PROJECT REPORT
DATABASE MANAGEMENT
CSE 303
GROUP 27
TEAM DATA ARMOUR

Name	ID
Md Tuhin Al Jobayer	1831124
Farsheed Rahman	1830360
S.M. Arif Mahmud	1830398
Safayet Khan	1820080
Sheikh Raiyan Hossain	2021937
Md Samiur Rahman	1930661

Contents

CH-1 INTRODUCTION:

BACKGROUND OF THE ORGANIZATION:

Independent University, Bangladesh (IUB) established in 1993 is the leading private university in Bangladesh with an explicit focus on Research and Global partnerships.

The Independent University, Bangladesh (IUB) has robust and versatile schools – notably consisting of following:
● School of Business & Entrepreneurship
● School of Engineering, Technology & Sciences
● School of Environment and Life Sciences
● School of Liberal Arts & Social Sciences
● School of Pharmacy and Public Health.

The institution has actively contributed to the development of the education industry in Bangladesh and has produced competent and knowledgeable scholars who have made contributions both domestically and internationally. The University Grants Commission (UGC), the Ministry of Education, and other necessary institutions for each of the schools, along with regular curriculum updates, the implementation of a system to track student performance based on a quantified approach between course curriculum and standards set by UGC and the Bangladesh government, and ongoing student performance monitoring have all helped IUB achieve this.
The objectives of IUB are to produce graduates of international standards in the local environment who have the knowledge and necessary skills to provide leadership in business, public service, and welfare; to encourage and support useful research; to create knowledge; and to offer opportunities for adults to continue their education.

Figure 1: Independent University, Bangladesh

BACKGROUND OF THE PROJECT:

Our project's goal is to create, develop, and distribute software that, in our opinion, will assist universities worldwide in promoting a more fruitful and efficient method of student evaluation. As the central concept of our project, we've introduced the notion of Course Outcomes (COs) and Program Learning Outcomes (PLOs), where each CO is mapped to a PLO, and each PLO represents a particular valuable skill that students are expected to acquire or improve at the conclusion of that course, such as problem analysis, design, implementation of a skill and spider chart.
The details will all be present in the course outline for the students to have easy access and have all the necessary details regarding a course. The project will determine whether each student has successfully completed the PLOs that are linked to the COs requirements in order to evaluate them effectively through tools such as spider charts. IEB input is accepted by the system when establishing PLO criteria. For the system to map the COs to PLO appropriately, the faculties then input the COs for each of their students. It was discovered via the execution of this project that the efficiency not only reduced time but also increased quality. The PLOs are carefully and deliberately selected to guarantee that each student gets the most skills out of a course.

We also have the feature where faculties can input the questions in the question bank which can be accessed by the students which will help them gain knowledge on their desired topics and will provide them a vast field to practice.
Students can monitor their progress in each area and identify their areas for growth and improvement. Our program also aims to help the institutional bodies, including faculty, administrative, and departmental bodies, track student development, departmental performance, and better distribute and allocate resources.

OBJECTIVES OF THE PROJECT:

Our project aims to develop an interactive, user-friendly program that will serve as a platform for university staff, faculty, and other participants to assist in enhancing the standard of instruction and revolutionizing how we incorporate technology into our education. We are confident that the information we have gathered, assessed, and organized will open doors for significant improvements in the educational sector as well as the field of computer science. In this situation, SMPS will broaden the project's scope in order to benefit all the departments

SCOPE OF THE PROJECT

Our approach entails building a Web application called SPMS 2 that makes use of a Relational Database Management System (RDMS) to store, edit, add, and update the data required for tracking student performance as well as for producing and archiving related OBE data, reports, and documents. We created hypothetical users for the web based SPMS system and made assumptions about their usage patterns and the information and data they would require. Since issues can occur at many different points throughout all business processes, we will create unique user interfaces and login options for various stakeholders who will also be using this system. Since our data is stored using a (RDBMS), obtaining relevant files, tabular data, and page layouts is made possible and reports become exceedingly simple, enabling real-time interaction with the required data. Additionally, we develop user interfaces that allow all users to quickly access these data and use them to produce download reports, etc. We create a platform through which faculties may work together to create course outline, course reports, marksheets, assessments, map assessments to COs and PLOs for PLO successes, and keep track of student evaluations for all their courses throughout the semester and upload questions in the question bank for the students. The systems for reaching findings are also available to students, the IUB leadership team, and governmental organizations. Each stakeholder will only see the data that is specifically relevant to them, and data will also be protected.

CH-2: REQUIREMENT ANALYSIS

EXISTING BUSINESS SYSTEM (WITH RICH PICTURE)

We are creating a platform through which faculties can work together to create course descriptions, course reports, make assessments, track assessments to COs and PLOs for the success of PLOs, and keep track of student evaluation for all of their courses throughout the semester. This platform is also available to students, the IUB admin and management, and UGC. Each stakeholder will see and monitor the data that specifically relevant to them and the data will also be protected. Students can give responses to their assessments via the platform to their faculties who then can grade the assessments and return. The system receives the assessment records, and it stores them. The system keeps a record of every report. The system offers bar graphs, pie charts and tables that display PLO achievement for all students.

The admin can use the system to update PLO requirements after managements sends them the updated PLO requirements through the system. The admin can also create new users for the system. The registrar’s office also plays a role in the system. Students can ask for grade change to the faculty who in turn can ask for grade change to the registrar’s office. The registrar’s office then sends the change grade. The registrar’s office can also use the system to get general reports and assessment reports about COs and PLOs.

The management are the body of power who updates and sends the PLO requirements. They also deal with governmental organizations like UGC to determine their curriculum and PLOs.
The system offers all users illuminating bar graphs, pie charts, and tables that display PLO achievement for all students, PLO achievement for a specific student, and PLO achievement regarding certain courses. Student responses to questions posed by the faculty are then given back to the faculty. The system receives the assessment records after it has been completed and stores them. The system keeps a record of every report.

The system offers all users illuminating bar graphs, pie charts, and tables that display PLO achievement for all students, PLO achievement for a specific student, and PLO achievement about certain courses.

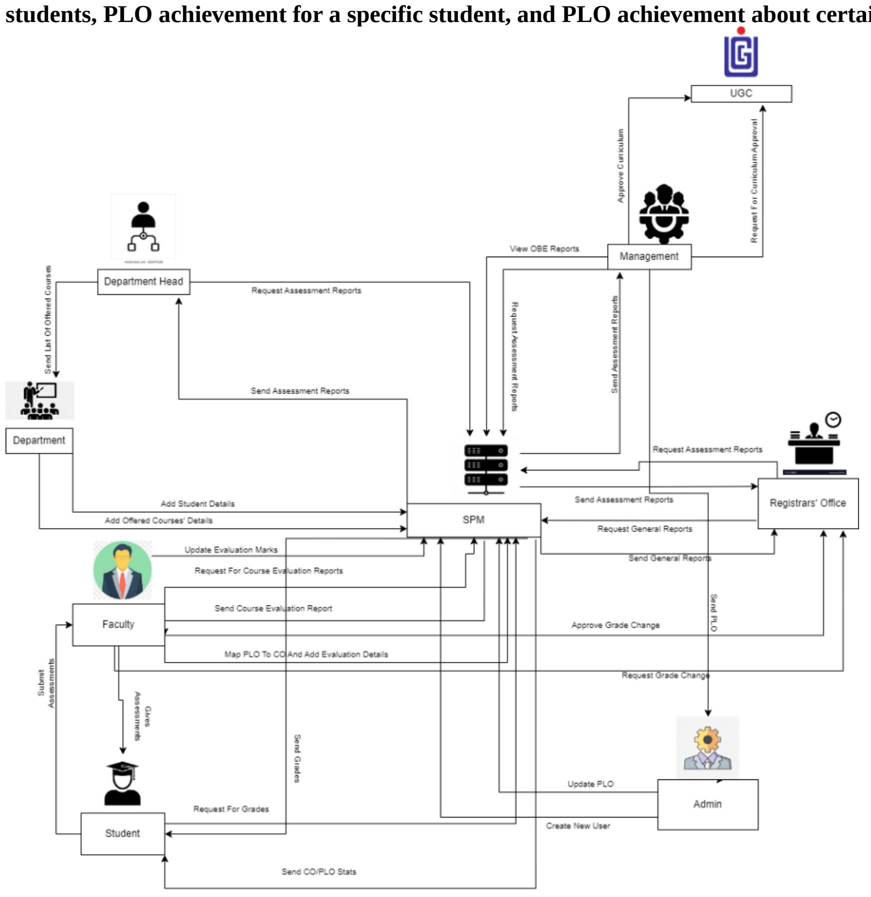


Figure 2: Rich Picture of Existing System

PROCESSES ALONG WITH SIX SYSTEM ELEMENT ANALYSIS

The Six Elements Analysis gives a thorough explanation of each element's function in each process. The table below makes it very evident that human entities predominate all important system operations, particularly the two most important ones—mapping course outcomes and examining documents associated with them. The existing approach, for instance, relies significantly on manually handled and processed hardcopy databases. As a result, there is a considerable amount of waiting involved in the interdependent processes before the Human components may perform their obligations.

Process	**System Roles**					
	Human	**Non-Computing Hardware**	**Computing Hardware**	**Software**	**Database**	**Network and Communication**
Student Registration	**Student:** **a) Search for the website** **b) Goes to the website.** **c) Clicks on the form option.** **d) Fill up the form with required Information.** **Registrar's Office:** **a) Checks and verifies student enrollment information from the forms from the website or hardcopy forms.** **b) Registrar Office's Admin logs into the system using Admin-ID and password.** **c) Sends verified student information as an attachment to Admin/Team.**	Paper and Stationery: **a) Used to collect information about students through enrollment forms.**	**Computer/ Laptop** **a) SPMS admin will use Computers to access and update data.** **b) Users will use the computer to view the data.** **Database Server** **a) Used by SPMS Developers to collect data and maintain the**	**Operating Software** **a) Utilized by Registrar Office and SPMS.** **Student** **a) Uses to fill up the form from the website.** **SPMS** **a) The software for which the administrator will set up user accounts.**	**Register Office Database** **a) Used by the registrar's office to compile student data into an excel file for sending to SPMS.** **SPMS** **a) For any upgrades or new user accounts, information is kept in the database.** **Excel** **a) Data from student accounts**	**Internet** **a) To access and store data to SPMS it is used.** **b) It is used to collect the student form from the student to registrar office.** **c)The Registrar office sends all the student information to SPM admin by using it.**

	Admin: **a) Admin logs into the system using SPMS User-ID and password.** **b) Receives the student enrollment information in the attached files.** **c) Admin updates the student enrollment information in Database.** **d) Notifies respected Stakeholders** **Department Head:** **a) Logs into the system using them User-ID and password.** **b) Inputs the desired time period for number of students enrolled.** **Higher Authority (VC/ Dean):** **a) Logs into the system using their User-ID and password.** **b) Inputs the desired time period and**		**softwar e.** **Networ king Devices (Router, Switch, Bridge, Hub):** **a) Used to access SPMS**		**may be kept in an excel file and used later in SPMS.**	

	compare School/Department for the number of students enrolled accordingly. Faculty: a) logs into the system using Faculty ID and password b) Inputs the ID of the section the faculty is taking to view the students enrolled.					
Student Performance Based on CGPA	**Student: a) Logs into the System using Student-ID and password. b) Inputs the desired time period to view self CGPA Progress. Registrar's Office: a) Logs into the System using User-ID and password. b) Inputs the desired time period and School, Department or program to view Statistically and analyzed CGPA**		**Computer/ Laptop a) User will need a computer to access SPMS Printer a) Used to print out the report if need be. Networking Devices (Router, Switch, Bridge, Hub):**	**Operating Software a) The user uses it to execute SPMS SPMS a) A performance trend will be generated by the software.**	**SPMS Database a) Obtain performance using the database.**	**Internet a) To login into and access the SPMS it is used.**

	trend of students. Department Head: a) Logs into the System using User-ID and password. b) Inputs the desired time period and school, Department or program. c) View statistically analyzed CGPA trend of students or any individual student. Faculty: a) Logs into the system using Faculty-ID and password. b) Inputs the desired time period and program to view statistically and analyzed CGPA trend of students or any individual student those who attended the faculty's Section. Higher Authority: a) Logs into the system using their User-ID		a) Used to access the Internet.			

	and password. b) Inputs the desired time period, School and Department c) View statistically analyzed CGPA trend of students.					
Course-wise student performance based on CGPA	Student: a) Logs into the system using Student-ID and password. b) Inputs the course c) View self GPA for the course. Department Head: a) Logs into the System using User-ID and password. b) Inputs the desired time-period Course-ID c) View statistically analyzed GPA trend of Students. Registrar's office: a) Logs into the System using Admin-ID and password. b) Inputs the desired time -period and coursed c) view		Computer/ Laptop a) User will need a computer to access SPMS Printer a) Used to print out the report if need be. Networking Devices (Router, Switch, Bridge, Hub): a) Used to access the Internet.	SPMS a) A performance trend based on GPA will be generated by the software.	SPMS Database a) Here, the performance will be stored and updated.	Internet a) To login into and access the SPMS it is used.

	statistically analyzed GPA trend of students. **Faculty:** **a) Logs into the System using Faculty-ID and password.** **b) Inputs the desired time period Course-ID under the faculty** **c)view statistically analyzed GPA trend of students who faculty's section.** **Higher Authority:** **a) Logs into the system using their User-ID and password.** **b) Inputs the desired time-period and Course-ID** **c)View statistically analyzed GPA trend of students for that specific course.**					
Selective Number of	**Department Head:**		**Comput er/ Laptop**	SPMS **a) The software will**	**SPMS Database**	**Internet** **a) To login into and**

Instructor-wise student performance based on the GPA	**a) Logs into the system using User-ID and password. b) Inputs the desired time-period Course-ID c)View statistically analyzed GPA trend of students for a selective number of Instructors. Registrar's office: a) Logs into the system using Admin-ID and password. b) Inputs the desired time-period Course-ID c) View statistically analyzed GPA trend of students for a selective number of Instructors Faculty: a) Logs into the system using Faculty-ID and password. b) Inputs the desired time -period & Course-ID c)View statistically**		**a) User will need a computer to access SPMS Printer a) Used to print out the report if need be. Networking Devices (Router, Switch, Bridge, Hub): a) Used to access the Internet.**	**produce a performance trend for a specified instructor.**	**a) Here, the performance will be stored and updated.**	**access the SPMS it is used.**

	analyzed GPA trend of students for a selective number of Instructors. **Higher Authority:** **a) Logs into the System using User-ID and password.** **b) Inputs the desired time-period Course-ID** **c) View statistically analyzed GPA trend of students for a selective number of Instructors.**					
VC-wise, dean-wise, or department head-wise student performance	**Department Head:** **a) Logs into the system using User-ID and password.** **b) Select Input from VC/Dean/Department Head** **c) View the student performance trend as per choice.** **Registrar's office:** **a) Logs into the system using**		**Computer/ Laptop** **a) User will need a computer to access SPMS** **Printer** **a) Used to print out the report if need be.**	**SPMS** **a) The software will produce a performance trend**	**SPMS Database** **a) Here, the performance will be stored.**	**Internet** **a) To login into and access the SPMS it is used.**

	User-ID and password. b) Select Input from VC/Dean/Department Head c) View the student performance trend as per choice. Dean or VC a) Logs into the system using User-ID and password. b) Select Input from VC/Dean/Department Head c) View the student performance trend as per choice.		Networking Devices (Router, Switch, Bridge, Hub): a) Used to access the Internet.			
Instructor-wise student performance based on the GPA of the students	Department Head: a) Logs into the system using Department-ID and Password. b) Inputs a particular instructor Name/ID c)View the student performance trend of selected Instructor. Registrar's office: a) Logs into the system using		Computer/ Laptop a) User will need a computer to access SPMS Printer a) Used to print out the report if need be. Networking Devices (Router, Switch,	SPMS a) The software will produce a performance trend	SPMS Database a) The performance will be stored and updated in the database.	Internet a) To login into and access the SPMS it is used.

