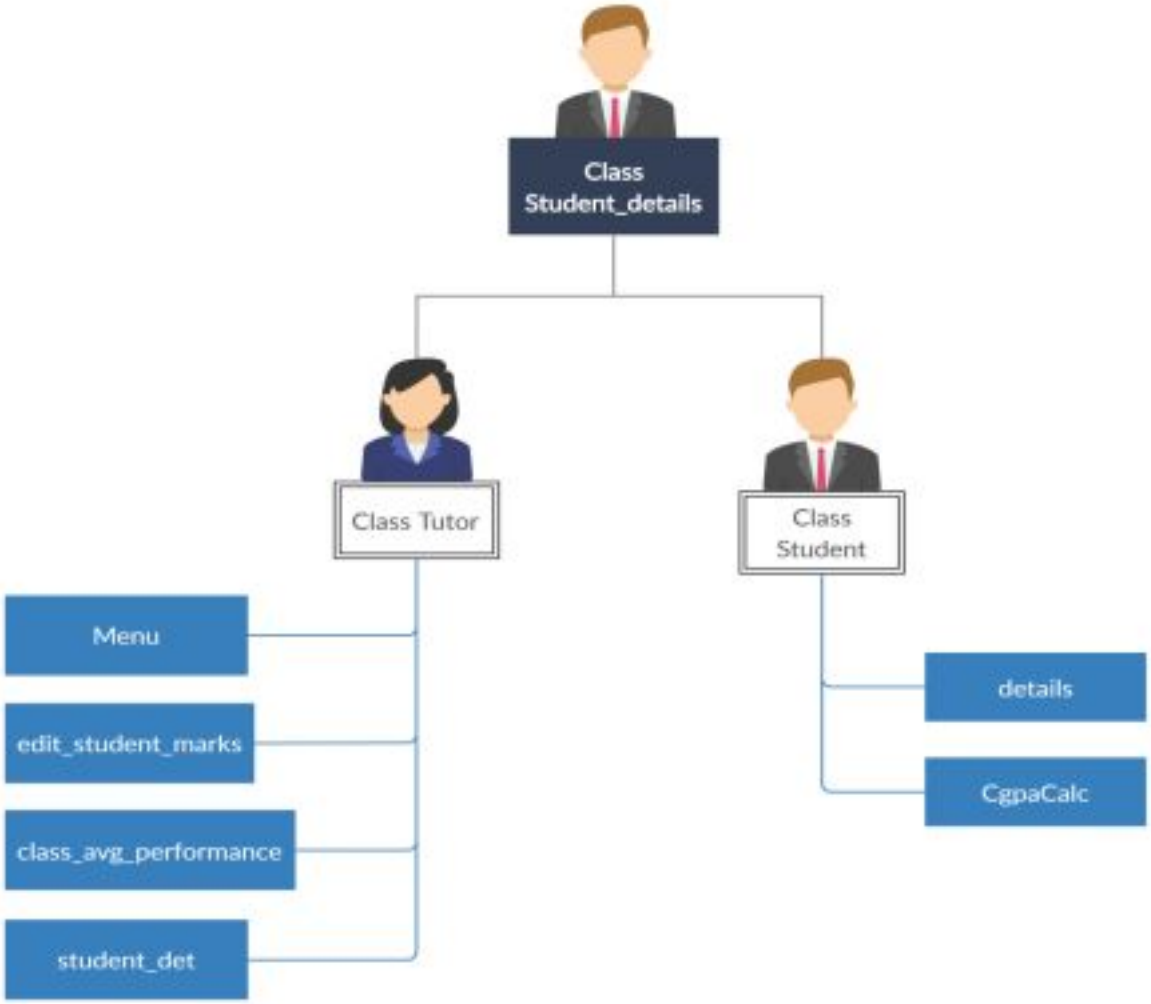
GROUP NUMBER – 6
GROUP HEAD – ADHARSH (19Z302)
GROUP MEMBERS :-
ADITYA SHARMA (19Z303)
BHOOVIKA (19Z310)
GOKUL (19Z314)
PAVITHRA YAZHINI (19Z337)
SHIVRAJ (19IZUS016)

Problem statement:

To develop a system called ***Student performance tracker*** using which one can view a student's academic performance. Another goal was to provide edit privileges only to certain types of accounts while the rest were barred from performing these actions. The whole system is to be defined such that students/faculty will be able to use the system through a graphical user interface(GUI).

