STUDENT PERFORMANCE MONITORING SYSTEM

FINAL REPORT
GROUP -02

Name	ID
Farha Jabin Oyshee	1722033
Syeda Nazia Haque	1830928
Shahriar Mahmud Anik	1830872
Amit Mahmud	1820013
Sharmin Islam Shroddha	1822217
Mim Islam	1820804

Table of Contents

LIST OF FIGURES:

CHAPTER 1 – INTRODUCTION:

In a university a student gets admitted under a specific Degree program. Each program belongs to a department and the related departments are kept under Schools. Students usually take courses as per the curriculum of their respective programs - notably consisting of the following:

- Business & Entrepreneurship
- Engineering, Technology & Sciences
- Environment and Life Sciences
- Liberal Arts & Social Sciences
- Pharmacy and Public Health.

The universities in Bangladesh work closely with relevant government education institutions and organizations such as the University Grants Commission (UGC), Ministry of Education, and other necessary institutes for each of the schools, regularly updating its curriculums and putting in a system to monitor student performance based on a quantified approach between course curriculum and standards set by UGC and the Bangladesh government and constantly tracking student performance for every semester – mainly, using Outcome-Based Education (OBE) for monitoring performance and setting university curriculum.

This report highlights the study of the current student performance monitoring system that all universities of Bangladesh can use, fulfill the required analysis of its processes, and propose a new and better improved version of the system that reduces error, makes analysis of data and report generation easier by all vested quarters and produce/show valuable information needed for universities and its collaborators in making necessary improvements in academia to produce better scholars. The first part focuses on the details of the organizations in question and the project that we have undertaken for it. The second part focuses on the existing system and its shortcomings and an introduction of the proposed system that we plan to replace the existing system with. The third and fourth will be heavily technical and focus on how we plan to bring the proposed system into being.

According to our research on the existing system of student performance monitoring we have come across areas where some essential changes are required to make the system more efficient so there's an ease of communication between the stakeholders. Moreover, the changes will take away chances for error, data duplication, and most importantly stakeholders can have access to a large dataset and view meaningful information from our system instead of manually going through documents. In other words, our system, department/course/student wise analysis will be present, so users can view them directly from the system.

As we proceed with the report, we will dig deeper into how the current student performance monitoring system operates, the business processes involved, where there are concerns and issues related to data management, and how we can make a better system to address these issues for fixing and improvement.

A. BACKGROUND OF THE ORGANIZATION - ALL UNIVERSITIES:

Universities in Bangladesh are mainly categorized into three different types: public (government owned and subsidized), private (private sector-based universities), and international (operated and funded by international organizations such as the Organization of Islamic Cooperation). Currently there are 49 public universities, 107 private universities, and 3 international universities in Bangladesh.

All the universities, over-time, continued producing graduates with marketable skills only because of staying disciplined and up to date with the on-going curriculum and progress system. Bangladeshi universities are affiliated with the UGC, a commission created according to the Presidential Order (P.O. No. 10 of 1973) of the government of the People's Republic of Bangladesh.

Continuously growing its lab facilities and flourishing on its curriculum according to current market economic demands, all the Departments of Schools has constantly worked with IEB, UGC and the Ministry of Education to track their students' overall performance under specific periods by quantifying specific courses and its relating assessments into measurable trackers to gain valuable insights for improvement of students over the years as a student in a certain department. The students are assessed based on the processes and credentials set by IEB (for engineering department) and other government potentials to shape them to become coming graduates from universities in Bangladesh. These sets of standards come in the form of Program Educational Objectives (PEO) and Program Learning Outcomes (PLO) [1] for specific departments in an Accreditation Manual which are mapped to specific courses by relevant Course Instructors and Co-Ordinator. This allows the Departments and all other relevant stakeholders to have a calculating assessment of the current state-of-affairs and the performance of each student under each course for every semester. This will also allow users to track performance of faculties, courses, departments and schools and provides valuable insight for making necessary improvements for the future.

B. BACKGROUND OF THE PROJECT SPMS:

The goal of the project is to build a software system for the institutional bodies and education authorities such as IEB, UGC, as well as other stakeholders and retrieve the statistical information about programs, departments, and student's overall performance information. So, our system will give organized data to measure their (instructors, students, departments and so on) productivity regarding the outcome consequences of the course activities.

C. OBJECTIVE OF THE PROJECT SPMS:

Our project is completely concerned about making a system which surely monitors the impact of policies against overall administrative goals and summarizes the academics of all universities through the database. For evaluation purposes the system would be able to store individual assessment marks as well as the marks of those assessments with respect to their Course Outcomes (CO) and Program Learning Outcomes (PLO) accordingly in the database of the system to observe the outcome and performance of the students, instructors, and departments. The students can look over the overall performance to their satisfaction of a particular course objectives. And the higher authorities can manage the degree in comparison to different course outcomes based on students' performance and their achievements and on what topics to work on. This automated software along with easy-to-use interfaces will reduce the manual processes drastically and give ideas about the students, instructors, and different participants of the university while it comes in terms of improving the quality of the education system.

D. SCOPE OF THE PROJECT:

Scope of the project is a necessity to ensure the accomplishment of a project. As we are modifying an existing system, we have to ensure that the proposed system will be more effective than the existing one. Because it is very inefficient to maintain detailed records of students' performance and documents, therefore there is a need for an upgraded and automated student performance monitoring system (SPMS). Here we basically focused on all universities which will confirm the OBE regulations for which we are doing this project as well as the project has the potential of being useful to IEB/UGC.

So, the purpose of this project is to develop the current system with our proposed solution where we are looking forward to:

- Create a system which takes input with an easy to go interface.
- Make the system accessible by authorized holders.
- Mapping COs and PLOs
- Data assembling and sorting
- Instantly insert, update, and delete necessary data from the database
- Program analysis
- Generating reports

CHAPTER 2 – REQUIREMENT ANALYSIS:

The Requirement Analysis requires using industry tools, methods, and standards, to research and visualize the current system and the processes that go into the business operation of a certain organization. “Requirements Analysis is the process of determining what the database is to be used for. It involves interviews with user groups and other stakeholders to identify what functionality they require from the database, what kinds of data they wish to process and the most frequently performed operations.” This analysis will give us an idea about each stakeholder and how they interact with each other. We use simple notations and symbols to give anyone the idea of how a business process works and dissect it accordingly. First, we will come across issues that usually deal with the manual work regarding student performance monitoring in the existing system followed by third party individuals causing errors in the system.

A. RICH PICTURE – EXISTING BUSINESS SYSTEM:

Rich pictures are modelled using diagrams and illustrated graphically which has the main elements and relationship they have between them according to the business system in place. A complete rich picture could be of significance to other stakeholders of the problems in an existing system and permit them to apprehend many different facets of the situation. Both the structure and the processes of a given situation are focused on Rich Picture. The Rich Picture Analysis also considers the following:

- Structures
- Processes
- Climate
- People
- Issues expressed by people.
- Conflict

The following rich picture was created keeping that in mind.