	User-ID and password. b) Inputs a particular instructor c) View the student performance trend of selected Instructor. Faculty: a) Logs into the system using User-ID and password. b) Input their Name/ID. c) View the student performance trend. Dean: a) Logs into the system using User-ID and password. b) Inputs a particular instructor c)View the student performance trend of selected instructor VC a) Logs into the system using User-ID and password. b) Inputs a particular		Bridge, Hub): a) Used to access the Internet.			

	instructor c)View the student performance trend of selected instructor.					
Total PLO percentage achieved and attempted by the student along with the departmental average	**Student: a) Logs into the system using Student-ID and Password b) Inputs the time- period c)Views their comparison of attempted vs achieved PLO percentage along with the departmental Average. Department Head: a) Logs into the system using User-ID and Password b) Inputs the time- period c) Views the comparison of students attempted PLO vs achieved PLO percentage along with the departmental average. Registrar's office:**		**Computer/ Laptop a) User will need a computer to access SPMS Printer a) Used to print out the report if need be. Networking Devices (Router, Switch, Bridge, Hub): a) Used to access the Internet.**	**Operating system a) Used by the SPMS SPMS a) A comparison of the attempted vs. achieved PLO as well as the departmental average will be produced by the software.**	**SPMS Database a) Here, the performance will be stored.**	**Internet a) To login into and access the SPM it is used.**

	a) Logs into the system using User-ID and Password b) Inputs the time- period c) Views the comparison of students Attempted PLO vs Achieved PLO percentage along with the departmental average. Faculty: a) Logs into the system using User-ID and Password. b) Inputs the time period. c) Views the comparison of students attempted PLO vs Achieved PLO percentage along with the departmental Average. Dean a) Logs into the system using User ID and Password b) Inputs the time period c) Views the comparison of students Attempted PLO vs					

	achieved PLO percentage along with the departmental average. VC a) Logs into the system using User-ID and Password. b) Inputs the time- period. c) Views the comparison of students attempted PLO vs Achieved PLO percentage along with the departmental average.					
PLO achievem ent	Student: a) Logs into the system using Student-ID And password. b) Selects PLO achievement c) View PLO Achievement. Department Head: a) Logs into the System using user-ID and password. b) Selects PLO achievement c) View PLO Achievement. Registrar's office:		Comput er/ Laptop a) User will need a comput er to access SPMS Printer a) Used to print out the report if need be. Networ king Devices (Router,	SPMS a) A PLO achievement will be generated by the software.	SPMS Database a) Here, the performa nce will be stored and updated.	Internet a) To login into and access the SPM it is used.

	a) Logs into the system using user-ID and password. b) Selects PLO achievement. c) View PLO Achievement. **Faculty:** a) Logs into the System using Faculty-ID and password. b) Selects PLO Achievement. c) View PLO Achievement. **Dean** a) Logs into the System using user-ID and password. b) Selects PLO achievement. c) View PLO Achievement. **VC** a) Logs into the system using user-ID and password. b) Selects PLO achievement. c) View PLO achievement		Switch, Bridge, Hub): a) Used to access the Internet.			
Expected PLO-achievement versus actual	**Student:** a) Logs into the system using Student-ID and password.		**Computer/ Laptop** a) User will	**SPMS** a) A) The software will calculate the expected vs.	**SPMS Database** a) The performance will	**Internet** a) To login into and access the

score (for course's, student's, Department's, program's or school's)	**b) Selects PLO achievement comparison** **c) View PLO achievement Comparison.** **Department Head:** **a) Logs into the system using user-ID and password.** **b) Selects PLO achievement comparison** **c) View PLO achievement Comparison.** **Registrar's office:** **a) Logs into the system using user-ID and password.** **b) Selects PLO achievement comparison.** **c) View PLO achievement comparison.** **Faculty:** **a) Logs into the System using Faculty-ID and password.** **b) Selects PLO achievement comparison.** **c) view PLO Achievement comparison.** **Dean**		**need a computer to access SPMS** **Printer** **a) Used to print out the report if need be.** **Networking Devices (Router, Switch, Bridge, Hub):** **a) Used to access the Internet.**	**achieved PLO.**	**be stored and updated in the database.**	**SPMS it is used.**

	a) Logs into the system using user-ID and password. b) Selects PLO achievement comparison. c) View PLO achievement Comparison. VC a) Logs into the system using user-ID and password. b) Selects PLO achievement comparison c) View PLO achievement Comparison.					
CO-PLO achievem ent summary	**Student: a) Logs into the system using Student-ID and password. b) Selects CO - PLO achievement summary. c) View CO- PLO achievement summary. Department Head: a) Logs into the system using user-ID and password. b) Selects CO -PLO achievement summary. c) View CO - PLO achievement**		**Comput er/ Laptop a) User will need a comput er to access SPMS Printer a) Used to print out the report if need be. Networ king Devices (Router, Switch, Bridge, Hub): a) Used to**	**SPMS a) The software will produce a summary of CO-PLO accomplishm ents.**	**SPMS Database a) The Summar y will be stored and updated in the database .**	**Internet a) To login into and access the SPMS it is used.**

	Summary. **Registrar's office:** **a) Logs into the system using user-ID and password.** **b) Selects CO -PLO achievement summary.** **c) View CO -PLO achievement Summary.** **Faculty:** **a) Logs into the system using Faculty-ID and password.** **b) Selects CO -PLO achievement summary.** **c) View CO - PLO achievement Summary.** **Dean** **a) Logs into the system using user-ID and password.** **b) Selects CO -PLO achievement summary.** **c) View CO - PLO achievement Summary.**		**access the Internet .**			

	VC **a) Logs into the system using user-ID and password.** **b) Selects CO -PLO achievement summary.** **c) view CO - PLO achievement summary.**					

EXISTING PROBLEMS & ANALYSIS OF THE PROBLEM

Process Name	**Stakeholders**	**Concerns (Problems)**	**Analysis (Reason of the Problem)**	**Proposed Solution**
Student Enrollment	1. Student 2. Department Head 3. Registrar's Office 4. Faculty 5. Admin	Comparison of Student who have Enrolled in each Department with respect to a given Time Period/Semester	Student enrolled stat is recorded school-wise, department-wise, and program-wise but was not compared with respect to time period or semesters.	We want to keep the record in the count of students enrolled along with a visual comparison of the student stats as per school-wise, department-wise, program-wise and semester-wise.
Assessments and Grading	1. Faculty 2. Students	1) Condition of Question paper and Answer Script 2) Giving and Receiving Process 3) Unreliable Storage 4) Lack of Visibility of Learning and Question Difficulty 5) No method to Submit Assessments and Grades	1) The question papers and answer script which are being stored physically can get damaged or may get lost. 2) The Process of completing the assessment and giving it to the teacher in person is slow. 3) There may be a shortage of physical space due to increase number of papers. 4) Need to find the domain of	The question papers and answer scripts can be stored into the database so there is no problem of storage. Once a question is placed inside the question bank, the question gets its difficulty level and domain of learning automatically assigned. Online submission of assessment

			learning and difficulty of the question manually and that also takes a lot of time. 5) Adding method to Insert Grades and Cos of a course.	saves time as it negates the necessity to submit a physical copy in person. And Adding Method to Submit Grading and CO assessments by importing CSV file.
Course Outline	1. Department 2. Faculty 3. Student	1) Waiting Delay for receiving Necessary Resources 2) Creating a Course Outline	1) The faculty needs to send requests to department and wait for them to send back the necessary materials. 2) It requires a lot of time to create a course outline manually.	A feature can be installed to generate the course outline automatically according to the things the faculty wants to add. It is stored in the database, and it can be downloaded by the stakeholders in a pdf file.
Student Performance based on CGPA	1. Student 2. Department Head 3. Registrar's Office 4. Faculty	Comparison of Student CGPA between Schools, Departments, Programs and Courses	The CGPA of students can only be observed individually but can be compared between different schools, departments, programs, and courses.	A system should be in place which will allow the stakeholders to analyze the CGPA not only individually but also based on different schools, departments, programs, and courses for a given time or semester.
CO-PLO Achievement	1. Student 2. Faculty 3. Admin	1) PLO Achievement of a Student for each Courses 2) Comparison of PLO Achievement within a Department 3) PLO Achievement Rate and Score 4) Reports based on CO-PLO	1) Students are unable to monitor progress of their PLO achievement for respective courses as it is only available to higher authorities and is done manually 2) The PLO and corresponding CO of all courses a student does is	A system should be implemented which will record the PLO' and COs in the database which will give easier access to the stakeholders. Comparisons regarding PLO achievements can then be made automatically

			never compared with cumulatively along with the departmental average performance. 3) PLO achieved versus attempted, and the actual score is done manually which can be extremely time consuming. 4) Reports based on PLO and CO may not be enough to give a clear picture.	which will save time. Charts can then be generated for better analysis.

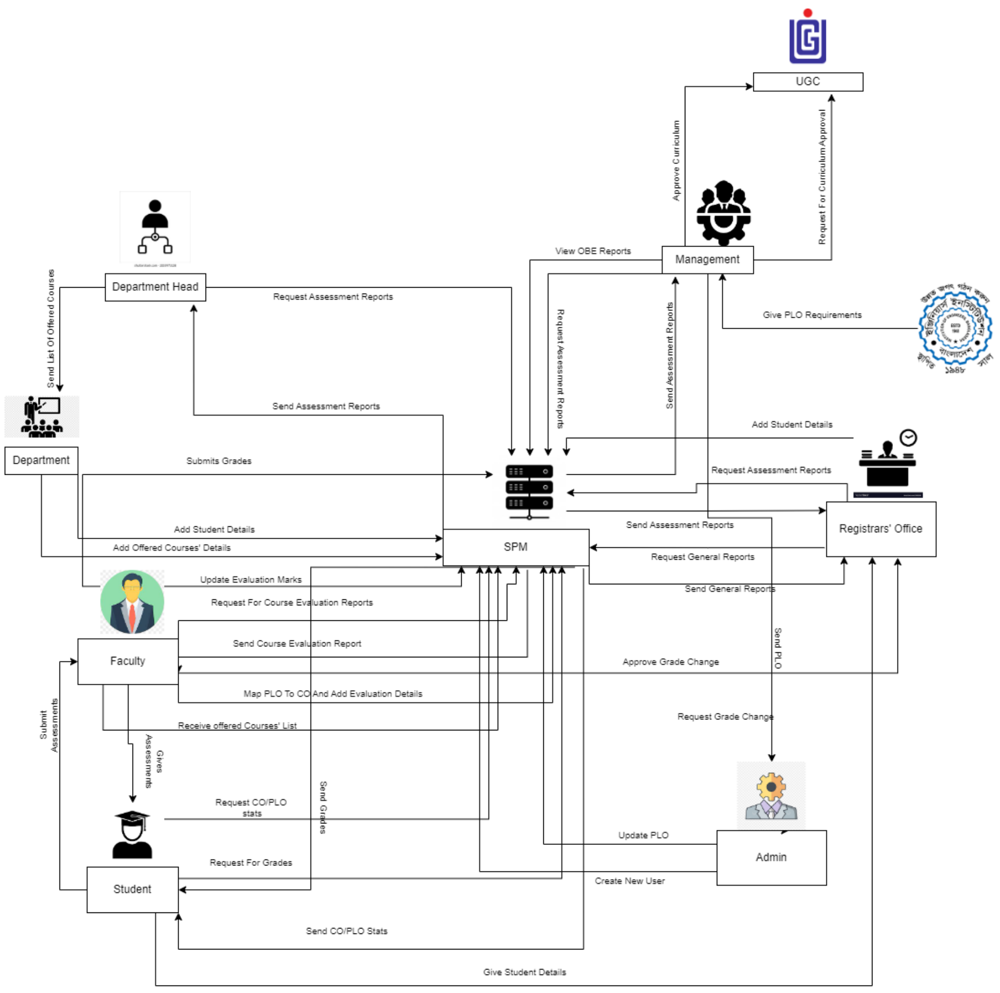
PROPOSED BUSINESS SYSTEM (WITH RICH PICTURE)

The new system will allow the Faculty User to insert CO percentage of a student into the Database by manually or by importing a csv file. The user will be given a text box to type the question. After the Faculty User adds the question, the applications will create an option to view the question. In the question view interface, the user will be able to see the domain of learning along with its level.

The faculty user will be able to Submit Grade of a student of Enrolled Course. Faculty can also submit The Grades and COs of multiple students at a time by importing CSV file.
Faculty User can download course-wise OBE report. Faculty User can check PLO achievement Analysis in Spider Chart Graph of each student by searching Student ID. Faculty User can also get Their Own Departments PLO Achievement Analysis Graph as Spider-Chart graph.

The student users will be able to see Analysis of their own achieved CO's of each graded Courses by searching Course ID and PLOs achievements off that students only in Spider Chart graph.

Bonus: Student User will be able to check their CGPA and Earned credits in their dashboard and can also download their Academic Transcript.



PROPOSED PROCESSES ALONG WITH SIX SYSTEM ELEMENT ANALYSIS

The six elements analysis of the proposed system is a continuation of an analysis process where each analysis is based on the one that comes before it. Based on the rich picture, the role of each element in the new system is further understood in the table below.

Process	**System Roles**					
	Human	**Non-Computing Hardware**	**Computing Hardware**	**Software**	**Database**	**Network and Communication**
Student Registration	Student: a) Search for the website b) Goes to the website. c) Clicks on the form	Paper and Stationery: a) Used to collect information about students through	Computer/ Laptop a) SPMS admin will use Computers to access and update data.	Operating Software a) Utilized by Registrar Office and SPMS Student a) Uses to fill	Register Office Database a) Used by the registrar's office to compile student data into an excel	Internet a) To access and store data to SPMS it is used. b) It is used to collect

	option. c) Fill up the form with required Information. Admin: a) Admin logs into the system using SPMS User-ID and password. b) Receives the student enrollment information in the attached files. c) Admin updates the student enrollment information in Database. d) Inputs the desired time period for number of students enrolled.	enrollment forms.	b) Users will use the computer to view the data. Database Server a) Used by SPMS Developers to collect data and maintain the software. Networking Devices (Router, Switch, Bridge, Hub): a) Used to access SPM S	up the form from the website. SPMS a) The software for which the administrator will set up user accounts.	file for sending to SPMS. SPMS a) For any upgrades or new user accounts, information is kept in the database. Excel a) Data from student accounts may be kept in an excel file and used later in SPMS.	the student form from the student to registrar office. c)The Registrar office sends all the student information to SPMS admin by using it.
Student Performance Based on CGPA	Student: a) Logs into the System using Student-ID and password. b) Inputs the desired time - period to view self CGPA Progress.		Computer/ Laptop a) User will need a computer to access SPMS Printer a) Used to print out the report if need be.	Operating Software a) The user uses it to execute SPMS 2.0 SPMS a) A performance trend will be generated by the software.	SPMS Database a) Obtain performance using the database.	Internet a) To login into and access the SPMS it is used.

	Admin: a) Logs into the System using User-ID and password. b) Inputs the desired time period and School, Department or program to view. Statistically and analyzed. CGPA trend of students. Faculty: a) Logs into the system using Faculty-ID and password. b) Inputs the desired time -period and program to view. statistically and analyzed CGPA trend of students or any individual's student those who attended. the faculty's Section.		Networking Devices (Router, Switch, Bridge, Hub): a) Used to access the Internet.			
Course-wise	Student:		Computer/ Laptop	SPMS	SPMS Database	Internet

student performance based on CGPA	a) Logs into the system using Student-ID and password. b) Inputs the course c) View self GPA for the course. **Admin:** a) Logs into the System using User-ID and password. b) Inputs the desired time-period Course-ID c) View statistically analyzed GPA trend of Students. Faculty: a) Logs into the System using Faculty-ID and password. b) Inputs the desired time - period Course-ID under the faculty c)view statistically analyzed. GPA trend of		a) User will need a computer to access SPMS Printer a) Used to print out the report if need be. Networking Devices (Router, Switch, Bridge, Hub): a) Used to access the Internet.	a) A performance trend based on GPA will be generated by the software.	a) Here, the performance will be stored and updated.	a) To login into and access the SPMS it is used.

	students who faculty’s section.					
Selective Number of Instructor-wise student performance based on the GPA	Admin: a) Logs into the system using User-ID and password. b) Inputs the desired time-period Course-ID c)View statistically analyzed GPA trend of students for a selective number of Instructors. Faculty: a) Logs into the system using Faculty-ID and password. b) Inputs the desired time - period & Course-ID c)View statistically analyzed GPA trend of students for a selective number of Instructors.		Computer/ Laptop a) User will need a computer to access SPMS Printer a) Used to print out the report if need be. Networking Devices (Router, Switch, Bridge, Hub): a) Used to access the Internet.	SPMS a) a) The software will produce a performance trend for a specified instructor.	SPMS Database a) Here, the performance will be stored and updated.	Internet a) To login into and access the SPMS it is used.

	GPA trend of students for a selective number of Instructors.					
Admin wise student performance	Admin: a) Logs into the system using User-ID and password. b) Select Input from from VC/Dean/Department Head c) View the student performance trend as per choice.		Computer/ Laptop a) User will need a computer to access SPMS Printer a) Used to print out the report if need be. Networking Devices (Router, Switch, Bridge, Hub): a) Used to access the Internet.	SPMS a) The software will produce a performance trend	SPMS Database a) Here, the performance will be stored.	Internet a) To login into and access the SPM it is used.
Instructor-wise student performance based on the GPA of the students	Admin: a) Logs into the system using Department-ID and Password. b) Inputs a particular instructor Name/ID c)View the student performance trend of		Computer/ Laptop a) User will need a computer to access SPMS Printer a) Used to print out the report if need be. Networking Devices (Router,	SPMS a) The software will produce a performance trend	SPMS Database a) The performance will be stored. and updated. in the database.	Internet a) To login into and access the SPM it is used.

	selected Instructor. Faculty: a) Logs into the system using User-ID and password. b) Input them Name/ID. c) View the student performanc e trend.		Switch, Bridge, Hub): a) Used to access the Internet.			
Total PLO percentage achieved and attempted by the student along with the departmen tal average	Student: a) Logs into the system using Student-ID and Password b) Inputs the time- period c)Views their comparison of attempted. vs achieved PLO. percentage along with the department al Average. Admin: a) Logs into the system using User-ID and Password b) Inputs the time-period		Computer/ Laptop a) User will need a computer to access SPMS Printer a) Used to print out the report if need be. Networking Devices (Router, Switch, Bridge, Hub): a) Used to access the Internet.	Operating system a) Used by the SPMS SPMS a) A comparison of the attempted vs. achieved PLO as well as the departmental average will be produced by the software.	SPMS Database a) Here, the performance will be stored.	Internet a) To login into and access the SPM it is used.

	c) Views the comparison of students attempted PLO vs achieved PLO percentage along with the departmental average. Faculty: a) Logs into the system using User-ID and Password. b) Inputs the time period. c) Views the comparison of students attempted PLO vs achieved PLO percentage along with the departmental Average.					
PLO achievement	Student: a) Logs into the system using Student-ID and password. b) Selects PLO achievement c) View PLO Achievement.		Computer/ Laptop a) User will need a computer to access SPMS Printer a) Used to print out the report if need be.	SPMS a) A PLO achievement will be generated by the software.	SPMS Database a) Here, the performance will be stored and updated.	Internet a) To login into and access the SPMS it is used.

	Admin: a) Logs into the System using user-ID and password. b) Selects PLO achievement c) View PLO Achievement. Faculty: a) Logs into the System using Faculty-ID and password. b) Selects PLO Achievement. c) View PLO Achievement.		Networking Devices (Router, Switch, Bridge, Hub): a) Used to access the Internet.			
Expected PLO-achievement versus actual score (for course's, student's, Department's, program's or school's)	Student: a) Logs into the system using Student-ID and password. b) Selects PLO achievement comparison c) View PLO achievement Comparison. Admin: a) Logs into the system		Computer/ Laptop a) User will need a computer to access SPMS Printer a) Used to print out the report if need be. Networking Devices (Router, Switch, Bridge, Hub): a) Used to	SPMS a) A) The software will calculate the expected vs. achieved PLO.	SPMS Database a) The performance will be stored and updated in the database.	Internet a) To login into and access the SPMS it is used.

	using user-ID and password. b) Selects PLO achievement comparison c) View PLO achievement Comparison. Faculty: a) Logs into the System using Faculty-ID and password. b) Selects PLO achievement comparison. c) view PLO Achievement comparison.		access the Internet.			
CO-PLO achievement summary	Student: a) Logs into the system using Student-ID and password. b) Selects CO -PLO achievement summary. c) View CO-PLO achievement summary. Admin:		Computer/ Laptop a) User will need a computer to access SPMS Printer a) Used to print out the report if need be. Networking Devices (Router, Switch,	SPMS a) The software will produce a summary of CO-PLO accomplishments.	SPMS Database a) The Summary will be stored and updated in the database.	Internet a) To login into and access the SPMS it is used.

	a) Logs into the system using user-ID and password. b) Selects CO -PLO achievement summary. c) View CO - PLO achievement Summary. Faculty: a) Logs into the system using Faculty-ID and password. b) Selects CO -PLO achievement summary. c) View CO - PLO achievement Summary.		Bridge, Hub): a) Used to access the Internet.			
CO percentage based on the obtained grades for each course summary	Student: a) Logs into the system using Student-ID and password. b) View CO percentage based on the obtained grades for each course summary.		Computer/ Laptop a) User will need a computer to access SPMS Printer a) Used to print out the report if need be**.**	SPMS a) The software will produce a summary of CO percentage based on the obtained grades for each course accomplishments.	SPMS Database a) The Summary will be stored and updated in the database.	Internet a) To login into and access the SPMS it is used.