To achieve all of these targets and to help us develop the back end(code that executes our actions) and the front end(GUI) we used lntelliJ IDEA. IntelliJ IDEA is a Java integrated development environment (IDE) for developing computer software, which is developed by JetBrains.

Class Diagram:



Features available to the user in the application:

We shall look into the features that the application offers by looking at the flow of control when the application is launched (Java code is executed).

1)Separate login details

First, we invoke the login page in the oops class. This will pop a user-friendly GUI where they have to enter the username and the password to proceed.

Class login:

{

This class has methods that are used to validate the user imputed *username and password*. If it is valid based on the username it will call either Class tutor or Class student. Methods:

void authentication()

{

It will first call the method check_if_valid() and store it in a variable called receive. Based on the value stored in receive the user is directed to the Student/Tutor login page.

}

Int check_if_valid()

{

It will run the username and password through the student_pass.txt or tutor_pass.txt and check if such a user exists and whether the password matches. If it doesn't match it pops up a small notification saying that the username or password is invalid.

}

}

2)Database handling:

To keep things light and simple we took our main database as an excel sheet where details about what courses the class is taking up and the marks scored in that particular course by a student.

Class student_details:

{

This class directly interacts with the data in the excel sheet and updates it as changes are being made.

Methods:

void fetch_course_details()

{

This method implements a part of *Appache POI* called XSSFWorkbook to directly interact with the excel sheet. It searches the columns and extracts the following details and stores them in arrays course and max_marks respectively:

>Name of the courses taken by the student

>Maximum marks the student can score in that course

}

void fetch_student_details()

{

This method uses the similar procedure as the previous method to access the data and store the marks scored in a particular course by the student.

}

void setmarks()

{

This method uses the *Appache POI* to update new details to the excel sheet as the program is being executed.

}

}

3)Restricted access to data based on the type of user:

Not every user will be given complete access to the database otherwise there is a high probability for malpractice in entering invalid values and such which could collapse the whole system and produce inaccurate results. Restricting data also helps increase the modularity of the code and allows

Users:

3.1)Tutor:

This user type has complete access to the database. They can access data pertaining to any student of their choice and edit them if necessary.

Class tutor

{

Methods:

void menu()

{

This will show a list of actions the tutor can perform. The options

are:

1) Check student details

This action will display the marks scored in each course given his/her roll number. It achieves this by calling the student_det() method.

2) Edit student marks

This action will allow the tutor to change a particular students marks obtained in each subject. It achieves this by calling the edit_student_marks() method.

3) Class average performance

This action will display how the whole class has performed in a particular course. It achieves this by calling the class_avg_performace() method

4) Exit

This action will take the user back to the login page once they are done. This helps secure the use of this account by anyone else but that particular tutor.

}

void student_det()
{
This method creates an object of the class student_details and uses its getter methods to obtain data and display it by running it through a for loop.
}

void edit_student_marks()
{
This method creates an object of the class student_details and uses its setmarks methods to update the new marks to the excel sheet.
}

void class_avg_performace()
{
This class creates an object of the class student_details and displays the details from the excel sheet.
}

}

3.2)Student:

This users access to the database is restricted to only viewing it. This helps ensure the originality of the data and denies the possibility of malpractice from happening.

Class student
{

Methods:

Void details()
{
This method acts as a menu for the student interface. It lists out the possible actions the student can perform. Those actions are as listed:
1) View marks
This action displays the students the marks he has scored in all the courses. It achieves this by using the getter methods of the student_details class.
2) Calculate CGPA
This action asks the student to enter the grades he got in the semester exam for each course along with his previous semesters CGPA to help him easily calculate his current CGPA. It achieves this by calling the CgpaCalc() method.
3) Exit
This action will take the user back to the login page once they are done. This helps secure the use of this account by anyone else but that particular student.
}

void CgpaCalc()
{
This method calculates the students cgpa based on the credits and marks scored in the current semester exam.
}
}

4)Graphical User Interface:

The system uses Swing to develop a user-friendly graphical interface that would help the user properly navigate through the application.

Challenges faced:

1) The *Appache POI* for some reason just wouldn't read the data and this took a while to debug.
2) Developing the back end was quite simple but it wasn't the same case for the front end. We were able to create a panel,buttons,etc, but when the button was clicked to perform an action the ActionListner linked to it would receive the request but it wouldn't take the user into their particular login pages.
3) Linking of the GUI with the back end was the greatest difficulty we faced as the code just rejected the inclusion of the login or student_details class in it. We found a work around through this by making the button's action listener point to the class itself.