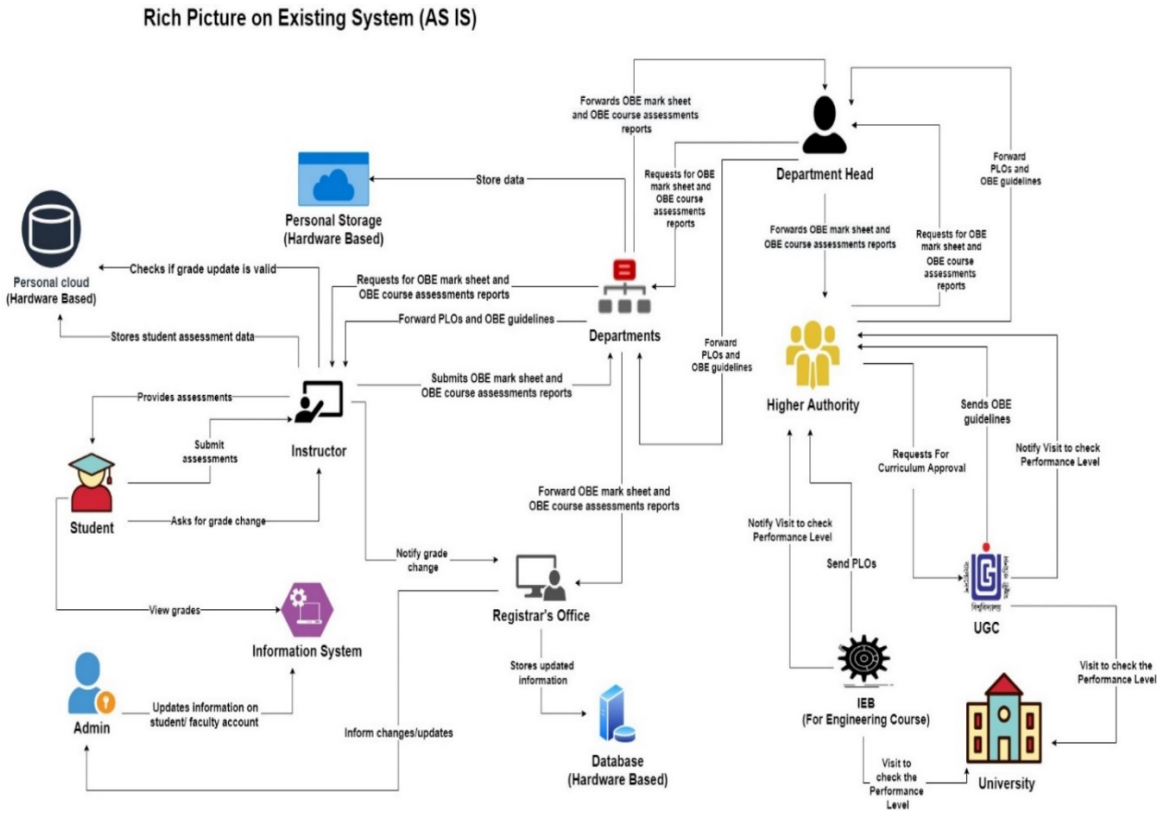


Figure 1: Rich Picture of Existing System

The Rich Picture Analysis shows us that we have the following types of stakeholders:

1. IEB/UGC
2. Higher Authority: VC/Board of Trustees
3. Department (working under Head of Department)
4. Instructor
5. Register's Office
6. Admin (working under Register's Office)
7. Students

We can also identify four separate storage systems or facilities, namely:

1. The Department Storage
2. The Registrar's Office Storage
3. Information Storage
4. Instructor's Personal Storage

From this Rich Picture we have drawn out 4 processes that are key to monitoring student performance and improving curriculum. The processes are as follows:

1. OBE outline and course outline Submission.
2. Course mapping/planning the course for a particular semester.
3. Conducting the course and performing regular assessment.
4. Grade change and prepare a performance report.

B. SIX ELEMENT ANALYSIS-EXISTING BUSINESS SYSTEM:

Process	System Roles					
	Human	**Non-Comp Hardware**	**Computing Hardware**	**Software**	**Database**	**Network and Communication**
Mapping of COs from PLOs	**UGC:** 1.Send OBE guidelines to the Higher Authority. **IEB:** 1.Send PLOs to the Higher Authority. **Higher Authority:** 1. Receives OBE guidelines and PLOs from UGC and IEB respectively. 2. Forward OBE guidelines and PLOs to the Department Head. **Department Head:** 1. Receives OBE guidelines and PLOs from the Higher	**Paper:** 1.Taking down necessary notes. 2.Used for printing reports. **Pen:** 1.Used for writing on paper. 2. Signing on the paper. **Stapler:** 1. Staples multiple pages.	**Computer:** 1.Uploads/ Downloads from the Website. 2. Used to send mails about necessary details. 3.Updating the information. 4.Storing the information. **Printer:** 1. Prints relevant documents. **Scanner:** 1. Scans hard copy reports, documents, list and others for future reference.	**Microsoft Office:** 1.Make notice, reports etc. using Microsoft word. 2. Use tables to map and manage using Microsoft Excel. 3. Make presentations to represent a summary of collected information using Microsoft PowerPoint 4.Using Microsoft Outlook for emailing purposes. **PDF reader:** 1. Reading the necessary documents. **Web Browser:** 1. Access information from other sources for	**Personal Storage:** 1.Store/access in cloud or physical storages. **Personal cloud:** 1.Store/access in cloud or physical storages. **Database:** 1.Store/access in cloud or physical storages. **File Cabinet:** 1.Maintaining the necessary documents in hard copy.	**Internet:** 1. Access the web to download and upload. 2. Sharing the information. 3.Accessingthe website and portals. **Intercom:** 1.Connect to people within the office for any requirement. **Networking devices (Router, Modem):** 1.Make necessary arrangements to access the Internet.

	Authority. 2.Forwards PLOs and OBE guidelines to the Department. 3.Directs Department to instruct Instructors to design course outline and course assessment reports accordingly. **Department:** 1.Receives OBE guidelines and PLOs from the Department Head. 2.Forward the OBE guidelines and PLOs to the Instructors. **Instructor:** 1. Receives OBE guidelines and PLOs from the Department. 2.List course contents. 3. List CO's 4.Map course content to course outcomes.			reference. 2. Access the web or any custom hosted server and upload, view and update related data. 3.Uploads/Downloads from the website. 4. Used to send mails to those who are concerned about necessary details and relevant documents.		

	5. Map COs from PLO's 6. Map COs to assessments. 7. Design OBE marksheet and OBE course assessment report accordingly.					
Conduct Course assessments according to Cos	**Instructor:** 1. After Mapping COs from PLOs manually Instructor Sets assessments according to mapped COs. 2.Conduct exams, assignments, and project work throughout the semester accordingly. **Student:** 1.Receives assigned assessments from the instructor. 2. After completing Submits the assessments.	**Paper:** 1. Taking down necessary notes. 2.Used for printing reports. **Pen:** 1. Used for writing on paper. 2.Signing on the paper. **Stapler:** 1. Staples multiple pages.	**Computer:** 1. Uploads/ Downloads from the website. 2. Used to send mails about necessary details. 3.Updating the information. 4.Storing the information. **Printer:** 1. Prints relevant documents. **Scanner:** 1. Scans hard copy reports, documents, list and others for future reference.	**Microsoft Office:** 1. Make notice, reports, etc. using Microsoft Word. 2. Use tables to map and manage using Microsoft Excel. 3. Make presentations to represent a summary of collected information using Microsoft PowerPoint 4. Using Microsoft Outlook for emailing purposes. **PDF reader:** 1. Reading the necessary documents. **Web Browser:** 1. Access	**Personal storage:** 1. Store/ access in cloud or physical storages. **Personal cloud:** 1. Store/ access in cloud or physical storages. **Database:** 1. Store/ access in cloud or physical storages. **File Cabinet:** 1.Maintaining the necessary documents in hard copy.	**Internet:** 1. Access the web to download and upload. 2. Sharing the information. 3.Accessing the website and portals. **Intercom:** 1. Connect to people within the office for any requirement. **Networking devices (Router, Modem):** 1. Make necessary arrangements to access the Internet.