	Admin: a) Logs into the system using user-ID and password. b) Selects CO percentage based on the obtained grades for each course summary. c) View CO percentage based on the obtained grades for each course summary. Faculty: a) Logs into the system using Faculty-ID and password. b) PLO Achievement Analysis graph of the Faculty's Department.		Networking Devices (Router, Switch, Bridge, Hub): a) Used to access the Internet.			
Checking Cumulative GPA, COs and PLOs	Student: (a) Student Logs into the System. (b)checks the Dashboard Cumulative GPA, Earned Credit, Course-wise COs, PLO	Paper and Stationery: a) Used to collect information about students through enrollment forms.	Computer/ Laptop a) SPMS admin will use Computers to access and update data. b) Users will use the	Operating Software a) Utilized by Registrar Office and SPMS Student a) Uses to fill up the form from the website.	Register Office Database a) Used by the registrar's office to compile student data into an excel file for sending to SPMS.	Internet a) To access and store data to SPMS it is used. b) It is used to collect the student form from the student

	Analysis Graph of that logged in Student in the Dashboard. Faculty: (a)Logs into the system (b) Checks Department Wise CO-PLO achievement graph analysis in the Dashboard. Admin: (a)Logs into the system (b) Checks any department's CO-PLO achievement		computer to view the data. Database Server a) Used by SPMS Developers to collect data and maintain the software. Networking Devices (Router, Switch, Bridge, Hub): a) Used to access SPMS	SPMS a) The software for which the administrator will set up user accounts.	SPMS a) For any upgrades or new user accounts, information is kept in the database. Excel a) Data from student accounts may be kept in an excel file and used later in SPMS.	to registrar office. c)The Registrar office sends all the student information to SPMS admin by using it.
Submitting Grades, COs	Student: (a)Students only participate in their assessments. Faculty: (a)Logs into the system (b)Submits Grades and CO1, CO2, CO3 and CO4 of each student by importing formatted CSV file.	Paper and Stationery: a) Used to collect information about students through enrollment forms.	Computer/ Laptop a) SPMS admin will use Computers to access and update data. b) Users will use the computer to view the data. Database Server a) Used by SPMS Developers to collect data and	Operating Software a) Utilized by Registrar Office and SPMS Student a) Uses to fill up the form from the website. SPMS a) The software for which the administrator will set up user accounts.	Register Office Database a) Used by the registrar's office to compile student data into an excel file for sending to SPMS. SPMS a) For any upgrades or new user accounts, information is kept in the database.	Internet a) To access and store data to SPMS it is used. b) It is used to collect the student form from the student to registrar office. c)The Registrar office sends all the student information to SPMS

	(c)Grades imported successfully. Faculty: (a)Logs into the system (b)Submits Grades and CO1, CO2, CO3 and CO4 of each student by inserting the values in form manually. (c)Grades imported successfully.		maintain the software. Networking Devices (Router, Switch, Bridge, Hub): a) Used to access SPM S		Excel a) Data from student accounts may be kept in an excel file and used later in SPMS.	admin by using it.
Download/ Generate Academic Transcript in PDF, OBE	Student: (a) Student Logs into the System. (b)Download Transcript of that logged in Student User Faculty: (a)Logs into the System (b) generate OBE report of Course (c)OBE report Downloaded successfully. (d) View CSV of OBE report	Paper and Stationery: a) Used to Print the Academic Transcript for the Student's official use	Computer/ Laptop a) SPMS admin will use Computers to access and update data. b) Users will use the computer to view the data. Database Server a) Used by SPMS Developers to collect data and maintain the software. Networking Devices (Router, Switch,	Operating Software a) Utilized by Registrar Office and SPMS Student a) Uses to fill up the form from the website. SPMS a) The software for which the administrator will set up user accounts.	Register Office Database a) Used by the registrar's office to compile student data into an excel file for sending to SPMS. SPMS a) For any upgrades or new user accounts, information is kept in the database. Excel a) Data from student accounts may be kept in an excel file and used later in SPMS.	Internet a) To access and store data to SPMS it is used. b) It is used to collect the student form from the student to registrar office. c)The Registrar office sends all the student information to SPMS admin by using it.

			Bridge, Hub): a) Used to access SPM S			

CH-3 LOGICAL SYSTEM DESIGN

BUSINESS RULES

Business rules describe the operations, definitions and constraints that govern the data model. As opposed to the ERD, they are made using regular English sentences so that a non-technical stakeholder can decipher information about the data model without notation knowledge.

The business rules that govern our data model are as follows:

1. A student must have one department. A STUDENT has StudentID, FirstName, LastName, DateofBirth, Gender, Email, Phone, Address, EnrollmentDate. A department must have many students.

2. Student may perform many Enrollments. An Enrollment includes RegistrationID, Semester, Year, Section Id, StutendID. An specific Enrollment must be performed by one student.

3. A section must mandatorily have many Enrollments. An enrollment has at least one section. A section includes SectionID, SectionNum, CourseId, FacultyID, Semester and Year.

4. Enrollment may belong to many EVALUATIONS. An evaluation mandatorily belongs to one enrollment. An evaluation contains EvaluationID, ObtainedMarks, AssessmentID, RegistrationID.

5. An evaluation must have one assessment. An Assessment must have many evaluations. Assessments contain AssesmentsID, AssessmentName, TotalMarks, SectionID and COID. An assessment must contain one section. A section contains one or many assessments.

6. An assessment must map with one CO’s. A CO’s maps with one or many assessments. A COs includes COID, CourseID and PLOID. A CO must contain one Course. A Course contains one or many CO’s. A course may have many prerequisites. A course must affiliate one mark distribution. A mark distribution may affiliate many courses. A Mark Distribution includes DistID, A, A-, B+, B, B-, C+, C, C-, D+, D, ThresoldMarks.

7. A CO’s must map with one PLO’s. A PLO’s must map with one or many CO’s. PLO includes PLOID, PLONum, Details, ProgramID.

8. A PLO must contain one program. A program contains one or many PLO’s. A program has ProgramID, ProgramName, DepartmentID. A program must contain one or many courses. A Course must contain one course.

9. A Course offered by a Program and has CO1, CO2, CO3, CO4 mapped with PLOs.

Course has CourseID, CourseName, ProgrameID.

10. A program must belong to one department. A department must belong to one or many programs. A department must contain DepartmentID, DepartmentName, SchoolID.

11. A department must contain one school. A School must contain one or many departments. A school includes SchoolID, SchoolName.

12. A User has Three sub-types (Student, Faculty, Admin). A User includes userID, FirstName, LastName, DateofBirth, Gender, Email, Phone, Address, role.

12. A school must run by only one Dean/Admin. A dean/Admin must run one school. A Dean/Admin has SchoolID, StartDate, EndDate.

13. A Department must manage one or many Department head. A department head must manage one department. A department head includes DepartmentID, StartDate, EndDate.

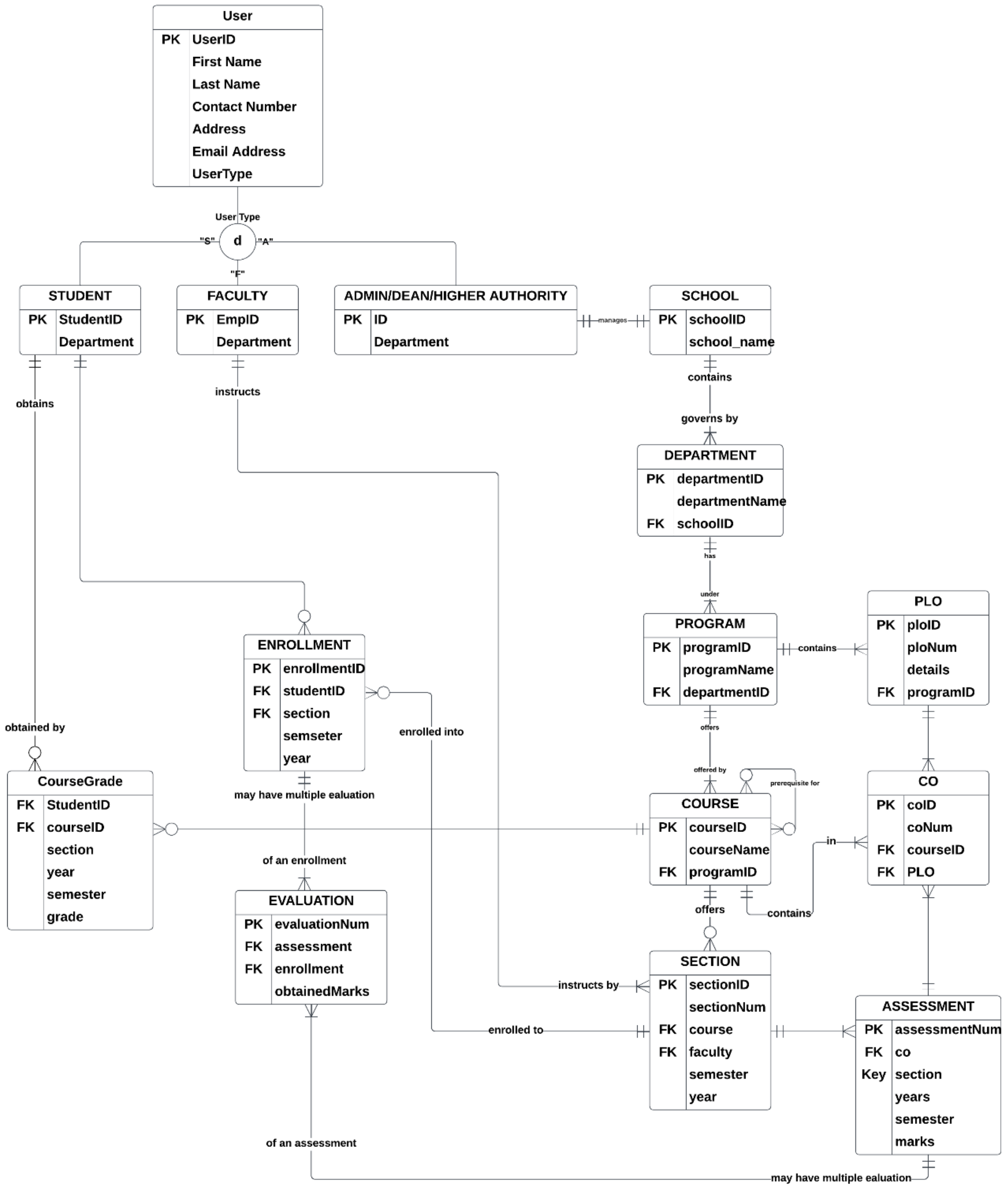
14. A Faculty must have one Department. A department must have one or many Faculties. A Faculty includes facultyID, DepartmentID. A faculty may teach many sections. A section must be taught by one faculty.

15. A PO belongs to exactly one program A program must have one or many PLOs. PLO includes ploID, poNum, details, programID. A PO must belong to one or many CO. A CO must have exactly one PO.

16. A student course performance evaluation is done for Enrollment exactly once. An Enrollment has student course performance evaluation done exactly once. Enrollment has one or many evaluation. An Evaluation has exactly one Enrollment.

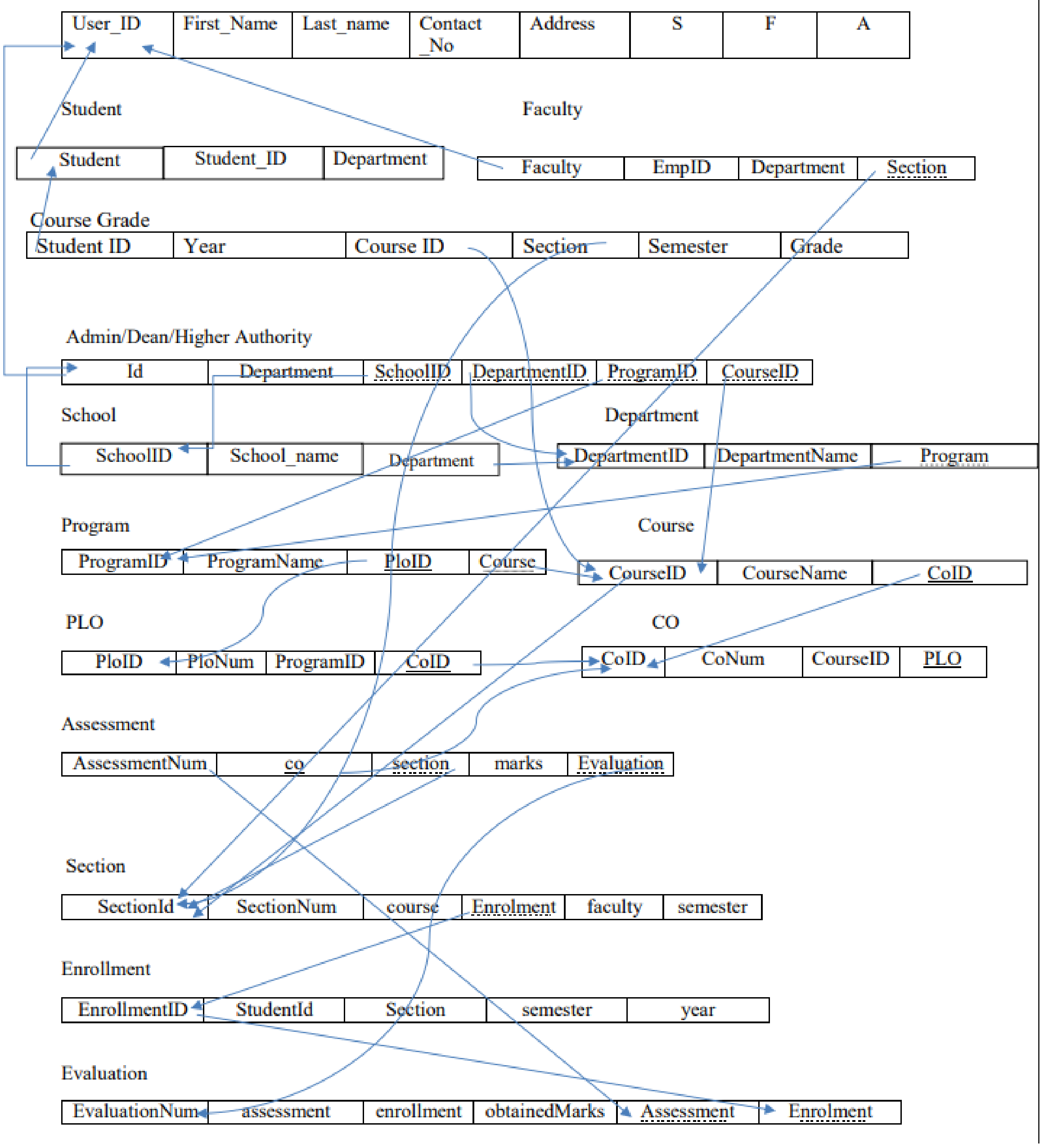
17. A CourseGPA is assigned to a student of a corresponding course which has valid section number. A CourseGPA has StudentID, CourseID, Section, Semester, Year and Grade

ENTITY RELATIONSHIP DIAGRAM (ERD)



ERD TO RELATIONS

Relational Schema



User

User_ID	First_Name	Last_name	Contact _No	Address	S	F	A

Student

Student	Student_ID	Department

Faculty

Faculty	EmpID	Department	Section

Course Grade

Student ID	Year	Course ID	Section	Semester	Grade

Admin/Dean/Higher Authority

Id	Department	SchoolID	DepartmentID	ProgramID	CourseID

School

SchoolID	School_name	Department

Department

DepartmentID	DepartmentName	Program

Program

ProgramID	ProgramName	PloID	Course

Course

CourseID	CourseName	CoID

PLO

PloID	PloNum	ProgramID	CoID

CO

CoID	CoNum	CourseID	PLO

Assessment

AssessmentNum	co	section	marks	Evaluation

Section

SectionId	SectionNum	course	Enrolment	faculty	semester

Enrollment

EnrollmentID	StudentId	Section	semester	year

Evaluation

EvaluationNum	assessment	enrollment	obtainedMarks	Assessment	Enrolment

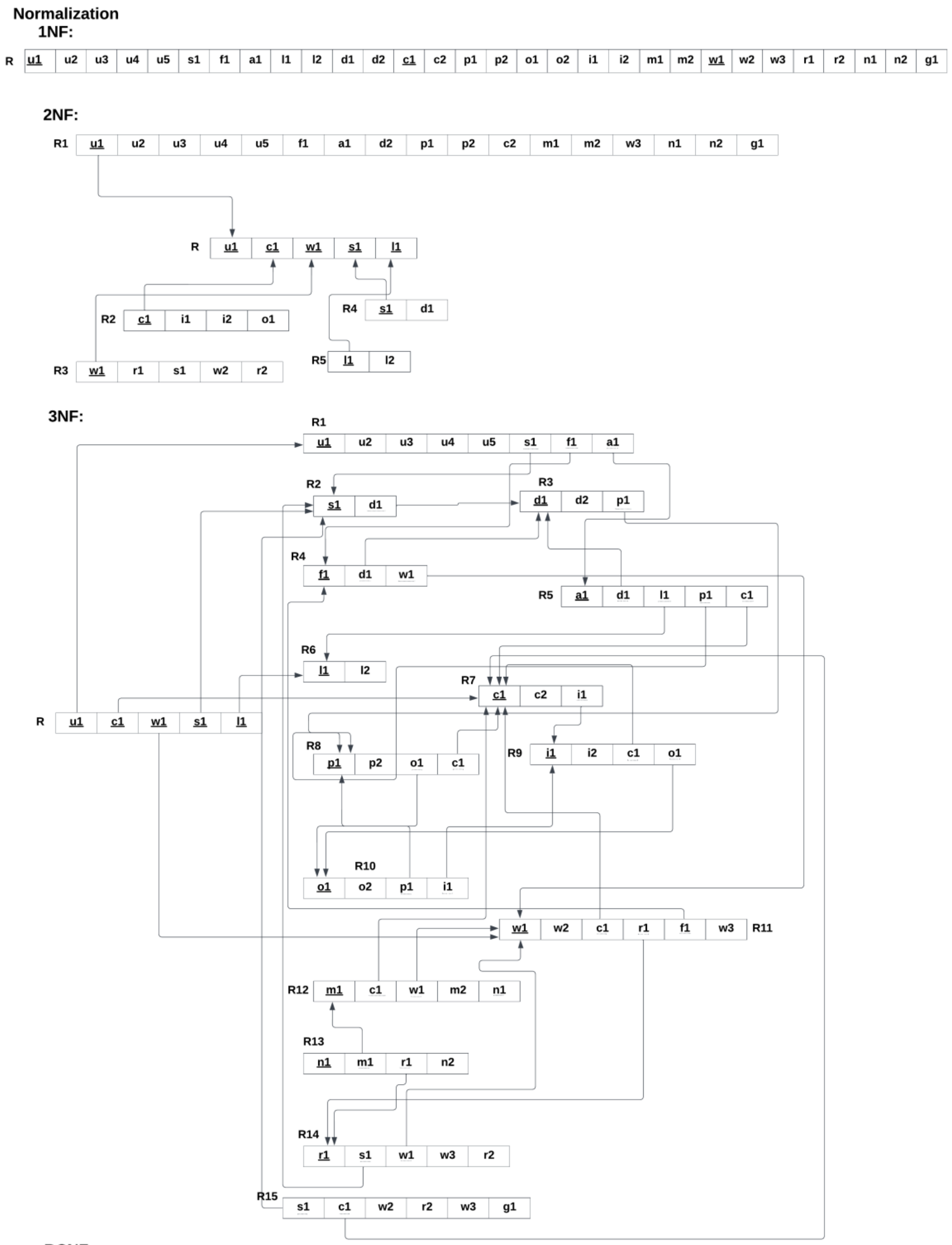
NORMALIZATION

User	User_ID	u1
	First_Name	u2
	Last_Name	u3
	Contact_No	u4
	Address	u5
	Student_ID	s1
	Faculty_ID	f1
	Admin_ID	a1
Student	Student_ID	s1
	Department_ID	d1
Faculty	Faculty_ID	f1
	Department	d1
Admin	Admin_ID	a1
	School_ID	l1
	Department_ID	d1
	Program_ID	p1
	Course_ID	c1
School	School_ID	l1
	School_Name	l2
Department	Department_ID	d1
	Department_Name	d2
	Program	p1
	Course_ID	c1