Contribution of team members:

1) 19Z302 - Adharsh : Debugging and linking the GUI with the back end 2) 19Z303 - Aditya Sharma : Worked on extracting data from the excel sheet and on the tutor class
3) 19Z310 - Bhoovika : Worked on GUI and helped in deciding the features the tutor and student class supposed to have
4) 19Z314 - Gokul : Dealt with customising the GUI and report work
5) 19Z337 - Pavithra Yazhini : Worked on student class and helped in deciding the features the tutor and student class supposed to have
6) 19IZUS016 - Shivraj : Report work

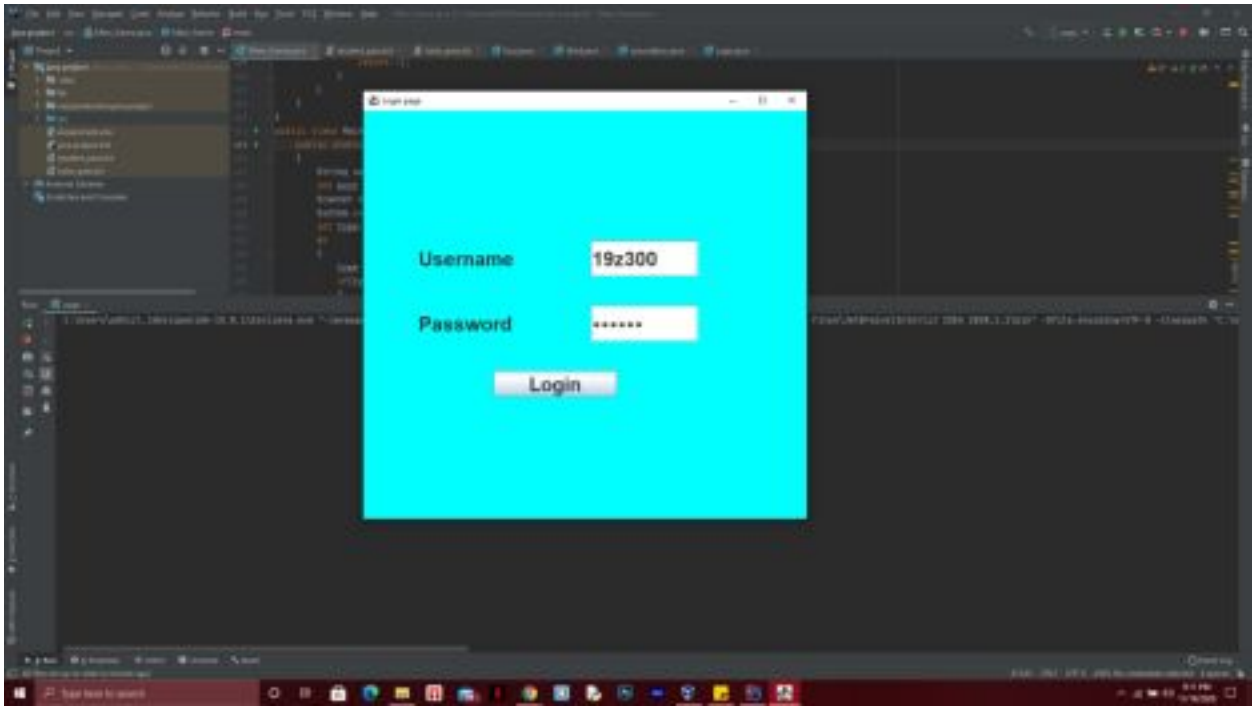
Annexure 1:

Java Code - *https://github.com/admin0911/Student-Profile-Tracker*

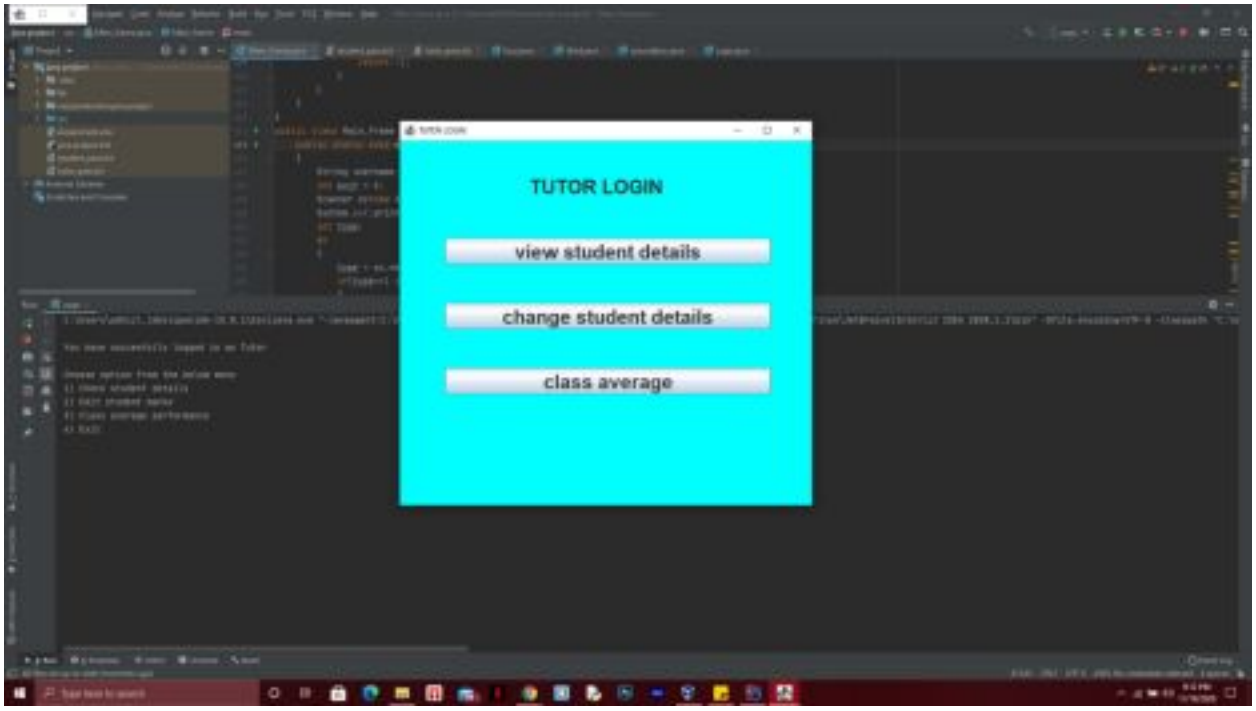
Annexure 2:

Snapshots of the code:

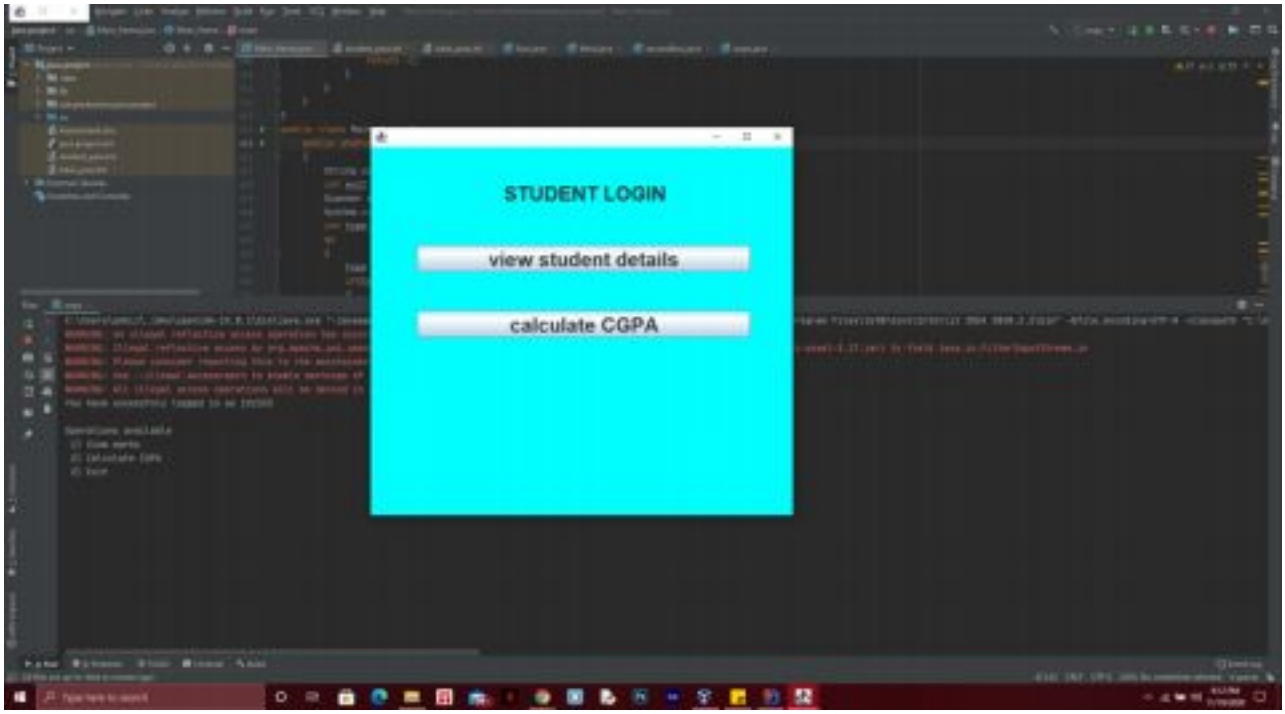
1)login page



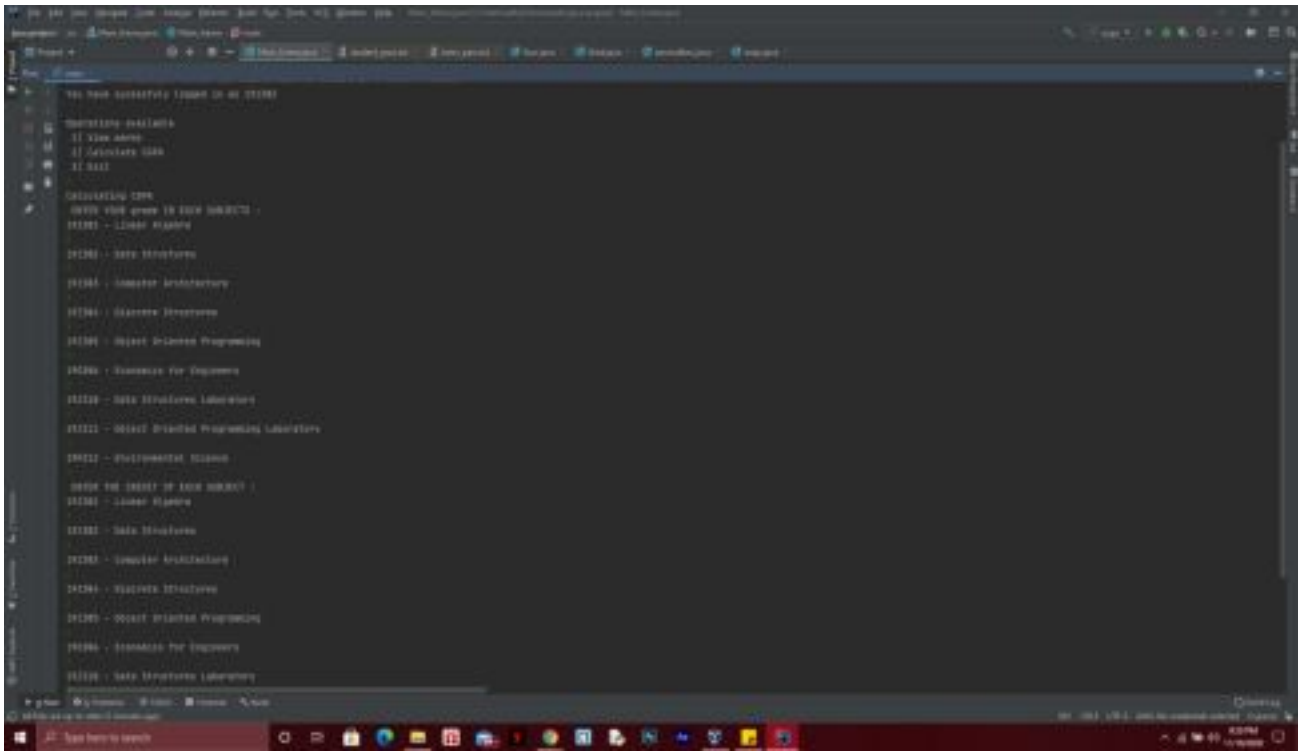
2)Tutor login:



3)Student login:



4)Features of the code being executed:



References:

1) Front end (Swing):

a) https://www.javatpoint.com/java-swing

b) https://www.youtube.com/watch?v=5o3fMLPY7qY

2) Back end (File handling and java):

a) https://www.javatpoint.com/how-to-read-csv-file-in-java

b) https://www.amazon.com/Java-Complete-Reference-Herbert Schildt/dp/0071808558

PLAGIARISM REPORT