				information from other sources for reference. 2. Access the web or any custom hosted server and upload, view and update related data. 3.Uploads/Downloads from the website. 4. Used to send mails to those who are concerned about necessary details and relevant documents.		
Create OBE Marksheet and OBE Course assessment Report	**Instructor:** 1. Evaluate assessment marks of every individual student according to mapped COs throughout the semester on softcopies and hardcopies. 2. Record all the assessment marks of every student for a course in a OBE marksheet by each	**Paper:** 1. Taking down necessary notes. 2.Used for printing reports. **Pen:** 1. Used for writing on paper. 2. Signing on the paper. **Stapler:** 1. Staples multiple pages. **Calculator:** 1. Used for calculations.	**Computer:** 1. Uploads/ Downloads from the website. 2. Used to send mails about necessary details. 3.Updating the information. 4.Storing the information. **Printer:** 1. Prints relevant documents. **Scanner:** 1. Scans hard copy reports,	**Microsoft Office:** 1. Make notice, reports, etc. using Microsoft Word. 2. Use tables to map and manage using Microsoft Excel. 3. Make presentations to represent a summary of collected information using Microsoft PowerPoint 4. Using Microsoft Outlook for	**Personal storage:** 1. Store/ access in cloud or physical storages. **Personal cloud:** 1. Store/ access in cloud or physical storages. **Database:** 1. Store/ access in cloud or physical storages.	**Internet:** 1. Access the web to download and upload. 2. Sharing the information. 3.Accessing the website and portals. **Intercom:** 1. Connect to people within the office for any requirement. **Networking devices (Router, Modem):** 1. Make necessary arrangements to access the Internet.

	assessment question under each CO. 3. Calculate the total marks received for each COs on the OBE marksheet. 4. Calculate percentage received for each COs from the OBE marksheet for each student. 5. Calculate percentage of total students that have achieved a specific CO and add as a tabular format to the OBE course assessment report. 6. Send the final version of the OBE marksheet and OBE course assessment report to the department. 7. Submit a final grade for each individual student in the information system. 8. Store the OBE marksheet		documents, list, and others for future reference.	emailing purposes. **PDF reader:** 1. Reading the necessary documents. **Web Browser:** 1. Access information from other sources for reference. 2. Access the web or any custom hosted server and upload, view and update related data. 3.Uploads/Downloads from the website. 4. Used to send mails to those who are concerned about necessary details and relevant documents.	**File Cabinet:** 1.Maintaining the necessary documents in hard copy.	

	and OBE course assessment report in his/her personal storage. **Department:** 1. Receive the final version of the OBE marksheet and OBE course assessment report from the instructor. 2. Forward the OBE marksheet, OBE course assessment report to the Higher Authority. 3. Forward the OBE marksheet and OBE course assessment reports to the Registrar's office. 4. Store the OBE marksheet and OBE course assessment reports in the department storage. **Higher Authority:** 1. Receive					

	the final version of the OBE marksheet and OBE course assessment report from the Department. 2. View and analyze the performance of the students physically. **Registrar's Office:** 1. Receive the final version of the OBE marksheet and OBE course assessment report from the Department. 2. Store the final version of the OBE marksheet and OBE course assessment report in the Registrar's Office storage.					

View and Request grade change	**Student:** 1. Logs in the information system to view grades. 2. Request for grade change to Instructor if needed. **Instructor:** 1. Receives grade change request from the student. 2. Check all the assessment records from his/her personal storage. 3. If change in any marks in the assessment and the grade affect the CO/ PLO then he/she needs to update marks in his/her personal storage and inform the Registrar's office. **Registrar's Office:** 1.Receives request for grade change from the instructor. 2. Notifies	**Paper:** 1. Taking down necessary notes. 2.Used for printing reports. **Pen:** 1. Used for writing on paper. 2. Signing on the paper. **Stapler:** 1. Staples multiple pages.	**Computer:** 1. Uploads/ Downloads from the website. 2. Used to send mails about necessary details. 3.Updating the information. 4.Storing the information. **Printer:** 1. Prints relevant documents. **Scanner:** 1. Scans hard copy reports, documents, list and others for future reference.	**Microsoft Office:** 1. Make notice, reports, etc. using Microsoft Word. 2. Use tables to map and manage using Microsoft Excel. 3. Make presentations to represent a summary of collected information using Microsoft PowerPoint 4. Using Microsoft Outlook for emailing purposes. **PDF reader:** 1. Reading the necessary documents. **Web Browser:** 1. Access information from other sources for reference. 2. Access the web or any custom hosted server and upload, view and update related data. 3.Uploads/Downloads from the website. 4. Used to send mails to those who are concerned	**Personal storage:** 1. Store/ access in cloud or physical storages. **Personal cloud:** 1. Store/ access in cloud or physical storages. **Database:** 1. Store/ access in cloud or physical storages. **File Cabinet:** 1. Maintaining the necessary documents in hard copy.	**Internet:** 1. Access the web to download and upload. 2. Sharing the information. 3.Accessing the website and portals. **Intercom:** 1. Connect to people within the office for any requirement. **Networking devices (Router, Modem):** 1. Make necessary arrangements to access the Internet.

	Admin to change the grade of a specific student. **Admin:** 1. Receives a grade change request of a specific student from the registrar's office. 2. Updates a new grade in the student's profile on the information system.			about necessary details and relevant documents.		
University inspection and Analyze student performance	**IEB/ UGC:** 1. Inform the University's Higher Authority of a deadline for physical inspection and check OBE Marksheets, OBE Course Assessment Reports and other documents which are needed for quality inspection to make necessary improvements to degree programs. 2. Acknowledge	**Paper:** 1. Taking down necessary notes. 2.Used for printing reports. **Pen:** 1. Used for writing on paper. 2. Signing on the paper. **Stapler:** 1. Staples multiple pages.	**Computer:** 1. Uploads/ Downloads from the website. 2. Used to send mails about necessary details. 3.Updating the information. 4.Storing the information. **Printer:** 1. Prints relevant documents. **Scanner:** 1. Scans hard copy reports, documents, list and others for	**Microsoft Office:** 1. Make notice, reports, etc. using Microsoft Word. 2. Use tables to map and manage using Microsoft Excel. 3. Make presentations to represent a summary of collected information using Microsoft PowerPoint 4. Using Microsoft Outlook for emailing purposes.	**Personal storage:** 1. Store/ access in cloud or physical storages. **Personal cloud:** 1. Store/ access in cloud or physical storages. **Database:** 1. Store/ access in cloud or physical storages. **File Cabinet:**	**Internet:** 1. Access the web to download and upload. 2. Sharing the information. 3.Accessing the website and portals. **Intercom:** 1. Connect to people within the office for any requirement. **Networking devices (Router, Modem):** 1. Make necessary arrangements to access the Internet.