Program	Program_ID	p1
	Program_Name	p2
	Course_ID	c1
	Plo_ID	o1
PLO	Plo_ID	o1
	Plo_Num	o2
	Program_ID	p1
	Course_ID	c1
CO	Co_ID	i1
	Co_Num	i2
	Couse_ID	c1
	Plo_ID	o1
Assessment	Assessment_Num	m1
	Course_ID	c1
	Section_ID	w1
	Marks	m2
	Evaluation	n1
Section	Section_ID	w1
	Section_Num	w2
	Course_ID	c1
	Enrolment_ID	r1

Course	Course_Name	c2		Faculty_ID	f1
	CO_ID	i1		Semester	w3
Enrollment	Enrollment_ID	r1	Evaluation	Evaluation_Num	n1
	Student_ID	s1		Assessment	m1
	Section_ID	w1		Enrollment_ID	r1
	Semester	w3		Obtain_Marks	n2
	Year	r2			
CourseGPA	Student_ID	s1			
	Course_ID	c1			
	Section_Num	w2			
	Year	r2			
	Semester	w3			
	Grade	g1			

u1	u2,u3,u4,u5,s1,a1,f1	p1	p2,o1,c1
s1	d1	o1	o2,p1,c1
f1	d1,w1	i1	i2,c1,o1
a1	l1,d1,p1,c1	m1	c1,m2,w1,n1
l1	l2	w1	w2,c1,r1,f1,w3
d1	d2,p1	r1	s1,w1,w2,r2
c1	c2,i1	n1	r1,n2
s1,c1	w2,r2,w3,g1		



BCNF

BCNF: All determinants are candidate keys. There is no determinant that is not unique identifier. Here, all the relations already are in BCNF.

DATA DICTIONARY

School_T

Name	**Data Type**	**Size**	**Remarks**
cSchoolID	**VARCHAR**	**10**	**This is the primary key of School. E.g.: “SETS”**
cSchoolName	**VARCHAR**	**255**	**This is the name of the school.** **E.g.: “School of Engineering, Technology & Science”.**

Program_T

Name	Data Type	Size	Remarks
cProgramID	VARCHAR	5	This is the primary key for a program. E.g.: “BSC1”
cProgramName	VARCHAR	255	This is the name of the program. E.g.: “Bachelor of Science”
cDepartmentID	VARCHAR	10	This is the foreign key from the Department table. E.g.: “CSE”

Department_T

Name	Data Type	Size	Remarks
cDepartmentID	VARCHAR	10	This is the primary key for the Department table. E.g.: “CSE”
cDepartmentName	VARCHAR	255	This is the name of the department. E.g.: “Computer Science and Engineering”.
cSchoolID	VARCHAR	10	This is a foreign key from the school table. E.g.: “SETS”.

Course_T

Name	Datatype	Size	Remarks
cCourseID	VARCHAR	7	This is the Primary Key for the Course. E.g.: “CSE203”
cCourseName	VARCHAR	255	This is the name of the Course. E.g.: “Discreet Mathematics”
nCreditNo	INTEGER		This is the number of credits for the Course. E.g.: “3”
cProgramID	VARCHAR	5	This is the Program nme related to the Course. E.g.: “BSC1”
cPrerequisiteCourse	VARCHAR	6	This is the Primary Key for the Course. E.g.: “CSE101”

CLO_T

Name	Data Type	Size	Remarks
nCLOID	INTEGER		This is the primary key for the CLO table. E.g.: “1”.
cCLONum	TEXT		E.g.: “CLO1”.
cPLOID	INT		This is the foreign key from the Program Learning Outcome table. E.g.: “PLO1”
cCourseID	VARCHAR	6	This is the Foreign Key from the Course_T. E.g.: “CSE203”

PLO_T

Name	Datatype	Size	Remarks
nPLOID	INTEGER		This is the primary key for Program Learning Outcome. E.g.: “1”
nPLONum	INTEGER		This is the PLO number. E.g.: “1”
cDetails	VARCHAR	255	This is the details for Program Learning Outcome. E.g.: “An ability to select and apply the knowledge, technique, skills and modern tools of the computer science and engineering discipline”
cProgramID	VARCHAR	5	This is the foreign key from the pPogram_T. E.g.: “BSC1”

Assessment_T

NAME	DataType	Size	Remarks
nAssessmentNo	INTEGER		This is the Primary Key of an assessments Eg:”124”
cMarks	NUMBER		This is the Marks of each assessments Eg:”65.6”
nCLOID	INTEGER		This is the Foreign Key From the CLO_T. E.g.: “1”.
cSectionID	VARCHAR	255	This is the Foreign Key from Section_T. E.g.: “summer23csc10101”

Evaluation_T

Name	Datatype	Size	Remarks
nEvaluationID	INTEGER		This is the Primary Key for Evaluation Table.
cObtainedMarks	NUMBER		This is the obtained marks of the student. E.g.: “24.5”
nAssessmentNo	INTEGER		This is the Foreign Key from Assessment_T Eg:”124”
nEnrollmentID	INTEGER		This is the Foreign Key from Enrollment_T.

Student_T

Name	Data Type	Size	Remarks
nStudentID	INTEGER		This is the primary key for the student table. E.g.: “1921834”.
cFirstName	VARCHAR	30	This is the first name of the student. E.g.: “Rakibul”.
cLastName	VARCHAR	30	This is the last name of the student. E.g.: “Hasan”.
dDateOfBirth	DATE	DD MM YYYY	This is the birth date of the student. E.g.: “21-12-1996”.
cEmail	VARCHAR	30	This is the email of the student. E.g.: “1921834@iub.edu.bd”
nPhone	NUMERIC	11	This is the phone of the student. E.g.: “01XXXXXXXXX”.
cAddress	VARCHAR	50	This is the address of the student. E.g.: “House 1, Road 4, Block D, Bashundhara RA”.
cProgramID	INTEGER		This is the foreign key from the program table. E.g.: “BSc1”
cDepartmentID	VARCHAR	3	This is the foreign key from the Department table. E.g.: “CSE”

Section_T

Name	Datatype	Size	Remarks
cSectionID	VARCHAR	255	This is the Primary Key for Section. E.g.: “summer23csc10101”
nSectionNum	INTEGER		This is the section number. E.g.: “1”

cCourseID	VARCHAR	7	This is the foreign key from the Course table. E.g.: "CSE101"
dYear	YEAR	yyyy	This is the year of registration. E.g.: "2019"
cSemester	VARCHAR	10	This is the semester of the section. E.g.: "Summer"
cFacultyID	NUMERIC	4	This is the foreign key from Faculty table. E.g.: "1801"

Enrollment_T

Name	Datatype	Size	Remarks
nEnrollmentID	INTEGER		This is the Primary Key for Registration. E.g.: "0101010101"
cStudentID	NUMERIC	7	This is the foreign key from Student Table extended from User_T. E.g.: "1830398"
cSemester	VARCHAR	10	This is the semester of registration. E.g.: "Spring"
dYear	YEAR	yyyy	This is the year of registration. E.g.: "2019"
nSectionID	VARCHAR	255	This is the Foreign Key from Section_T. E.g.: "summer23csc10101"

Faculty_T

Name	Datatype	Size	Remarks
nFacultyID	INTEGER		This is the primary key for the faculty table. E.g.: "4250"
dJoinDate	DATE	dd-mm yyyy	This is starting date. E.g.: "01-03-2020"
cRank	VARCHAR	30	This is the rank of the faculty. E.g.: "Assistant Professor"
cDepartmentID	VARCHAR	3	This is the foreign key from the Department table.

			E.g.: “CSE”

Admin_T

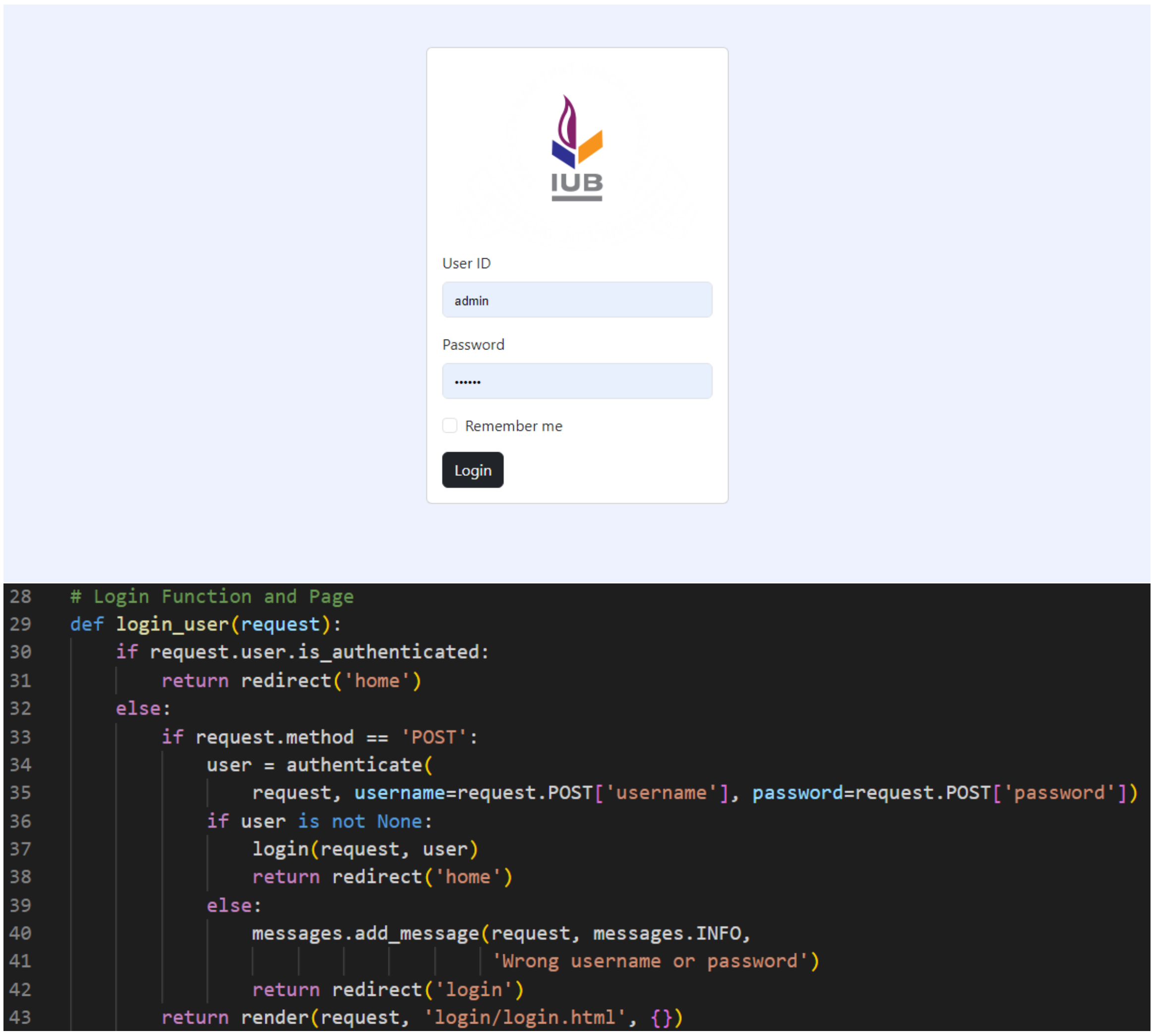
Name	Datatype	Size	Remarks
nAdminID	INTEGER		This is the primary key for the admin table. E.g.: “4250”
cAdminType	VARCHAR	30	This is the type of user logging in E.g.: “VC”
dJoinDate	DATE	dd-mm yyyy	This is starting date. E.g.: “01-03-2020”
cRank	VARCHAR	30	This is the rank of the admin. E.g.: “Assistant Professor”
dEndDate	DATE	dd-mm yyyy	This is the date the admin retires from his post. E.g.: “01-03-2024”
cDepartmentID	VARCHAR	3	This is the foreign key from the Department table. E.g.: “CSE”
cSchoolID	VARCHAR	5	This is a foreign key from the school table. E.g.: “SETS”.

CourseGrade_T

Name	Datatype	Size	Remarks
nID	INTEGER		This is the primary key for the CourseGrade_T table. It increaments automatically E.g.: “4250”
cStudentID	NUMERIC	7	This is the foreign key from Student Table extended from User_T. E.g.: “1830398”
dEduYear	YEAR	yyyy	This is the year of registration or Enrollment.

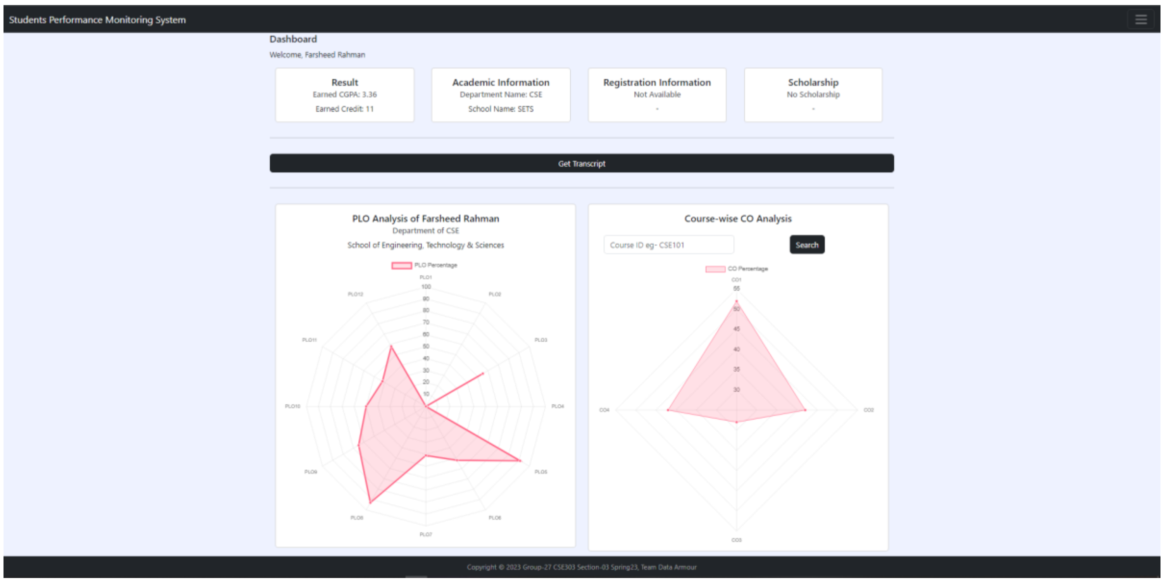
			E.g.: "2019"
cEduSemester	VARCHAR	10	This is the semester of registration or Enrollment. E.g.: "Spring"
cCourseID	VARCHAR	6	This is the foreign key from the Course table. E.g.: "CSE101"
nSectionNum	INTEGER		This is the section number. E.g.: "1"
cGrade	VARCHAR	2	This is the Grade of a course example: "B"

CH-4 PHYSICAL SYSTEM DESIGN



```python
# Login Function and Page
def login_user(request):
    if request.user.is_authenticated:
        return redirect('home')
    else:
        if request.method == 'POST':
            user = authenticate(
                request, username=request.POST['username'], password=request.POST['password'])
            if user is not None:
                login(request, user)
                return redirect('home')
            else:
                messages.add_message(request, messages.INFO,
                                     'Wrong username or password')
                return redirect('login')
        return render(request, 'login/login.html', {})
```

Figure: Sign in Form for all user with Backend Authentication code


Students Performance Monitoring System

Dashboard

Welcome, Farsheed Rahman

Result
Earned CGPA: 3.36
Earned Credit: 11

Academic Information
Department Name: CSE
School Name: SETS

Registration Information
Not Available
-

Scholarship
No Scholarship
-

Get Transcript

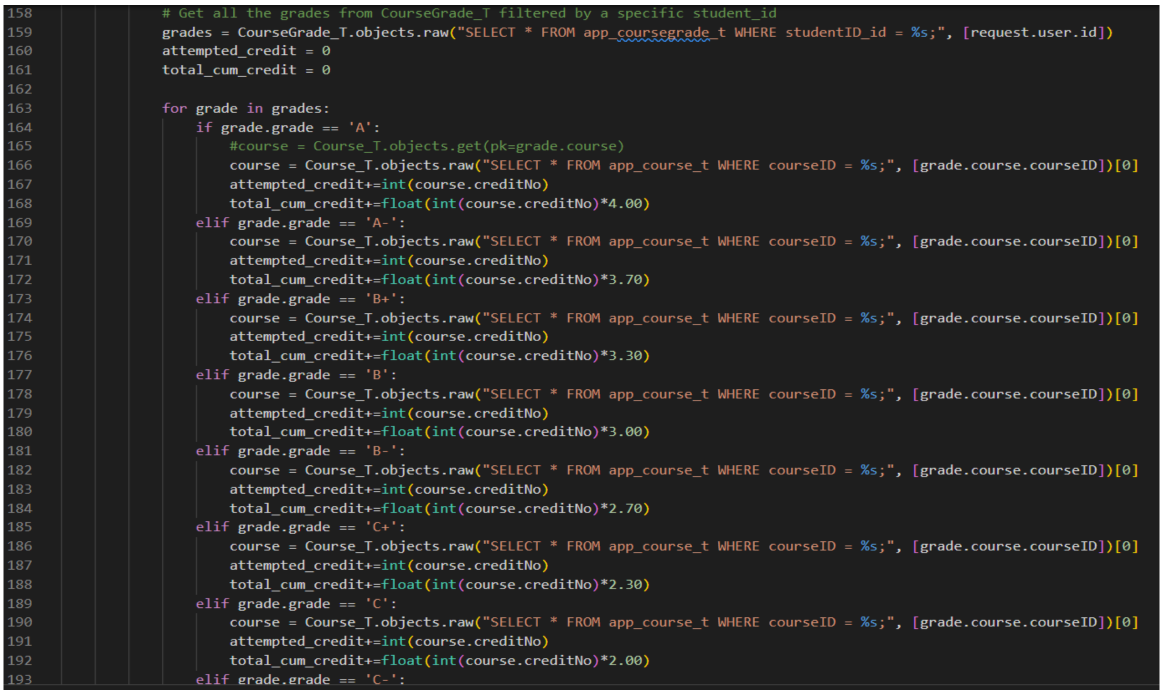
PLO Analysis of Farsheed Rahman
Department of CSE
School of Engineering, Technology & Sciences

Course-wise CO Analysis

Course ID eg- CSE101

Search

Copyright © 2023 Group-27 CSE303 Section-03 Spring23, Team Data Armour


```
    # Get all the grades from CourseGrade_T filtered by a specific student_id
    grades = CourseGrade_T.objects.raw("SELECT * FROM app_coursegrade_t WHERE studentID_id = %s;", [request.user.id])
    attempted_credit = 0
    total_cum_credit = 0

    for grade in grades:
        if grade.grade == 'A':
            #course = Course_T.objects.get(pk=grade.course)
            course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
            attempted_credit+=int(course.creditNo)
            total_cum_credit+=float(int(course.creditNo)*4.00)
        elif grade.grade == 'A-':
            course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
            attempted_credit+=int(course.creditNo)
            total_cum_credit+=float(int(course.creditNo)*3.70)
        elif grade.grade == 'B+':
            course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
            attempted_credit+=int(course.creditNo)
            total_cum_credit+=float(int(course.creditNo)*3.30)
        elif grade.grade == 'B':
            course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
            attempted_credit+=int(course.creditNo)
            total_cum_credit+=float(int(course.creditNo)*3.00)
        elif grade.grade == 'B-':
            course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
            attempted_credit+=int(course.creditNo)
            total_cum_credit+=float(int(course.creditNo)*2.70)
        elif grade.grade == 'C+':
            course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
            attempted_credit+=int(course.creditNo)
            total_cum_credit+=float(int(course.creditNo)*2.30)
        elif grade.grade == 'C':
            course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
            attempted_credit+=int(course.creditNo)
            total_cum_credit+=float(int(course.creditNo)*2.00)
        elif grade.grade == 'C-':
```

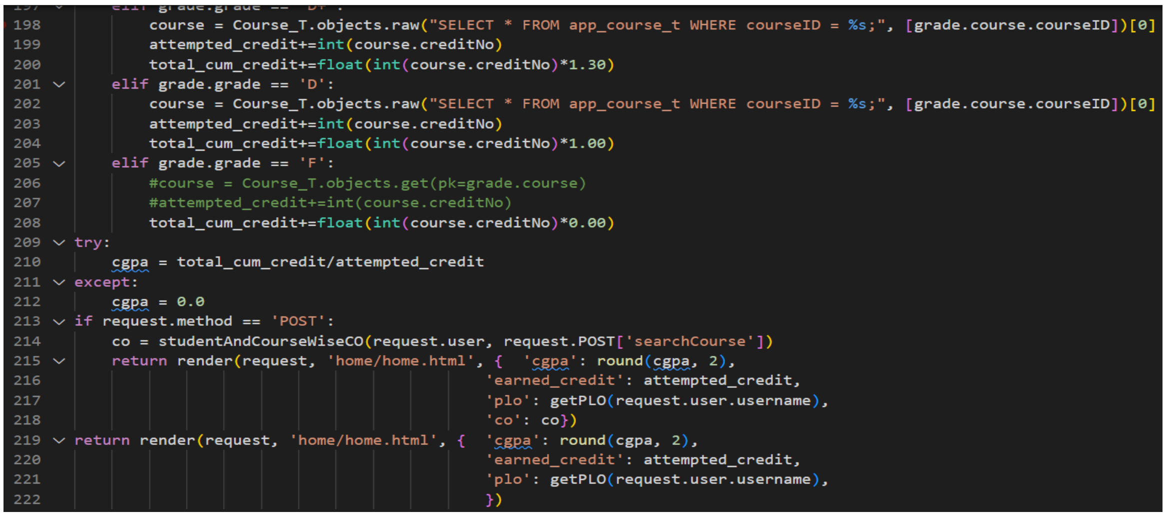
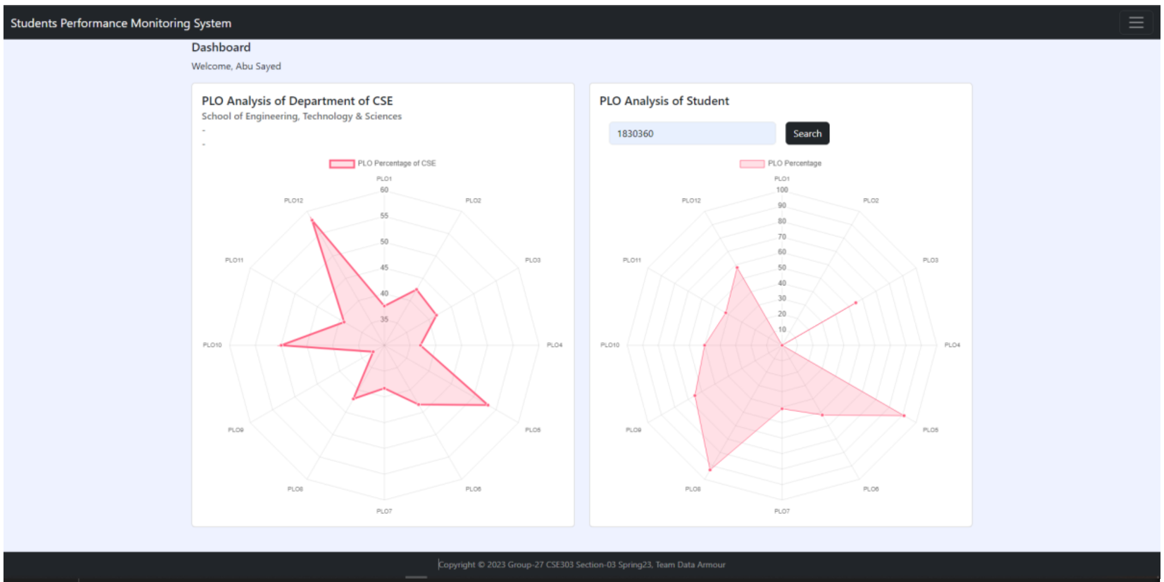

```
            course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
            attempted_credit+=int(course.creditNo)
            total_cum_credit+=float(int(course.creditNo)*1.30)
        elif grade.grade == 'D':
            course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
            attempted_credit+=int(course.creditNo)
            total_cum_credit+=float(int(course.creditNo)*1.00)
        elif grade.grade == 'F':
            #course = Course_T.objects.get(pk=grade.course)
            #attempted_credit+=int(course.creditNo)
            total_cum_credit+=float(int(course.creditNo)*0.00)
    try:
        cgpa = total_cum_credit/attempted_credit
    except:
        cgpa = 0.0
    if request.method == 'POST':
        co = studentAndCourseWiseCO(request.user, request.POST['searchCourse'])
        return render(request, 'home/home.html', {  'cgpa': round(cgpa, 2),
                                                    'earned_credit': attempted_credit,
                                                    'plo': getPLO(request.user.username),
                                                    'co': co})
    return render(request, 'home/home.html', {  'cgpa': round(cgpa, 2),
                                                'earned_credit': attempted_credit,
                                                'plo': getPLO(request.user.username),
                                                })
```

Figure: Student Dashboard Navbar to show the CGPA, earned credit, PLO analysis graph with Backend code


Students Performance Monitoring System

Dashboard

Welcome, Abu Sayed

PLO Analysis of Department of CSE

School of Engineering, Technology & Sciences

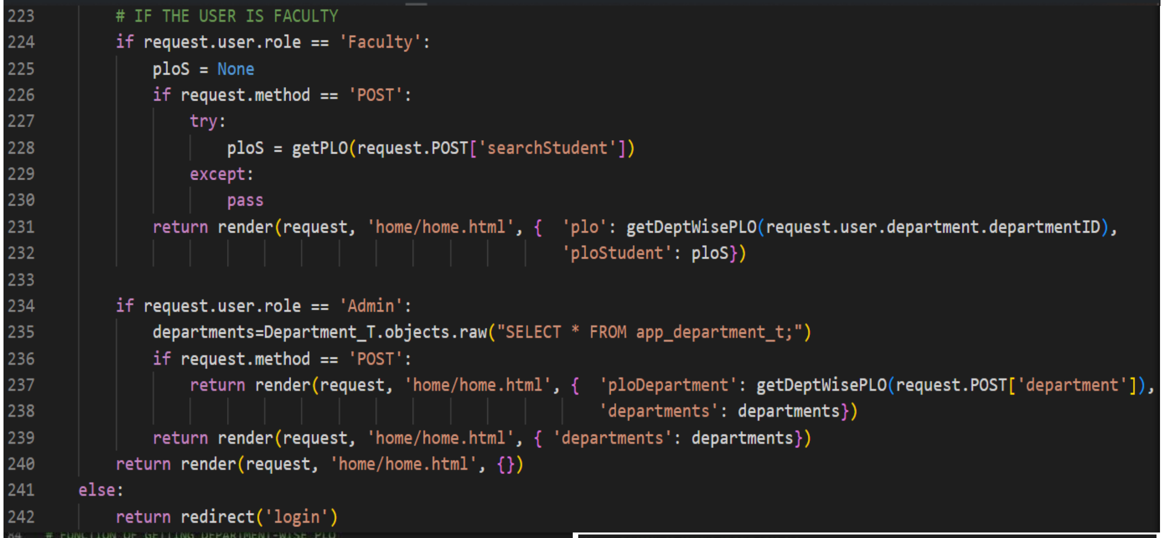
-

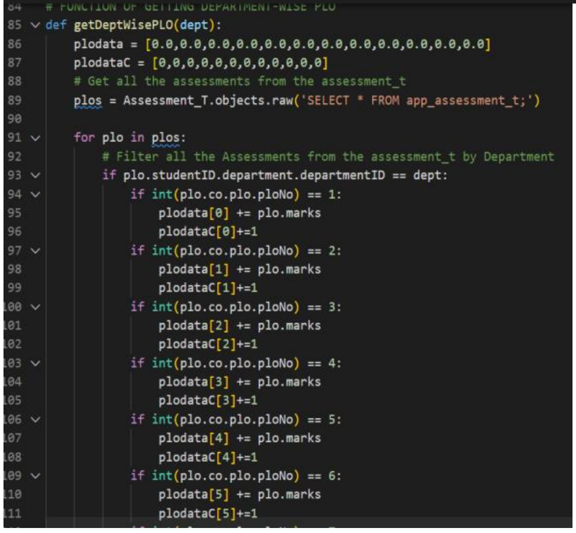
-

PLO Analysis of Student

1830360

Copyright © 2023 Group-27 CSE303 Section-03 Spring23, Team Data Armour


```python
    # IF THE USER IS FACULTY
    if request.user.role == 'Faculty':
        ploS = None
        if request.method == 'POST':
            try:
                ploS = getPLO(request.POST['searchStudent'])
            except:
                pass
        return render(request, 'home/home.html', {  'plo': getDeptWisePLO(request.user.department.departmentID),
                                                    'ploStudent': ploS})

    if request.user.role == 'Admin':
        departments=Department_T.objects.raw("SELECT * FROM app_department_t;")
        if request.method == 'POST':
            return render(request, 'home/home.html', {  'ploDepartment': getDeptWisePLO(request.POST['department']),
                                                        'departments': departments})
        return render(request, 'home/home.html', { 'departments': departments})
    return render(request, 'home/home.html', {})
else:
    return redirect('login')
```


```python
# FUNCTION OF GETTING DEPARTMENT-WISE PLO
def getDeptWisePLO(dept):
    plodata = [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
    plodataC = [0,0,0,0,0,0,0,0,0,0,0,0]
    # Get all the assessments from the assessment_t
    plos = Assessment_T.objects.raw('SELECT * FROM app_assessment_t;')

    for plo in plos:
        # Filter all the Assessments from the assessment_t by Department
        if plo.studentID.department.departmentID == dept:
            if int(plo.co.plo.ploNo) == 1:
                plodata[0] += plo.marks
                plodataC[0]+=1
            if int(plo.co.plo.ploNo) == 2:
                plodata[1] += plo.marks
                plodataC[1]+=1
            if int(plo.co.plo.ploNo) == 3:
                plodata[2] += plo.marks
                plodataC[2]+=1
            if int(plo.co.plo.ploNo) == 4:
                plodata[3] += plo.marks
                plodataC[3]+=1
            if int(plo.co.plo.ploNo) == 5:
                plodata[4] += plo.marks
                plodataC[4]+=1
            if int(plo.co.plo.ploNo) == 6:
                plodata[5] += plo.marks
                plodataC[5]+=1
```

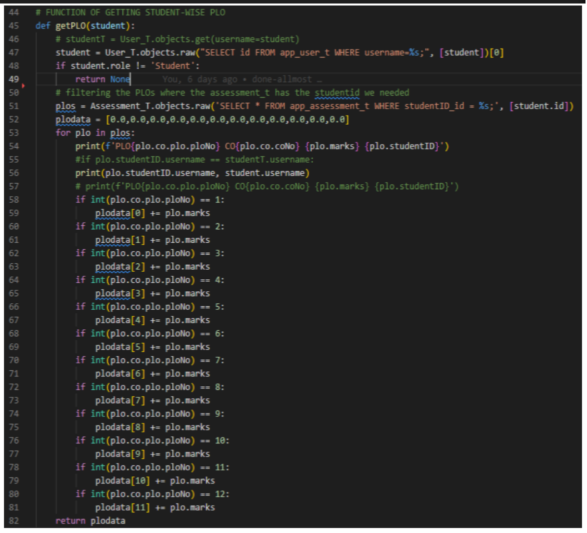
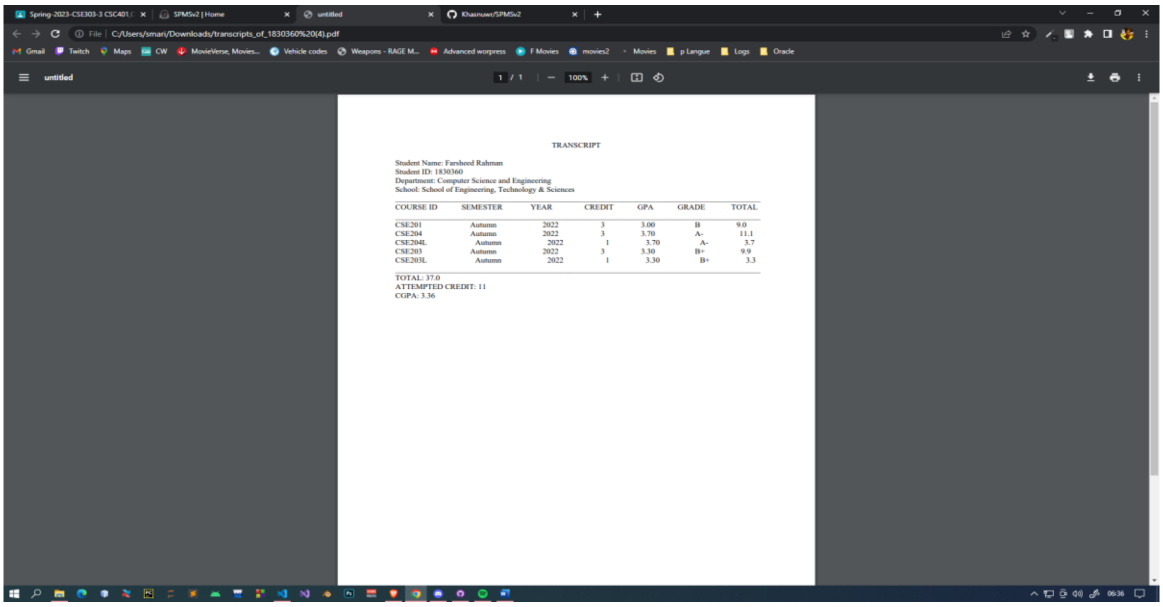

```python
# FUNCTION OF GETTING STUDENT-WISE PLO
def getPLO(student):
    # studentT = User_T.objects.get(username=student)
    student = User_T.objects.raw("SELECT id FROM app_user_t WHERE username=%s;", [student])[0]
    if student.role != 'Student':
        return None        You, 6 days ago • done-allmost …
    # filtering the PLOs where the assessment_t has the studentid we needed
    plos = Assessment_T.objects.raw('SELECT * FROM app_assessment_t WHERE studentID_id = %s;', [student.id])
    plodata = [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
    for plo in plos:
        print(f'PLO{plo.co.plo.ploNo} CO{plo.co.coNo} {plo.marks} {plo.studentID}')
        #if plo.studentID.username == studentT.username:
        print(plo.studentID.username, student.username)
        # print(f'PLO{plo.co.plo.ploNo} CO{plo.co.coNo} {plo.marks} {plo.studentID}')
        if int(plo.co.plo.ploNo) == 1:
            plodata[0] += plo.marks
        if int(plo.co.plo.ploNo) == 2:
            plodata[1] += plo.marks
        if int(plo.co.plo.ploNo) == 3:
            plodata[2] += plo.marks
        if int(plo.co.plo.ploNo) == 4:
            plodata[3] += plo.marks
        if int(plo.co.plo.ploNo) == 5:
            plodata[4] += plo.marks
        if int(plo.co.plo.ploNo) == 6:
            plodata[5] += plo.marks
        if int(plo.co.plo.ploNo) == 7:
            plodata[6] += plo.marks
        if int(plo.co.plo.ploNo) == 8:
            plodata[7] += plo.marks
        if int(plo.co.plo.ploNo) == 9:
            plodata[8] += plo.marks
        if int(plo.co.plo.ploNo) == 10:
            plodata[9] += plo.marks
        if int(plo.co.plo.ploNo) == 11:
            plodata[10] += plo.marks
        if int(plo.co.plo.ploNo) == 12:
            plodata[11] += plo.marks
    return plodata
```

Figure: Dashboard of Faculty User with PLO Analysis Graph Department wise and Searching PLO Analysis Graph Student wise by searching student ID