	the university's Higher Authority if govt. officials will visit the campus. 3. Visit the university and relevant Departments. 4. Receive the necessary documents gathered by the department physically. 5. Evaluate and decide the if need to change/ improve the department's Educational resources based on students' performanc e Trends physically. **Higher Authority:** 1. Receive information about the deadline for physical inspection and check OBE Marksheets, OBE Course Assessment Reports, and		future reference.	**PDF reader:** 1. Reading the necessary documents. **Web Browser:** 1. Access information from other sources for reference. 2. Access the web or any custom hosted server and upload, view and update related data. 3.Uploads/Do wnloads from the website. 4. Used to send mails to those who are concerned about necessary details and relevant documents.	1.Maintaining the necessary documents in hard copy.	

	other documents. 2. Inform the Department Head of a deadline for physical inspection and check OBE Marksheets, OBE Course Assessment Reports, and other documents. 3. Acknowledge the Department Head about if Govt. officials will visit the campus. **Department Head:** 1. Receive information from Higher authority about the deadline for physical inspection and check OBE Marksheets, OBE Course Assessment Reports, and other documents. 2. Receive information from the Higher Authority if Govt.					

	officials will visit the campus. 3. Request Department to gather records of OBE Marksheets, OBE course Assessment Reports and put them in a room to analyze students' performance trends. 4. Direct Department to gather necessary documents, OBE Marksheets, OBE course Assessment report for a given time-period specified by Govt. Officials and to put them in a room for inspection. **Departmen t:** 1. Gather necessary OBE Marksheets, OBE course Assessment Reports & other documents.					