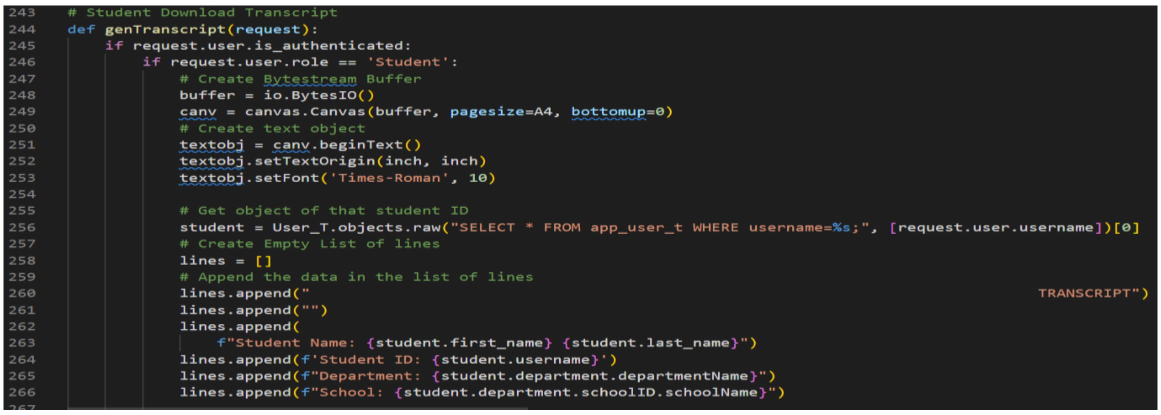


TRANSCRIPT

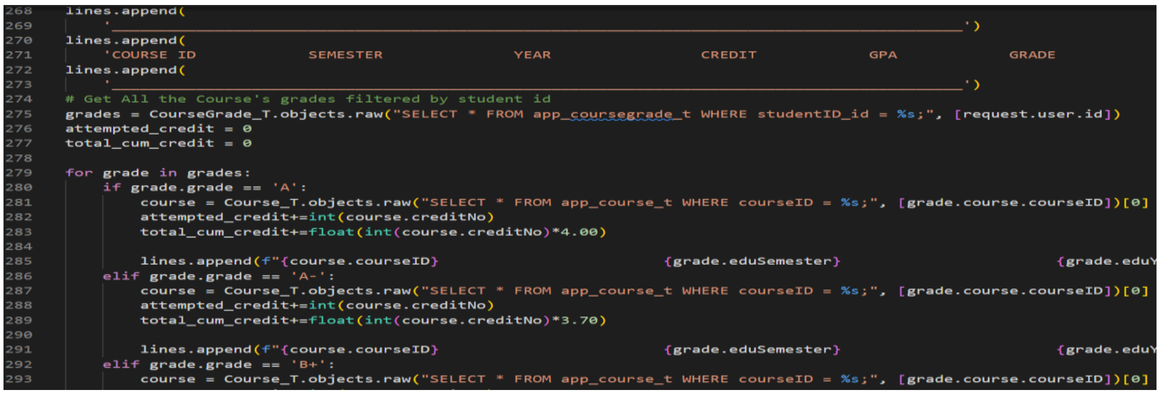
Student Name: Farsheed Rahman
Student ID: 1830360
Department: Computer Science and Engineering
School: School of Engineering, Technology & Sciences

COURSE ID	SEMESTER	YEAR	CREDIT	GPA	GRADE	TOTAL
CSE201	Autumn	2022	3	3.00	B	9.0
CSE204	Autumn	2022	3	3.70	A-	11.1
CSE204L	Autumn	2022	1	3.70	A-	3.7
CSE203	Autumn	2022	3	3.30	B+	9.9
CSE203L	Autumn	2022	1	3.30	B+	3.3

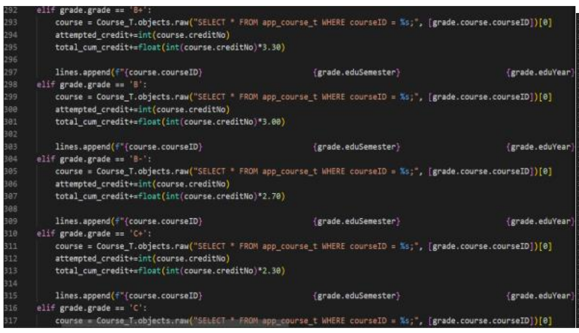
TOTAL: 37.0
ATTEMPTED CREDIT: 11
CGPA: 3.36



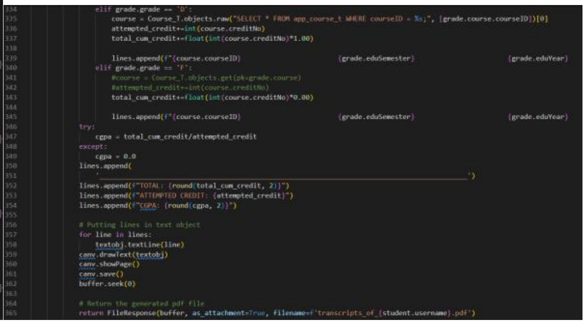
```python
# Student Download Transcript
def genTranscript(request):
    if request.user.is_authenticated:
        if request.user.role == 'Student':
            # Create Bytestream Buffer
            buffer = io.BytesIO()
            canv = canvas.Canvas(buffer, pagesize=A4, bottomup=0)
            # Create text object
            textobj = canv.beginText()
            textobj.setTextOrigin(inch, inch)
            textobj.setFont('Times-Roman', 10)

            # Get object of that student ID
            student = User_T.objects.raw("SELECT * FROM app_user_t WHERE username=%s;", [request.user.username])[0]
            # Create Empty List of lines
            lines = []
            # Append the data in the list of lines
            lines.append("                                                                            TRANSCRIPT")
            lines.append("")
            lines.append(
                f"Student Name: {student.first_name} {student.last_name}")
            lines.append(f'Student ID: {student.username}')
            lines.append(f"Department: {student.department.departmentName}")
            lines.append(f"School: {student.department.schoolID.schoolName}")
```



```python
            lines.append(
                '________________________________________________________________________________________')
            lines.append(
                'COURSE ID                SEMESTER                YEAR                CREDIT            GPA            GRADE
            lines.append(
                '________________________________________________________________________________________')
            # Get All the Course's grades filtered by student id
            grades = CourseGrade_T.objects.raw("SELECT * FROM app_coursegrade_t WHERE studentID_id = %s;", [request.user.id])
            attempted_credit = 0
            total_cum_credit = 0

            for grade in grades:
                if grade.grade == 'A':
                    course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
                    attempted_credit+=int(course.creditNo)
                    total_cum_credit+=float(int(course.creditNo)*4.00)

                    lines.append(f"{course.courseID}                        {grade.eduSemester}                    {grade.eduY
                elif grade.grade == 'A-':
                    course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
                    attempted_credit+=int(course.creditNo)
                    total_cum_credit+=float(int(course.creditNo)*3.70)

                    lines.append(f"{course.courseID}                        {grade.eduSemester}                    {grade.eduY
                elif grade.grade == 'B+':
                    course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
```

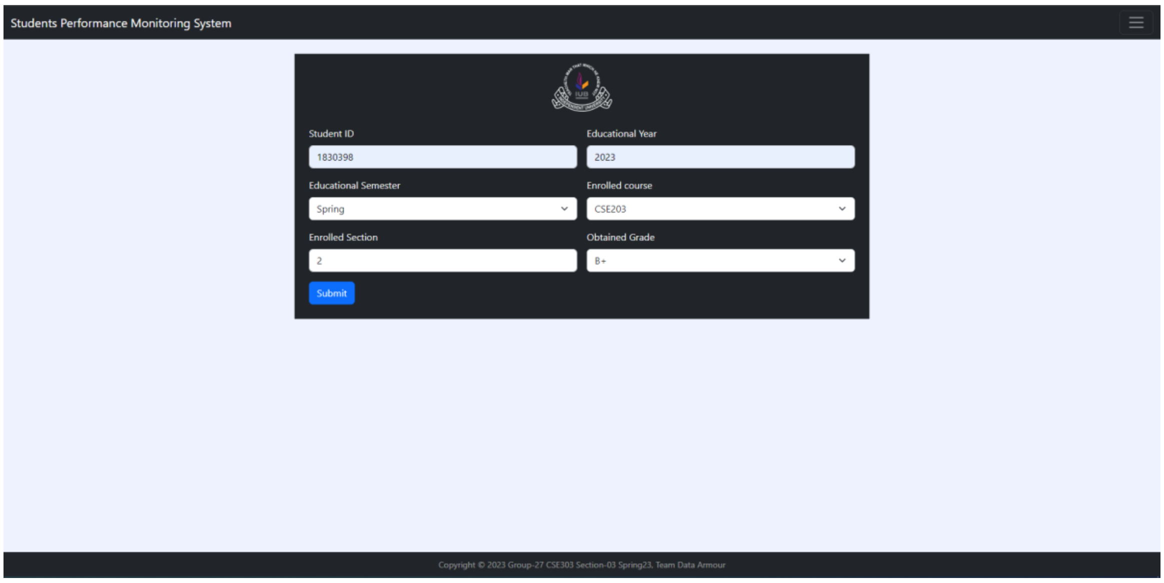


```python
                elif grade.grade == 'B+':
                    course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
                    attempted_credit+=int(course.creditNo)
                    total_cum_credit+=float(int(course.creditNo)*3.30)

                    lines.append(f"{course.courseID}                    {grade.eduSemester}                    {grade.eduYear}
                elif grade.grade == 'B':
                    course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
                    attempted_credit+=int(course.creditNo)
                    total_cum_credit+=float(int(course.creditNo)*3.00)

                    lines.append(f"{course.courseID}                    {grade.eduSemester}                    {grade.eduYear}
                elif grade.grade == 'B-':
                    course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
                    attempted_credit+=int(course.creditNo)
                    total_cum_credit+=float(int(course.creditNo)*2.70)

                    lines.append(f"{course.courseID}                    {grade.eduSemester}                    {grade.eduYear}
                elif grade.grade == 'C+':
                    course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
                    attempted_credit+=int(course.creditNo)
                    total_cum_credit+=float(int(course.creditNo)*2.30)

                    lines.append(f"{course.courseID}                    {grade.eduSemester}                    {grade.eduYear}
                elif grade.grade == 'C':
                    course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
```



```python
                elif grade.grade == 'D':
                    course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
                    attempted_credit+=int(course.creditNo)
                    total_cum_credit+=float(int(course.creditNo)*1.00)

                    lines.append(f"{course.courseID}                    {grade.eduSemester}                    {grade.eduYear}
                elif grade.grade == 'F':
                    #course = Course_T.objects.get(pk=grade.course)
                    #attempted_credit+=int(course.creditNo)
                    total_cum_credit+=float(int(course.creditNo)*0.00)

                    lines.append(f"{course.courseID}                    {grade.eduSemester}                    {grade.eduYear}
            try:
                cgpa = total_cum_credit/attempted_credit
            except:
                cgpa = 0.0
            lines.append(
                '________________________________________________________________________________________')
            lines.append(f"TOTAL: {round(total_cum_credit, 2)}")
            lines.append(f"ATTEMPTED CREDIT: {attempted_credit}")
            lines.append(f"CGPA: {round(cgpa, 2)}")

            # Putting lines in text object
            for line in lines:
                textobj.textLine(line)
            canv.drawText(textobj)
            canv.showPage()
            canv.save()
            buffer.seek(0)

            # Return the generated pdf file
            return FileResponse(buffer, as_attachment=True, filename=f'transcripts_of_{student.username}.pdf')
```

Figure: Academic Transcript Student wise for Student users with backend code

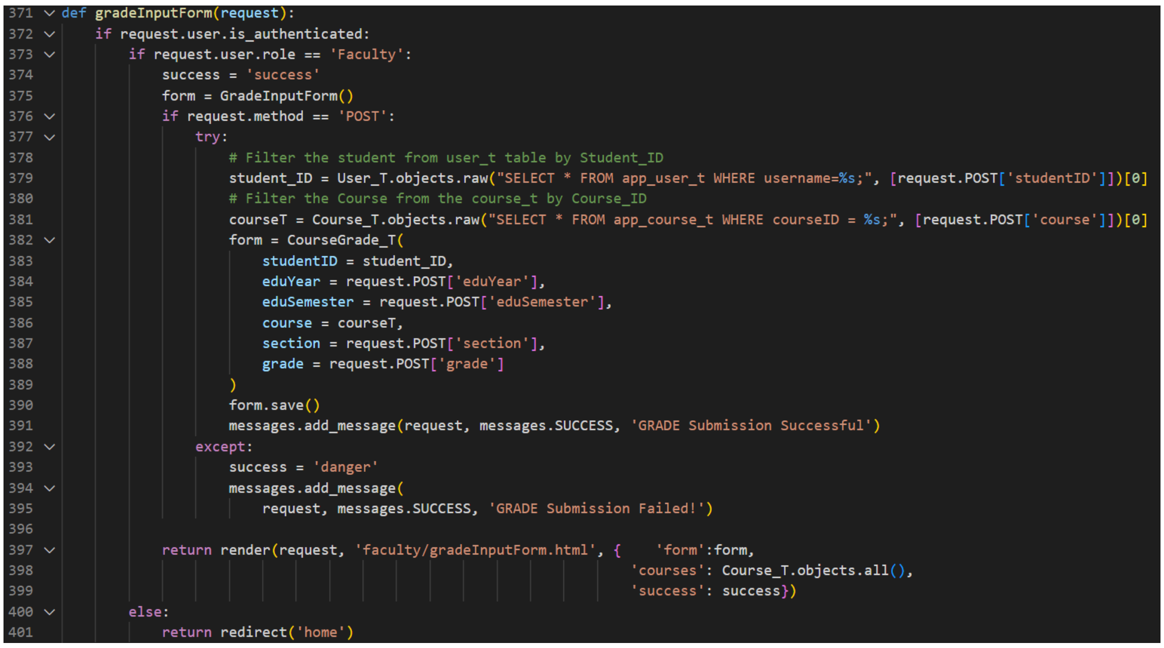


Students Performance Monitoring System

Student ID: 1830398
Educational Year: 2023
Educational Semester: Spring
Enrolled course: CSE203
Enrolled Section: 2
Obtained Grade: B+

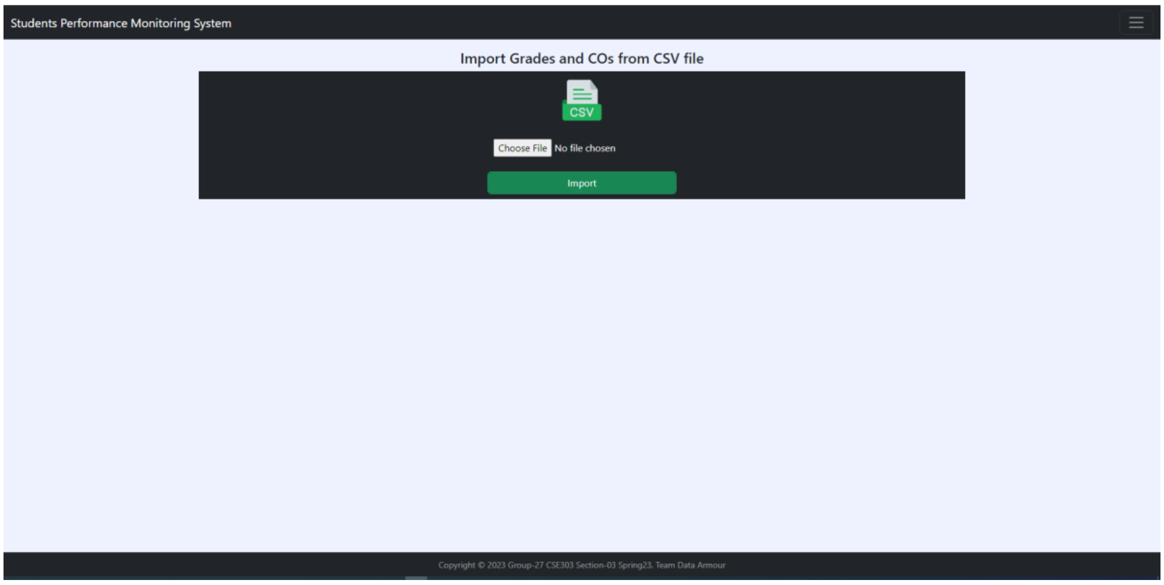
Submit

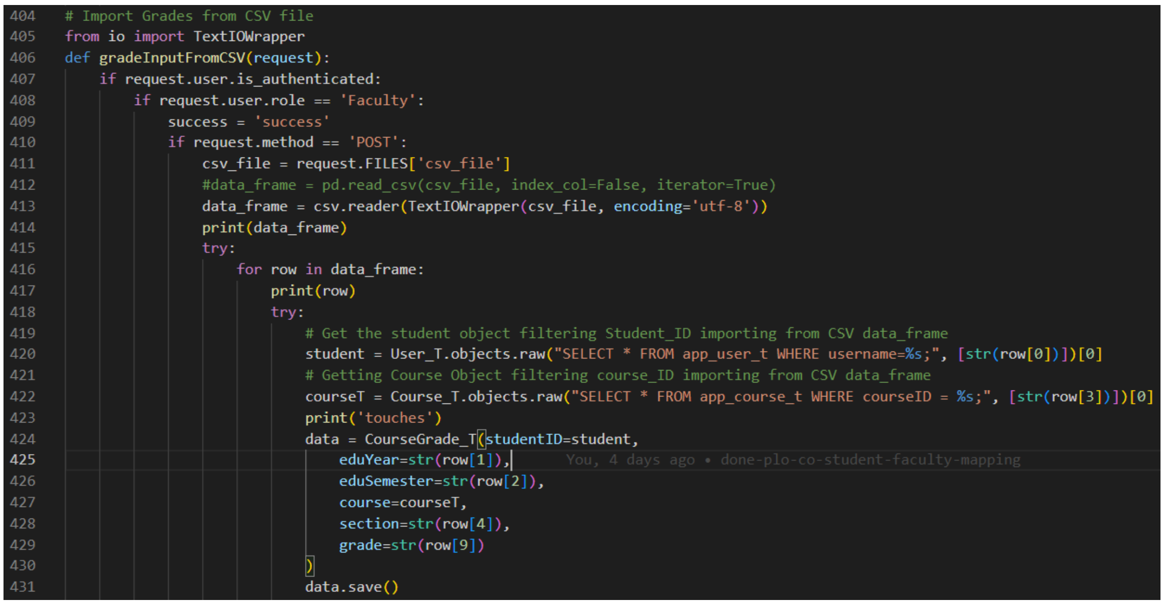
Copyright © 2023 Group-27 CSE303 Section-03 Spring23, Team Data Armour