	2. Store all the documents in a room for the Govt. Officials to inspect.					

C. PROCESS MODEL - EXISTING BUSINESS SYSTEM:

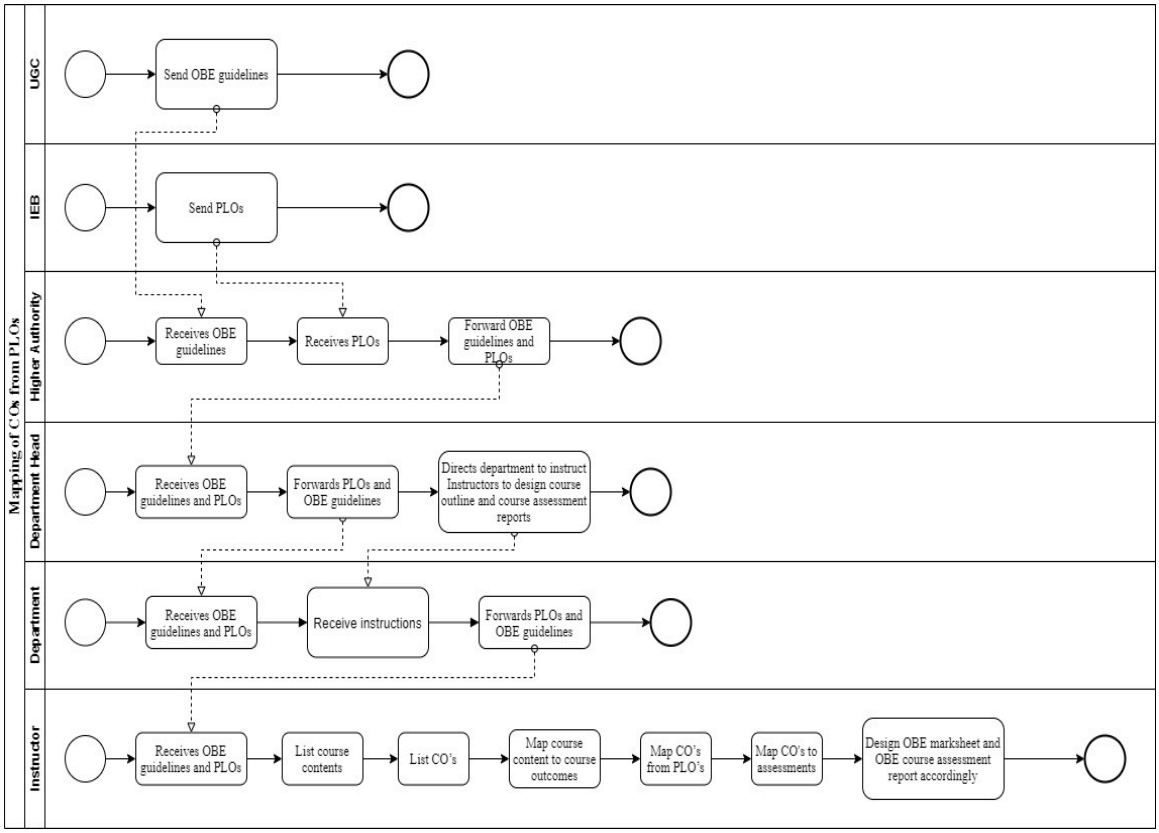


Figure 2: Mapping of COs from PLOs

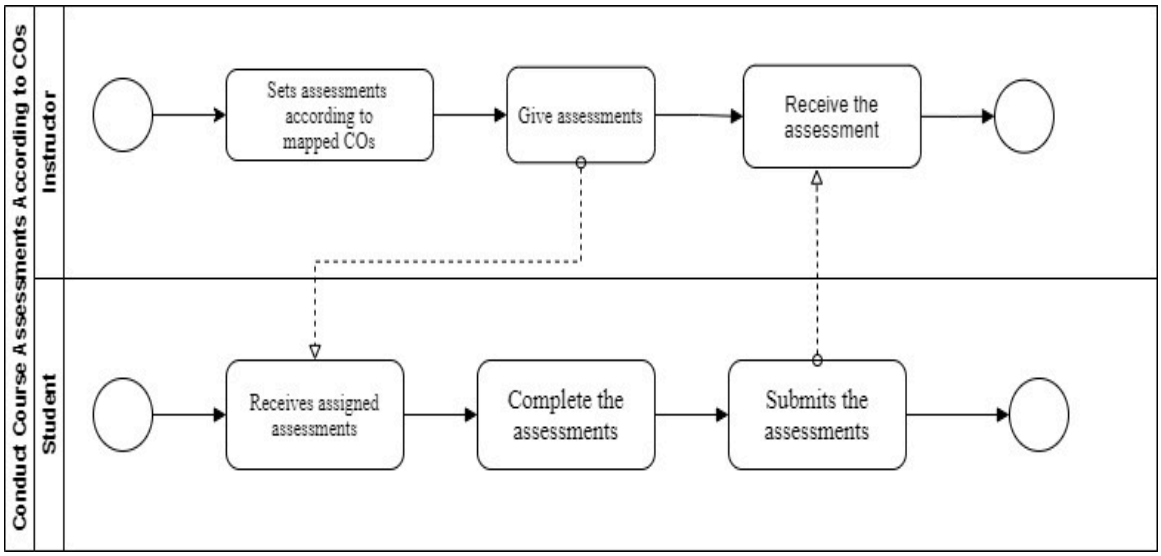


Figure 3: Conduct Course Assessments According to COs

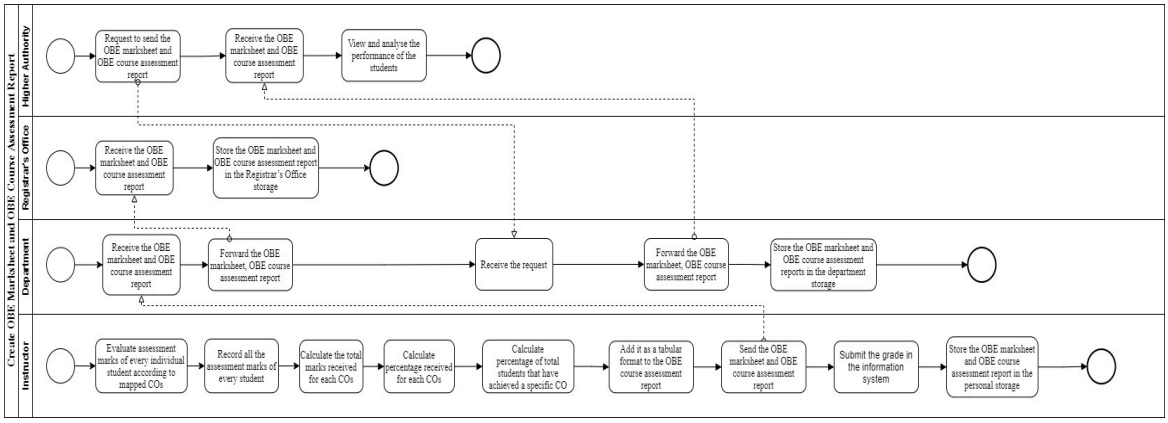


Figure 4: Create OBE Marksheet and OBE Course Assessment Report

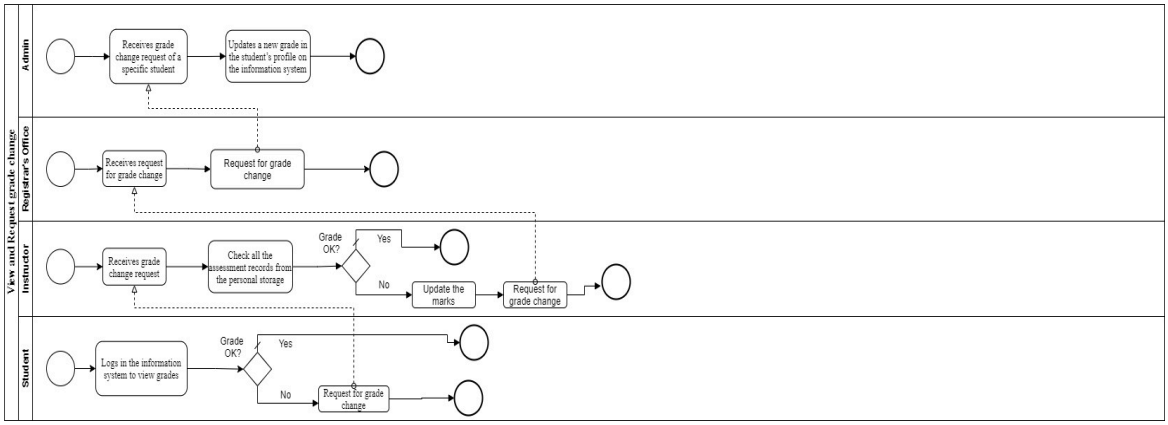


Figure 5: View and Request Grade Change

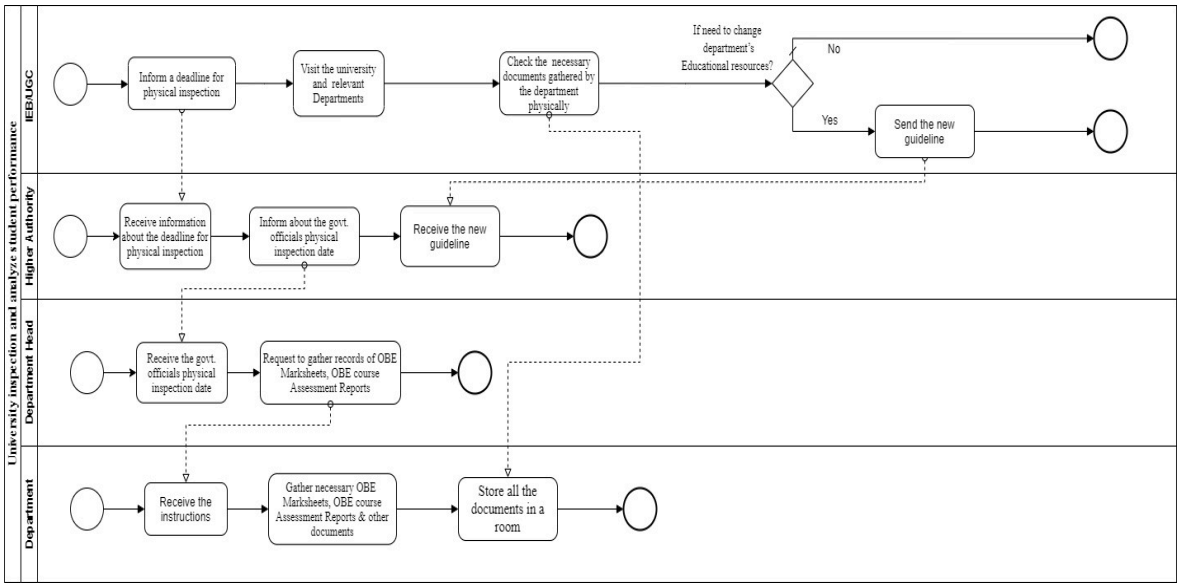


Figure 6: University Inspection and Analyze Student Performance

D. PROBLEM ANALYSIS – EXISTING BUSINESS SYSTEM:

Process	Stakeholders	Problems	Analysis	Proposed Solution
Mapping of COs from PLOs	1. UGC 2. IEB 3. Higher Authority 4. Department Head 5. Department 6. Instructor	1. Instructor does the PLO CO mapping manually. 2. Physically lists course contents, COs and course assessments.	1. This takes too much time to do manually. 2. It is a problem to handle all the PLOs and COs as it can get disorganized. 3. Requires too much non computing hardware. 4. Verification issues.	1. Our system will have all necessary details about the courses. Which includes: a) Level of the course. b) List of the courses in the OBE program. 2. Automatically map PLO and CO. Automatic system includes: a) Gives the option to choose the number of CO's and keywords. b) Instructor chooses the number of CO's and the required keywords under the CO. c) Automatically the mapping will be done.
Conduct Course assessments according to COs	1. Instructor 2. Student	1. Instructor Sets assessments according to mapped COs manually.	1. This takes too much time to do manually. 2. Requires too much non computing hardware. 3. It can get disorganized.	Our system will automatically map the assessments according to the COs.

Create OBE Marksheet and OBE Course Assessment Report	1. Instructor 2. Department 3. Higher Authority 4. Register's office	1. The Instructor evaluates assessment marks of every individual student according to mapped COs throughout the semester on softcopies and hardcopies manually. 2. Record all the assessment marks of every student for a course in a OBE marksheet by each assessment question under each CO manually. 3. Calculate the total marks received for each COs on the OBE marksheet physically. 4. Calculate percentage received for each COs from the OBE marksheet for each student physically. 5. Calculate percentage of total students that have achieved a specific CO and add as a tabular format to the OBE	1. This takes too much time to do manually. 2. Requires too much non computing hardware. 3. Risk of errors. 4. Chances of disorganized documents.	Our system will calculate assessment marks, percentages and generate charts showing the PLO CO achieved by each student who has taken the course in the given semester. The OBE course assessment report will be automatically generated.

		course assessment report physically.		
View and Request grade change	1. Student 2. Instructor 3. Register's office 4. Admin	1. When receiving a grade change request from the student, the instructor has to check all the assessment records from his/her personal storage manually. 2. If he/she needs to update marks in his/her personal storage to change the grade he/she has to inform the Registrar's office. 3. The Register's office informs the Admin and the Admin change the grade in the information system.	1. This takes too much time to do manually. 2. Requires too much non computing hardware. 3. Risk of errors. 4. Instructor has no direct access to the information system.	Our system will give direct access to the instructor in the system so that if he/she wants to change the grade of a student. Also, the Instructor will be able to store data in the system directly.

University inspection and Analyze student performance	1. UGC 2. IEB 3. Department Head 4. Department	1. UGC/ IEB have to come to the university physically for inspection. 2. They have to go through all the documents manually for a quality check. 3. The Department has to gather all the documents manually and put them in a room for inspection.	1. This takes too much time to do manually. 2. Requires too much non computing hardware. 3. Verification issues. 4. Chances of disorganized documents.	Our system will generate a performance trend for selective courses. UGC/ IEB don't need to visit physically for inspection as they will be able to get necessary information directly from the system.

E. RICH PICTURE – PROPOSED SYSTEM:

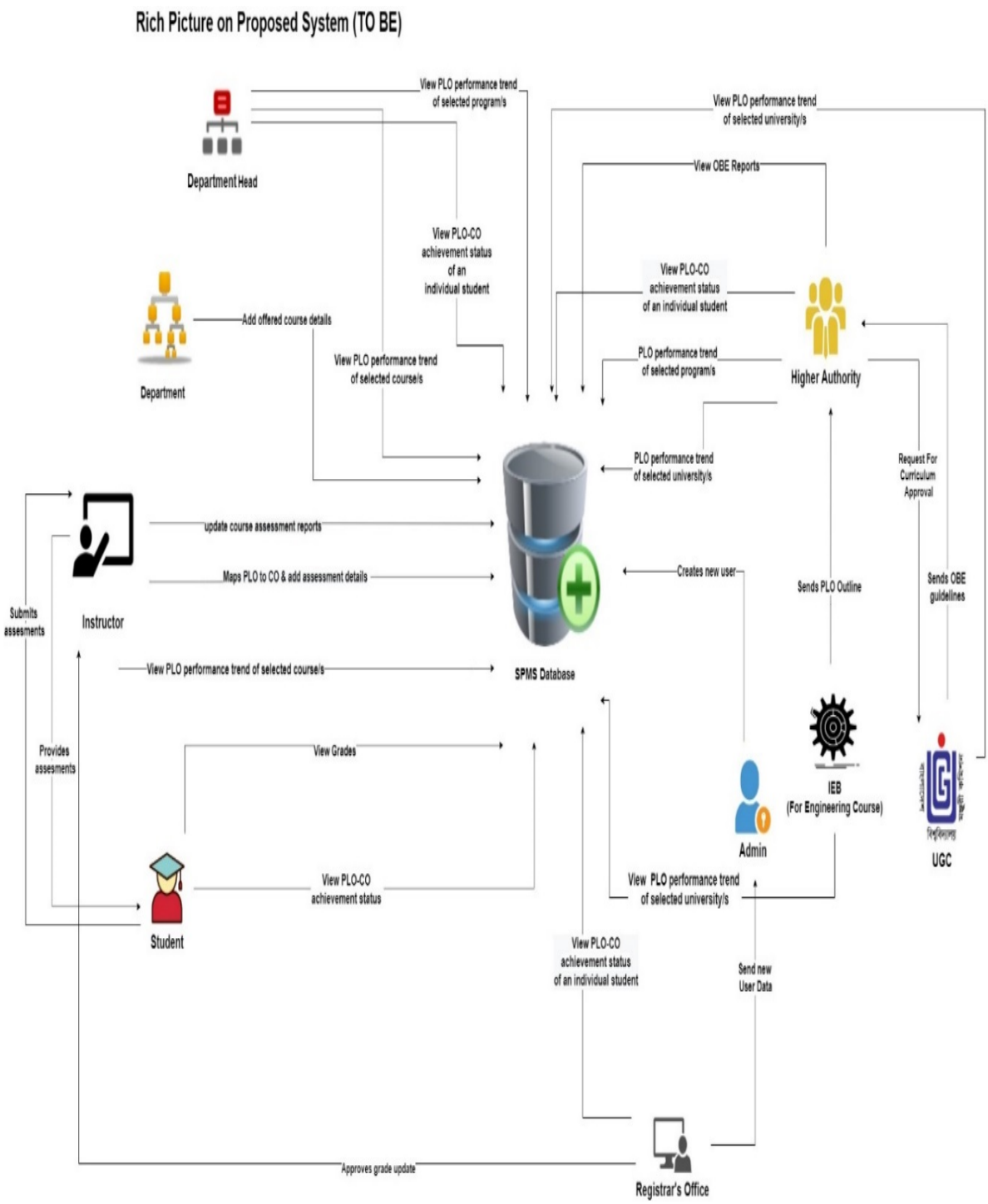


Figure 7: Rich Picture of Proposed System

F. SIX ELEMENT ANALYSIS – PROPOSED SYSTEM:

Process	System Role					
	Human	**Non-Comp Hardware**	**Computing Hardware**	**Software**	**Database**	**Network and Communication**
View PLO achievement stats of an individual student	**Student** 1) Logs into the system using User ID, password, from the dropdown select group and University name. 2)From the sidebar the user can select from the following options: i. Program wise PLO Achievement. ii. Course wise PLO Achievement iii. Failed and Achieved PLO Table. 3) If the user clicks on "Program **wise PLO Achievement** ": i) The user doesn't need to provide any further input to the system or click the display button	N/A	**Computer or laptop** 1)Users will need a computer to access the system. **Printer:** 1)Users can print out the document if needed. **Networking Devices (Router, Modem, Switch, Bridge, Hub):** 1)Used to Access the Internet.	**SPMS** 1)The software will generate a chart showing the PLO achievement status of an individual student for a selective course.	**SPMS** 1)The PLO achievement will be stored and updated here.	**Internet** 1)It is used to login into and access the system.

	because the database will have info of the Programs the user is under. ii) The system will process the request from the user. If there is any error in the request, then the system will notify the user of the error. Otherwise, the system will return the requested view of the chart showing statistically analyzed percentage score in each PLO against a program average. 4) If the user clicks on "**Course Wise PLO Achievement** ": i) Enter course ID from the dropdown list as the system will already have a record of all the courses taken by any student. ii) Click on "Search" iii) The					

	system will process the request from the user. If there is any error in the request, then the system will notify the user of the error. Otherwise, the system will return the requested view of the chart showing statistically analyzed percentage score in each PLO in a selected course against the average course. 5)If the user clicks on "**All failed and Achieved PLOs**": i) The user doesn't need to provide any further input to the system or click the display button because the student has selected to view PLO achievement of all the courses taken by, he/she so far. ii) The system will then process the					

	request and return the requested view for a table showing all the PLOs achieved and failed to achieve for all the courses taken so far by a specific student. **Department Head/ Registrar's Office** 1) Logs into the system using User ID, password, from the dropdown select group and University name. 2) The user needs to click on the dropdown button **"PLO Stats of a Student"** from the sidebar and see the following options: i. Course wise. ii. Program wise. iii. All Achieved and Failed PLOs. 3) If the user clicks on "**Course Wise**":					

	i) Enter student ID in the text field. ii) Enter course ID from the dropdown list. iii) Click on "Enter" iv) The system will process the request from the user. v) If the selected course is not taken by the entered studentID or if that specific student ID is not a valid ID, then the system will notify you is not a valid ID then the system will notify you about the incorrect information in your request. vi)Otherwise the system will return the requested view of the chart showing statistically analyzed percentage score in each PLO against a program average. 4) If the user clicks on "**Program wise**"					

	i) Enter student ID in the text field. ii) Enter the program name from the dropdown list. iii) Click on "Search" iv) The system will process the request from the user. v) If the entered studentID is not a part of that specific program or if that student ID is not a valid ID then the system will notify you about the incorrect information in your request. vi)Otherwise the system will return the requested view of the chart showing statistically analyzed percentage score in each PLO against a program average. 5)If the user clicks on "**All Achieved and Failed PLOs**" i) Enter student ID in the text field.					