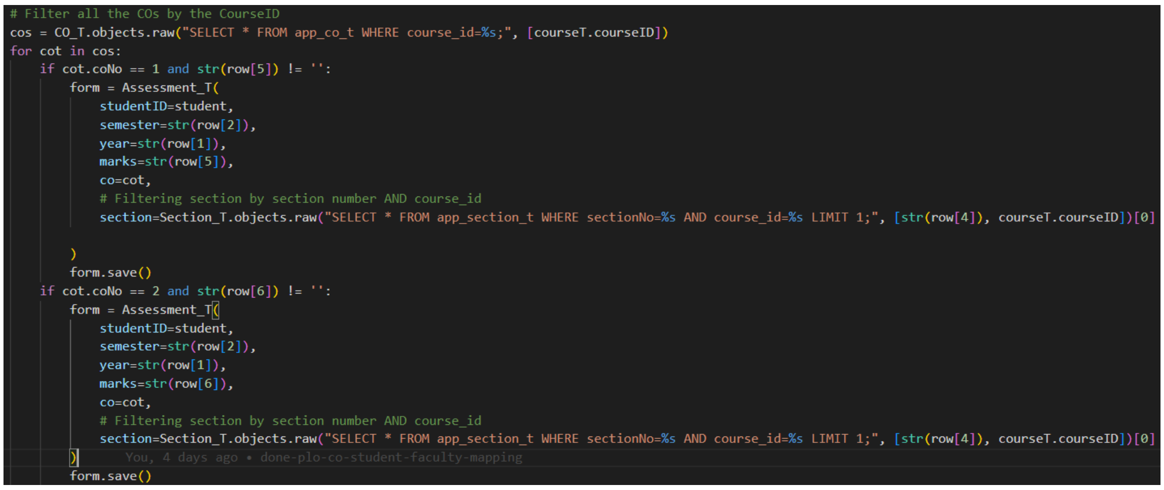
```python
def gradeInputForm(request):
    if request.user.is_authenticated:
        if request.user.role == 'Faculty':
            success = 'success'
            form = GradeInputForm()
            if request.method == 'POST':
                try:
                    # Filter the student from user_t table by Student_ID
                    student_ID = User_T.objects.raw("SELECT * FROM app_user_t WHERE username=%s;", [request.POST['studentID']])[0]
                    # Filter the Course from the course_t by Course_ID
                    courseT = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [request.POST['course']])[0]
                    form = CourseGrade_T(
                        studentID = student_ID,
                        eduYear = request.POST['eduYear'],
                        eduSemester = request.POST['eduSemester'],
                        course = courseT,
                        section = request.POST['section'],
                        grade = request.POST['grade']
                    )
                    form.save()
                    messages.add_message(request, messages.SUCCESS, 'GRADE Submission Successful')
                except:
                    success = 'danger'
                    messages.add_message(
                        request, messages.SUCCESS, 'GRADE Submission Failed!')

            return render(request, 'faculty/gradeInputForm.html', {    'form':form,
                                                                    'courses': Course_T.objects.all(),
                                                                    'success': success})
        else:
            return redirect('home')
```

Figure: Grade Submission form per Course and per student with Backend Code

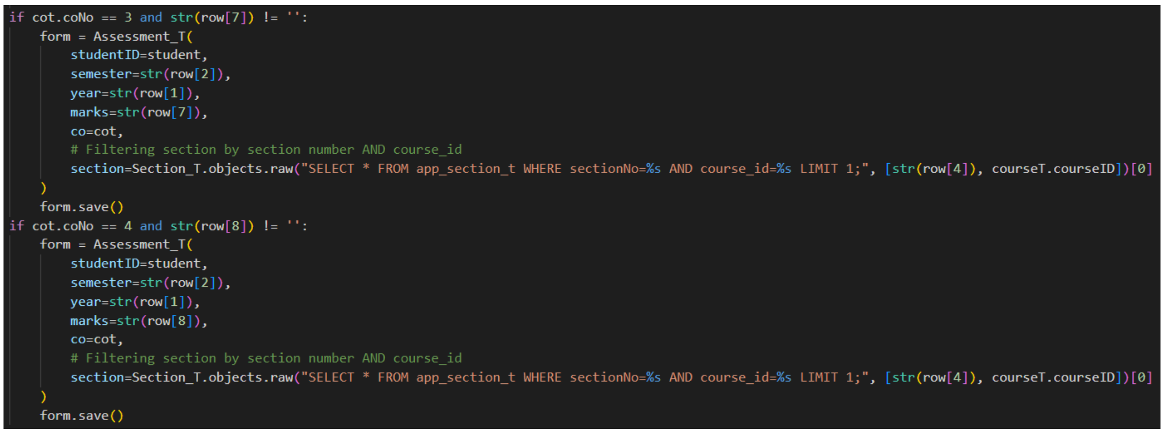

Students Performance Monitoring System
Import Grades and COs from CSV file
CSV
Choose File No file chosen
Import
Copyright © 2023 Group-27 CSE303 Section-03 Spring23. Team Data Armour



```
# Import Grades from CSV file
from io import TextIOWrapper
def gradeInputFromCSV(request):
    if request.user.is_authenticated:
        if request.user.role == 'Faculty':
            success = 'success'
            if request.method == 'POST':
                csv_file = request.FILES['csv_file']
                #data_frame = pd.read_csv(csv_file, index_col=False, iterator=True)
                data_frame = csv.reader(TextIOWrapper(csv_file, encoding='utf-8'))
                print(data_frame)
                try:
                    for row in data_frame:
                        print(row)
                        try:
                            # Get the student object filtering Student_ID importing from CSV data_frame
                            student = User_T.objects.raw("SELECT * FROM app_user_t WHERE username=%s;", [str(row[0])])[0]
                            # Getting Course Object filtering course_ID importing from CSV data_frame
                            courseT = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [str(row[3])])[0]
                            print('touches')
                            data = CourseGrade_T(studentID=student,
                                eduYear=str(row[1]),        You, 4 days ago • done-plo-co-student-faculty-mapping
                                eduSemester=str(row[2]),
                                course=courseT,
                                section=str(row[4]),
                                grade=str(row[9])
                            )
                            data.save()
```

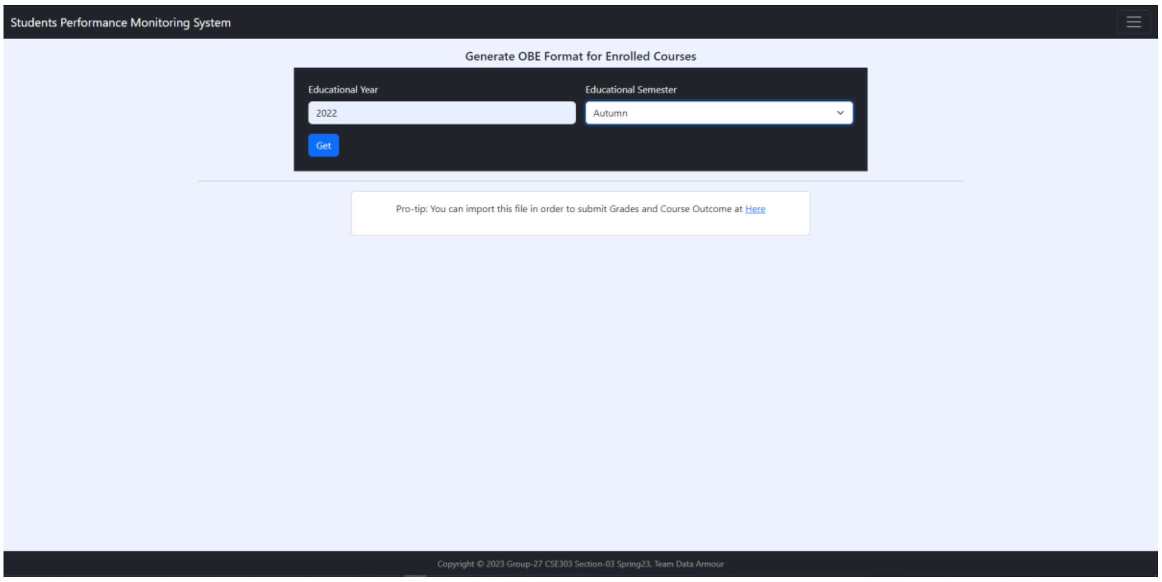


```
# Filter all the COs by the CourseID
cos = CO_T.objects.raw("SELECT * FROM app_co_t WHERE course_id=%s;", [courseT.courseID])
for cot in cos:
    if cot.coNo == 1 and str(row[5]) != '':
        form = Assessment_T(
            studentID=student,
            semester=str(row[2]),
            year=str(row[1]),
            marks=str(row[5]),
            co=cot,
            # Filtering section by section number AND course_id
            section=Section_T.objects.raw("SELECT * FROM app_section_t WHERE sectionNo=%s AND course_id=%s LIMIT 1;", [str(row[4]), courseT.courseID])[0]

        )
        form.save()
    if cot.coNo == 2 and str(row[6]) != '':
        form = Assessment_T(
            studentID=student,
            semester=str(row[2]),
            year=str(row[1]),
            marks=str(row[6]),
            co=cot,
            # Filtering section by section number AND course_id
            section=Section_T.objects.raw("SELECT * FROM app_section_t WHERE sectionNo=%s AND course_id=%s LIMIT 1;", [str(row[4]), courseT.courseID])[0]
        )    You, 4 days ago • done-plo-co-student-faculty-mapping
        form.save()
```



```
if cot.coNo == 3 and str(row[7]) != '':
    form = Assessment_T(
        studentID=student,
        semester=str(row[2]),
        year=str(row[1]),
        marks=str(row[7]),
        co=cot,
        # Filtering section by section number AND course_id
        section=Section_T.objects.raw("SELECT * FROM app_section_t WHERE sectionNo=%s AND course_id=%s LIMIT 1;", [str(row[4]), courseT.courseID])[0]
    )
    form.save()
if cot.coNo == 4 and str(row[8]) != '':
    form = Assessment_T(
        studentID=student,
        semester=str(row[2]),
        year=str(row[1]),
        marks=str(row[8]),
        co=cot,
        # Filtering section by section number AND course_id
        section=Section_T.objects.raw("SELECT * FROM app_section_t WHERE sectionNo=%s AND course_id=%s LIMIT 1;", [str(row[4]), courseT.courseID])[0]
    )
    form.save()
```

Figure: Grades and CO percentage insertion by importing formatted CSV file for Faculty User with Backend code



Students Performance Monitoring System

Generate OBE Format for Enrolled Courses

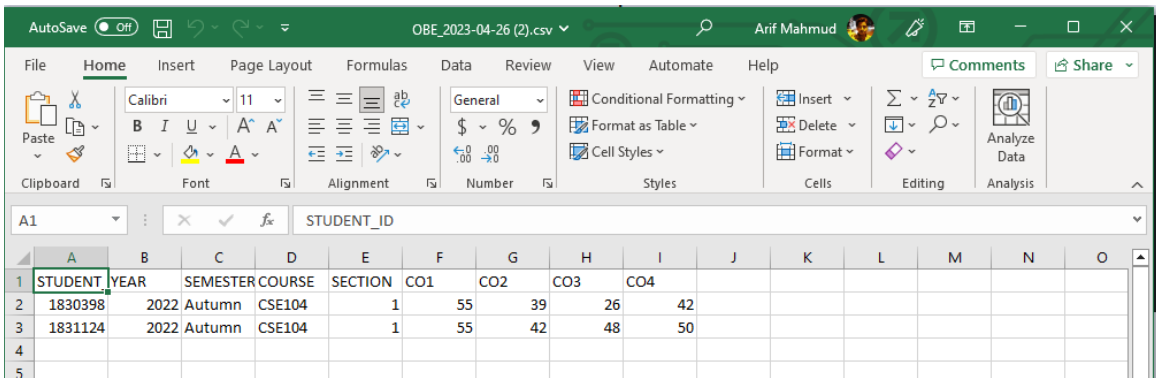
Educational Year: 2022

Educational Semester: Autumn

Get

Pro-tip: You can import this file in order to submit Grades and Course Outcome at Here

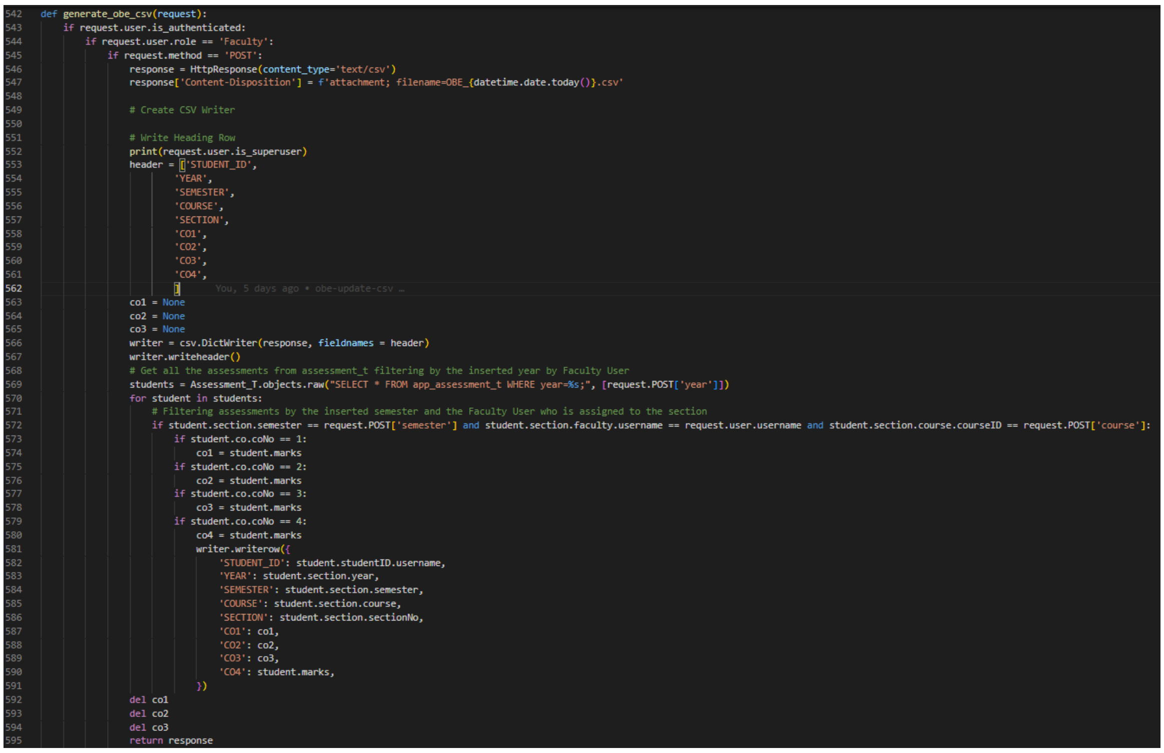
Copyright © 2023 Group-27 CSE303 Section-03 Spring23, Team Data Armour



OBE_2023-04-26 (2).csv

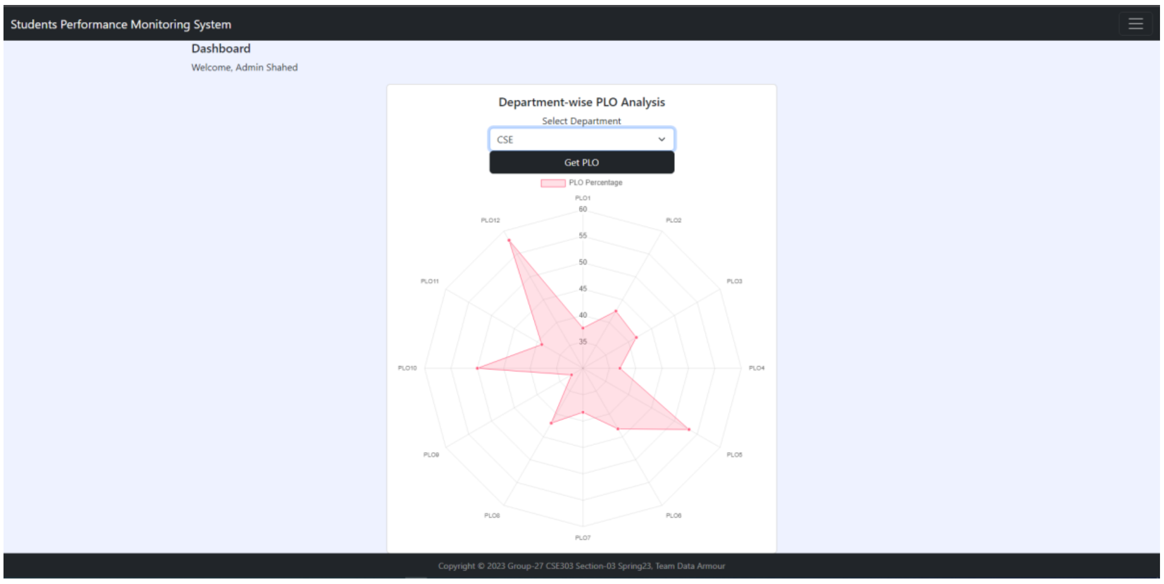
A1: STUDENT_ID

STUDENT	YEAR	SEMESTER	COURSE	SECTION	CO1	CO2	CO3	CO4
1830398	2022	Autumn	CSE104	1	55	39	26	42
1831124	2022	Autumn	CSE104	1	55	42	48	50



```python
def generate_obe_csv(request):
    if request.user.is_authenticated:
        if request.user.role == 'Faculty':
            if request.method == 'POST':
                response = HttpResponse(content_type='text/csv')
                response['Content-Disposition'] = f'attachment; filename=OBE_{datetime.date.today()}.csv'

                # Create CSV Writer

                # Write Heading Row
                print(request.user.is_superuser)
                header = ['STUDENT_ID',
                        'YEAR',
                        'SEMESTER',
                        'COURSE',
                        'SECTION',
                        'CO1',
                        'CO2',
                        'CO3',
                        'CO4',
                        ]       You, 5 days ago • obe-update-csv …
                co1 = None
                co2 = None
                co3 = None
                writer = csv.DictWriter(response, fieldnames = header)
                writer.writeheader()
                # Get all the assessments from assessment_t filtering by the inserted year by Faculty User
                students = Assessment_T.objects.raw("SELECT * FROM app_assessment_t WHERE year=%s;", [request.POST['year']])
                for student in students:
                    # Filtering assessments by the inserted semester and the Faculty User who is assigned to the section
                    if student.section.semester == request.POST['semester'] and student.section.faculty.username == request.user.username and student.section.course.courseID == request.POST['course']:
                        if student.co.coNo == 1:
                            co1 = student.marks
                        if student.co.coNo == 2:
                            co2 = student.marks
                        if student.co.coNo == 3:
                            co3 = student.marks
                        if student.co.coNo == 4:
                            co4 = student.marks
                            writer.writerow({
                                'STUDENT_ID': student.studentID.username,
                                'YEAR': student.section.year,
                                'SEMESTER': student.section.semester,
                                'COURSE': student.section.course,
                                'SECTION': student.section.sectionNo,
                                'CO1': co1,
                                'CO2': co2,
                                'CO3': co3,
                                'CO4': student.marks,
                            })
                del co1
                del co2
                del co3
                return response
```