	ii) Enter "Search" iii) The system will then process the request and if there is any error the user will be notified. iv) Otherwise, the system will return the requested view for a table showing all the PLOs achieved and failed to achieve for all the courses taken so far by an individual student.					
View Semester wise PLO Performance trend of selected courses	**Department Head:** 1) Logs into the system using User ID, password, from the dropdown select group and University name. 2) Click on the drop-down button called "**Course wise PLO trend** "from the sidebar. 3) Select from the three categories: i) Course Wise	N/A	**Computer or laptop:** 1)Users will need a computer to access the system. **Printer:** 1)Users can print out the document if needed. **Networking Devices (Router, Modem, Switch, Bridge, Hub):** 1)Used to	**SPMS:** 1)The software will generate a chart comparing the PLO achieved percentage for each PLO among the instructors who have taken the course in the given semester. **Operatin**	**SPMS:** 1)Use the database to obtain the performance.	**Internet:** 1)It is used to login into and access the system.

	ii) PLO Wise Achievement iii) Instructor Wise for a Course 3) If the user selects "**Course wise":** i) Enter From Semester from the dropdown list. ii) Enter To Semester from the dropdown list. iii) Click on "Enter". iv) The system will process the request from the user. v) The system will return the requested view for a chart showing the percentage of students who achieved each PLOs and who failed in the selected semester. 4) If the user selects "**PLO wise Achievement** ": i) Select one/more PLOs from the drop-down		Access the Internet.	**g Software :** Used by the User to run the system.		

	list. ii) Enter From Semester from the dropdown list. iii) Enter To Semester from the dropdown list. iv) Click on "Search". v) The system will process your request and return the requested view for a comparison of PLO achievement percentage among courses which have the same PLO/s that was/were selected. 5) If the user selects "**Instructor Wise for a Course**". i) Enter the Course from the dropdown list. ii) Enter From Semester from the dropdown list. iii) Enter To Semester from the dropdown list. iv) Click on "Enter"					

	v)The system will process the request from the user. vi)The system will then return the requested view of a chart comparing the PLO achieved percentage for each PLO among the instructors who have taken the course. **Instructor:** 1) Logs into the system using User ID, password, from the dropdown select group and University name. 2) Select Instructor Wise PLO Analysis from the sidebar 3) Enter the Course from the dropdown list. 4) Enter From Semester from the dropdown list. 5) Enter To Semester from the dropdown list. 6) Click on "Enter" 7) The system					

	will process the request from the user 8)The system will then return the requested view of a chart comparing the PLO achieved percentage for each PLO among the instructors who have taken the course.					
View PLO Performance Trend of Selected Program/s	**Department Head/ Higher Authority:** 1) Logs into the system using User ID, password, from the dropdown select group and University name. 2) Select the dropdown button "**Program wise PLO Comparison**" from the sidebar 3) The user can see two further options: i. PLO Achievements ii. Student PLO comparison	N/A	**Computer or laptop:** 1)Users will need a computer to access the system. **Printer:** 1)Users can print out the document if needed. **Networking Devices (Router, Modem, Switch, Bridge, Hub):** 1)Used to Access the Internet.	**SPMS:** 1) The software will generate a performance trend for a chosen program	**SPMS:** 1)Use the database to obtain the performance.	**Internet:** 1)It is used to login into and access the system.

	4)If the user selects "**PLO Achievements** ": i. Select one or more programs from the dropdown list. ii. Enter from Semester from the dropdown list. iii. Enter to Semester from the dropdown list. iv. Click on "Search" v. The system will process the request from the user. vi. The system will then return the requested view for a chart showing the count of students who achieved each PLO, segmented with color-code into the percentage of the count that came from the courses that have that PLO. vi. The user can click on any of the PLOs on the chart. vii. Upon clicking the					

	PLO/s the PIE CHART below will be updated. viii. The pie chart will display a clearer segmentation for the selected PLO. 5) If the user selects "**Student PLO comparison":** i. Select one or more programs from the dropdown list. ii. Enter from Semester from the dropdown list. iii. Enter to Semester from the dropdown list. iv. Click on "Search" v. The system will process the request from the user. vi. The system will then return the requested view for a chart showing the count of students who attempted each PLO against that of those who achieved.					

View University wise PLO Performance Trend	**UGC/IEB/ Higher Authority:** 1) Logs into the system using User ID, password, from the dropdown select group and University name. 2) Select the drop-down button "**University wise PLO Trend**" from the sidebar. 3) The user will see three further options: i. Program PLO Comparison. ii. Graduates PLO Achievement iii. All PLO performance 4) If the User selects "**Program PLO Comparison**" **:** i. Select one program from the dropdown list. ii. Enter from Semester from the dropdown list. iii. Enter to Semester from the dropdown	N/A	**Computer or laptop:** 1)Users will need a computer to access the system. **Printer:** 1)Users can print out the document if needed. **Networking Devices (Router, Modem, Switch, Bridge, Hub):** 1)Used to Access the Internet	**SPMS:** 1) The software will generate a performance trend for a chosen program in a university .	**SPMS:** 1)Use the database to obtain the performance.	**Internet:** 1) It is used to login into and access the system.

	list. iv. Click on "Search" v. The system will process the request from the user. vi. The system will then return the requested view for a radar chart showing the PLO achieved count comparison for each PLO within a chosen time frame. 5) If the User selects "**Graduates PLO Achievement ":** i) Select one or more programs from the drop-down list. ii) Enter "Search" iii)The system will process the request from the user. iv) The system will then return the requested view for a radar chart comparing the percentage of graduates who have					

	achieved all PLOs of the chosen programs. 6) If the User selects "**All PLO performance ":** i. Select one or more PLOs from the dropdown list. ii. Enter "Search" iii. The system will process the request from the user. iv) The system will then return a comparison of the percentage of students who achieved that/those chosen PLO/s (percentage derived from total attempted vs achieved).					