Figure: Form of Generate OBE CSV File for Enrolled Courses filtering Semester and Year of Logged in current Faculty User with BE code



Students Performance Monitoring System

Dashboard

Welcome, Admin Shahed

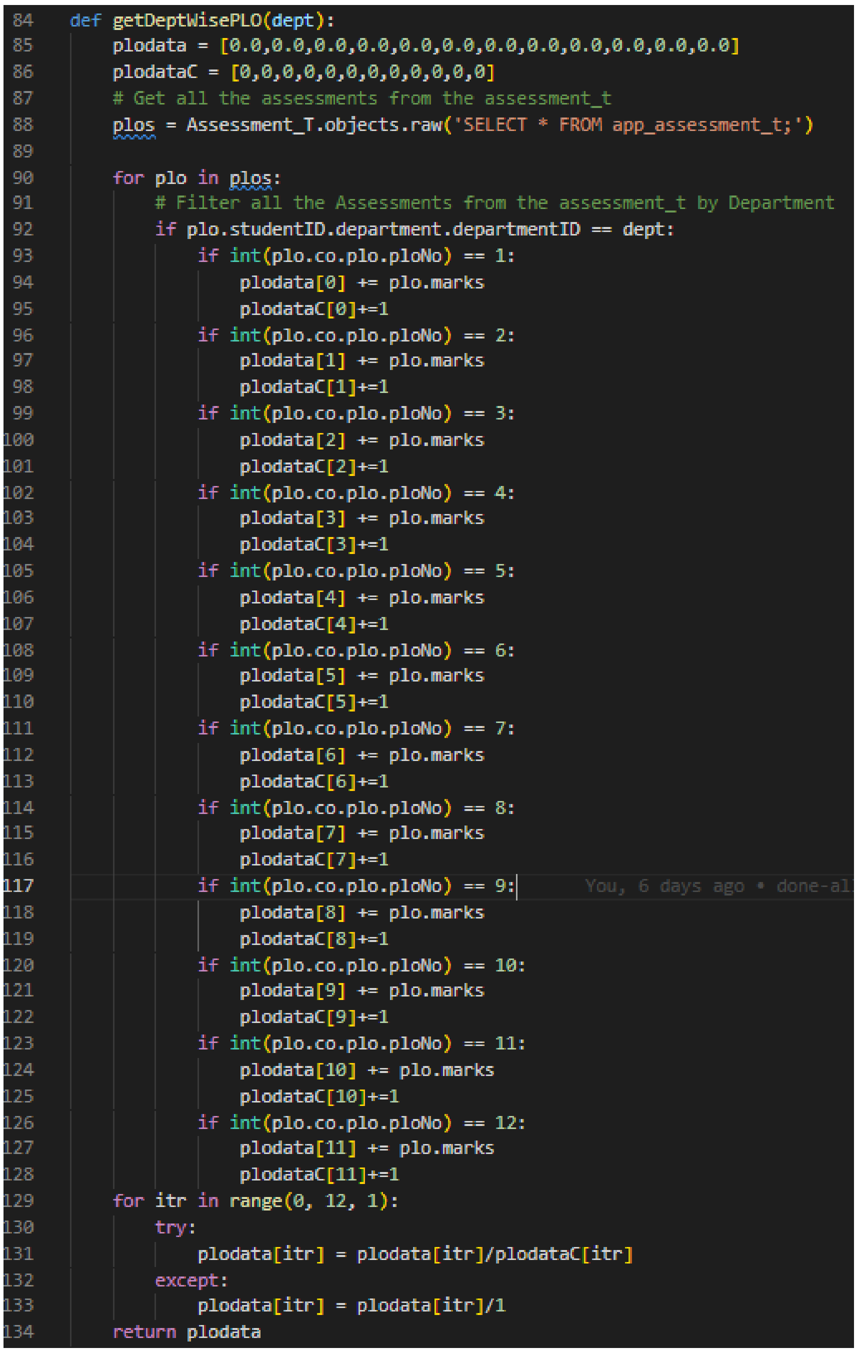
Department-wise PLO Analysis

Select Department

CSE

Get PLO

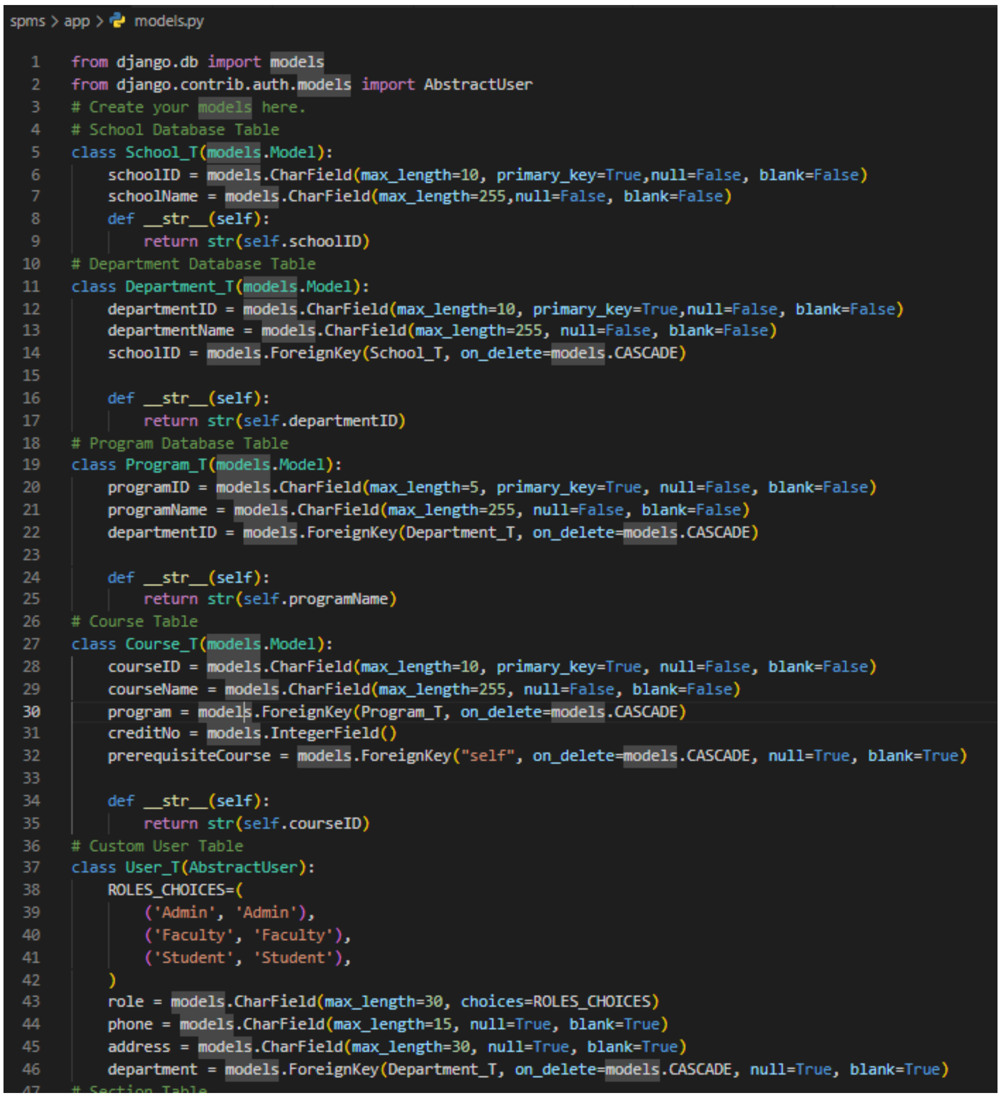
Copyright © 2023 Group-27 CSE303 Section-03 Spring23, Team Data Armour



```
def getDeptWisePLO(dept):
    plodata = [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
    plodataC = [0,0,0,0,0,0,0,0,0,0,0,0]
    # Get all the assessments from the assessment_t
    plos = Assessment_T.objects.raw('SELECT * FROM app_assessment_t;')

    for plo in plos:
        # Filter all the Assessments from the assessment_t by Department
        if plo.studentID.department.departmentID == dept:
            if int(plo.co.plo.ploNo) == 1:
                plodata[0] += plo.marks
                plodataC[0]+=1
            if int(plo.co.plo.ploNo) == 2:
                plodata[1] += plo.marks
                plodataC[1]+=1
            if int(plo.co.plo.ploNo) == 3:
                plodata[2] += plo.marks
                plodataC[2]+=1
            if int(plo.co.plo.ploNo) == 4:
                plodata[3] += plo.marks
                plodataC[3]+=1
            if int(plo.co.plo.ploNo) == 5:
                plodata[4] += plo.marks
                plodataC[4]+=1
            if int(plo.co.plo.ploNo) == 6:
                plodata[5] += plo.marks
                plodataC[5]+=1
            if int(plo.co.plo.ploNo) == 7:
                plodata[6] += plo.marks
                plodataC[6]+=1
            if int(plo.co.plo.ploNo) == 8:
                plodata[7] += plo.marks
                plodataC[7]+=1
            if int(plo.co.plo.ploNo) == 9:        You, 6 days ago • done-al
                plodata[8] += plo.marks
                plodataC[8]+=1
            if int(plo.co.plo.ploNo) == 10:
                plodata[9] += plo.marks
                plodataC[9]+=1
            if int(plo.co.plo.ploNo) == 11:
                plodata[10] += plo.marks
                plodataC[10]+=1
            if int(plo.co.plo.ploNo) == 12:
                plodata[11] += plo.marks
                plodataC[11]+=1
    for itr in range(0, 12, 1):
        try:
            plodata[itr] = plodata[itr]/plodataC[itr]
        except:
            plodata[itr] = plodata[itr]/1
    return plodata
```

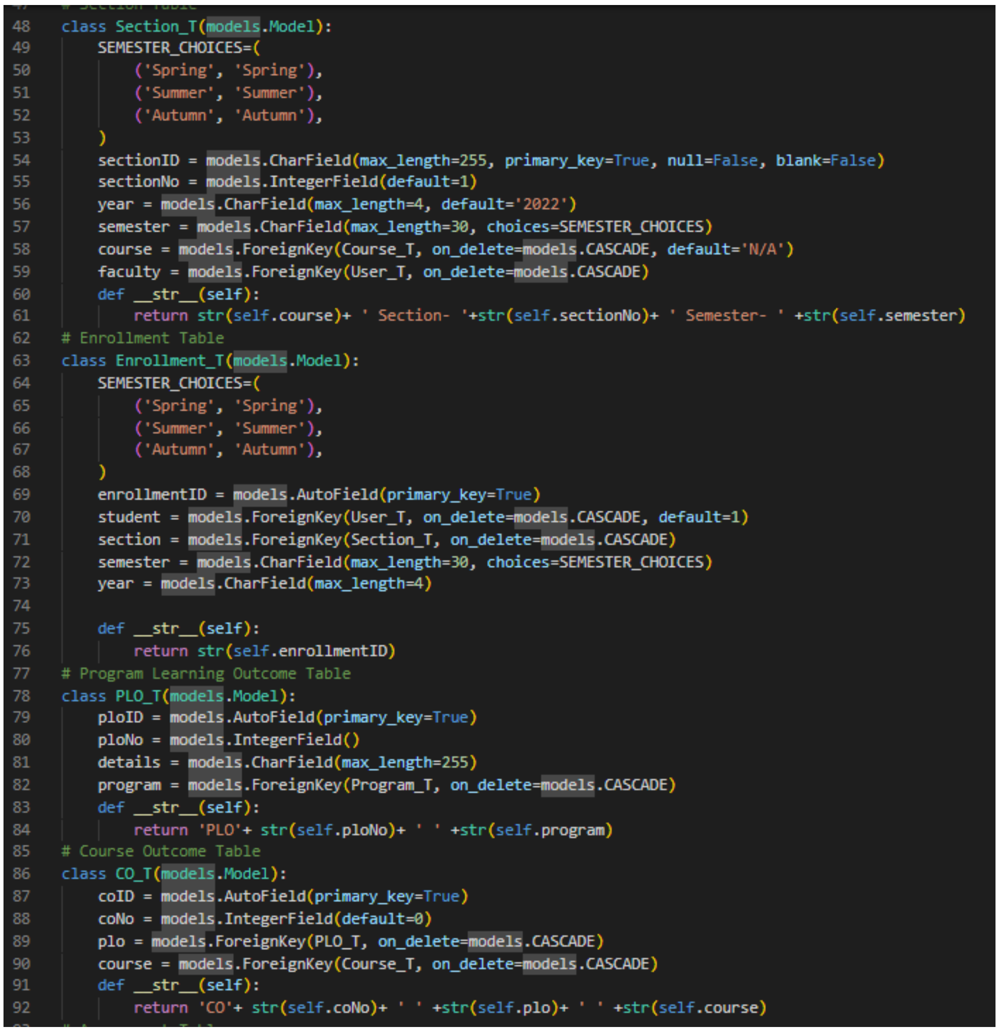
Figure: Admin Dashboard, can check All the Department-wise PLO and CO Graph Analysis with Backend Code

Database Model:


spms > app > models.py

```python
from django.db import models
from django.contrib.auth.models import AbstractUser
# Create your models here.
# School Database Table
class School_T(models.Model):
    schoolID = models.CharField(max_length=10, primary_key=True,null=False, blank=False)
    schoolName = models.CharField(max_length=255,null=False, blank=False)
    def __str__(self):
        return str(self.schoolID)
# Department Database Table
class Department_T(models.Model):
    departmentID = models.CharField(max_length=10, primary_key=True,null=False, blank=False)
    departmentName = models.CharField(max_length=255, null=False, blank=False)
    schoolID = models.ForeignKey(School_T, on_delete=models.CASCADE)

    def __str__(self):
        return str(self.departmentID)
# Program Database Table
class Program_T(models.Model):
    programID = models.CharField(max_length=5, primary_key=True, null=False, blank=False)
    programName = models.CharField(max_length=255, null=False, blank=False)
    departmentID = models.ForeignKey(Department_T, on_delete=models.CASCADE)

    def __str__(self):
        return str(self.programName)
# Course Table
class Course_T(models.Model):
    courseID = models.CharField(max_length=10, primary_key=True, null=False, blank=False)
    courseName = models.CharField(max_length=255, null=False, blank=False)
    program = models.ForeignKey(Program_T, on_delete=models.CASCADE)
    creditNo = models.IntegerField()
    prerequisiteCourse = models.ForeignKey("self", on_delete=models.CASCADE, null=True, blank=True)

    def __str__(self):
        return str(self.courseID)
# Custom User Table
class User_T(AbstractUser):
    ROLES_CHOICES=(
        ('Admin', 'Admin'),
        ('Faculty', 'Faculty'),
        ('Student', 'Student'),
    )
    role = models.CharField(max_length=30, choices=ROLES_CHOICES)
    phone = models.CharField(max_length=15, null=True, blank=True)
    address = models.CharField(max_length=30, null=True, blank=True)
    department = models.ForeignKey(Department_T, on_delete=models.CASCADE, null=True, blank=True)
# Section Table
```

Figure: Database Model Code Snippets



```
class Section_T(models.Model):
    SEMESTER_CHOICES=(
        ('Spring', 'Spring'),
        ('Summer', 'Summer'),
        ('Autumn', 'Autumn'),
    )
    sectionID = models.CharField(max_length=255, primary_key=True, null=False, blank=False)
    sectionNo = models.IntegerField(default=1)
    year = models.CharField(max_length=4, default='2022')
    semester = models.CharField(max_length=30, choices=SEMESTER_CHOICES)
    course = models.ForeignKey(Course_T, on_delete=models.CASCADE, default='N/A')
    faculty = models.ForeignKey(User_T, on_delete=models.CASCADE)
    def __str__(self):
        return str(self.course)+ ' Section- '+str(self.sectionNo)+ ' Semester- ' +str(self.semester)
# Enrollment Table
class Enrollment_T(models.Model):
    SEMESTER_CHOICES=(
        ('Spring', 'Spring'),
        ('Summer', 'Summer'),
        ('Autumn', 'Autumn'),
    )
    enrollmentID = models.AutoField(primary_key=True)
    student = models.ForeignKey(User_T, on_delete=models.CASCADE, default=1)
    section = models.ForeignKey(Section_T, on_delete=models.CASCADE)
    semester = models.CharField(max_length=30, choices=SEMESTER_CHOICES)
    year = models.CharField(max_length=4)

    def __str__(self):
        return str(self.enrollmentID)
# Program Learning Outcome Table
class PLO_T(models.Model):
    ploID = models.AutoField(primary_key=True)
    ploNo = models.IntegerField()
    details = models.CharField(max_length=255)
    program = models.ForeignKey(Program_T, on_delete=models.CASCADE)
    def __str__(self):
        return 'PLO'+ str(self.ploNo)+ ' ' +str(self.program)
# Course Outcome Table
class CO_T(models.Model):
    coID = models.AutoField(primary_key=True)
    coNo = models.IntegerField(default=0)
    plo = models.ForeignKey(PLO_T, on_delete=models.CASCADE)
    course = models.ForeignKey(Course_T, on_delete=models.CASCADE)
    def __str__(self):
        return 'CO'+ str(self.coNo)+ ' ' +str(self.plo)+ ' ' +str(self.course)
```

Figure: Database Model Code Snippets

CH-5 CONCLUSION

PROBLEM & SOLUTION

1. **Our ability to utilize this program to its full potential has been hampered by the limited period of the semester. We intend to make enhancements with greater analysis when given more time, but we believe we have produced the best program we could give the time and resources available.**
2. **We might think that we could have produced far more trustworthy and accurate outcomes, representations, and predictions if given more tools and information to work with.**

ADDITIONAL FEATURE & FUTURE DEVELOPMENT

Future Development scope:

1. **The number of users will be increased to include advisers, who will receive pertinent data on the students they are advising for better and more advantageous interactions between students and advisors.**
2. **Project goals include adding a component that predicts a candidate's grade based on prior grades and performance.**
3. **Whenever Faculties will Update a Student's COs and Grade that student will get Email notifications of updated PLO Analysis**
4. **All the Stakeholder's will have limited access to the System, for example: UGC will have limited access to the system and will have Overview of the Academic's necessary Incites.**

CONCLUSION & RECOMMENDATIONS

We think the idea we had for our SPM software has been created, built, and implemented in the greatest way possible. With the appropriate application of this software, we intend to significantly raise the standard of education offered by institutions. This program can be used by students who want to become better and more capable scholars, by faculties to keep better track of their students and adjust their teaching strategies accordingly, and by institution members to more effectively manage their resources.