G. PROCESS MODEL – PROPOSED SYSTEM:

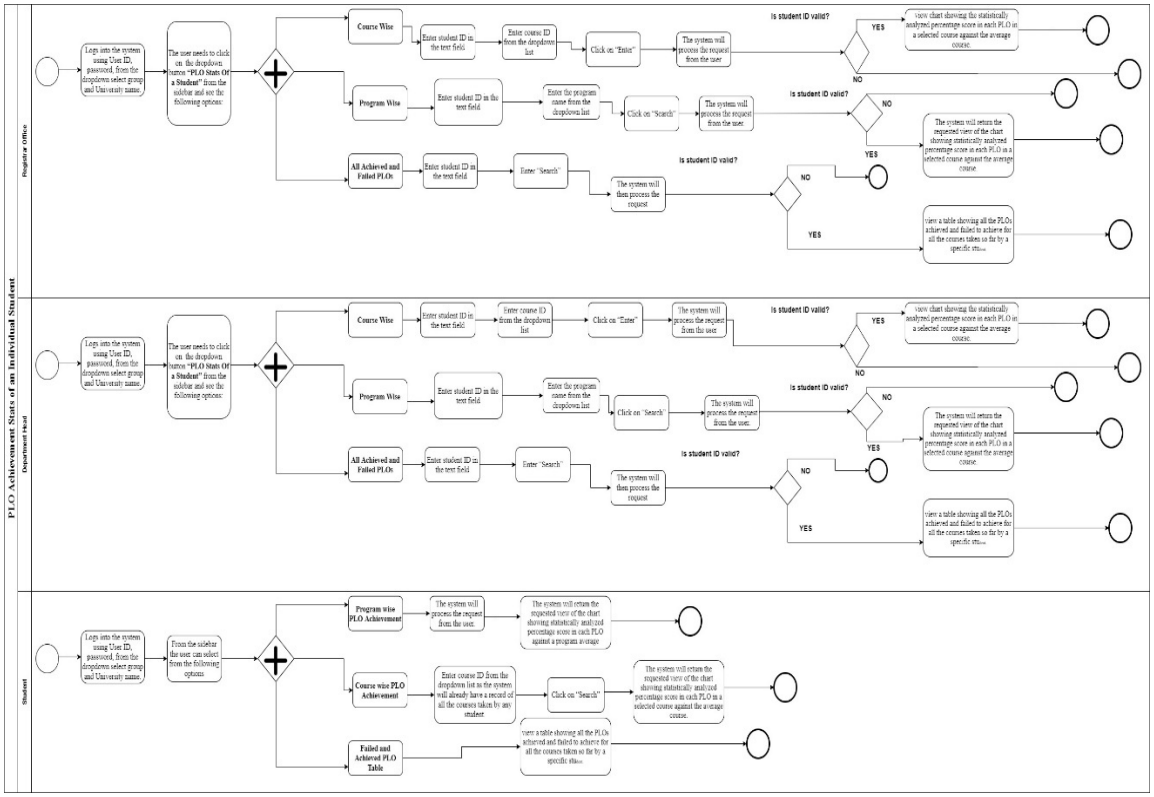


Figure 8: PLO Achievement Stats of an Individual Student

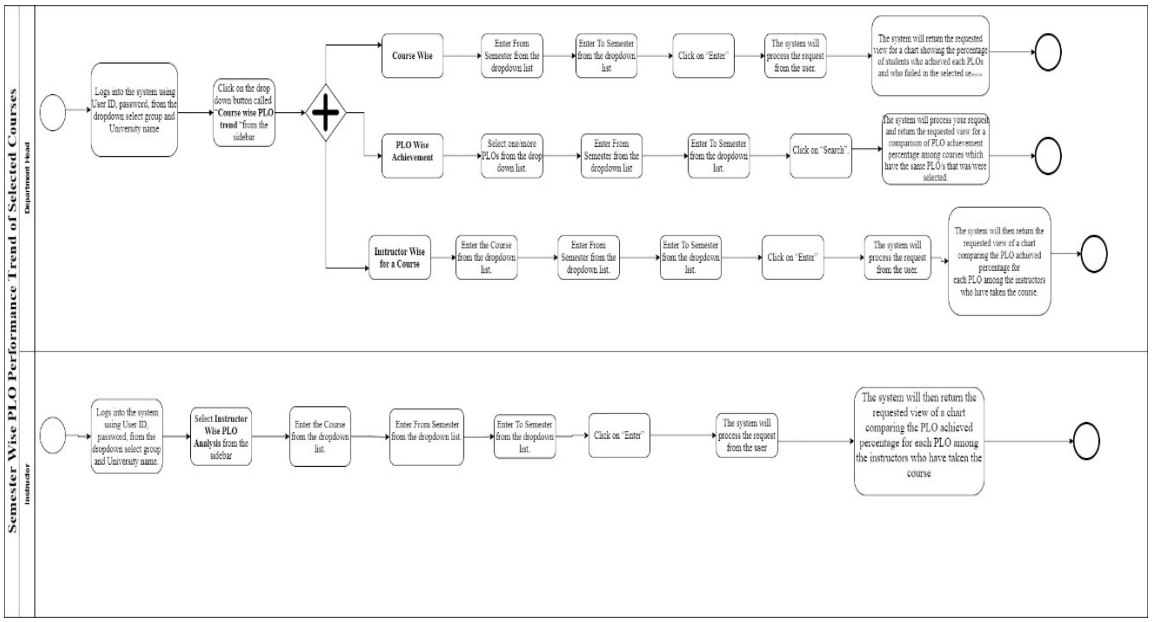


Figure 9: Semester Wise PLO Performance Trend of Selected Courses

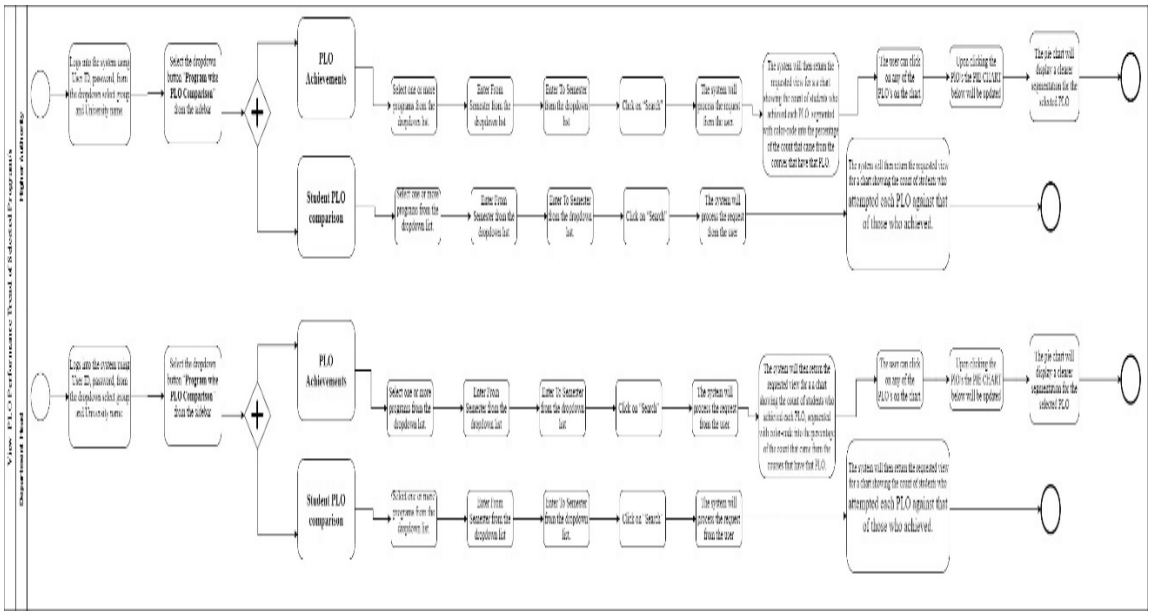


Figure 10: View PLO Performance Trend of Selected Program/s

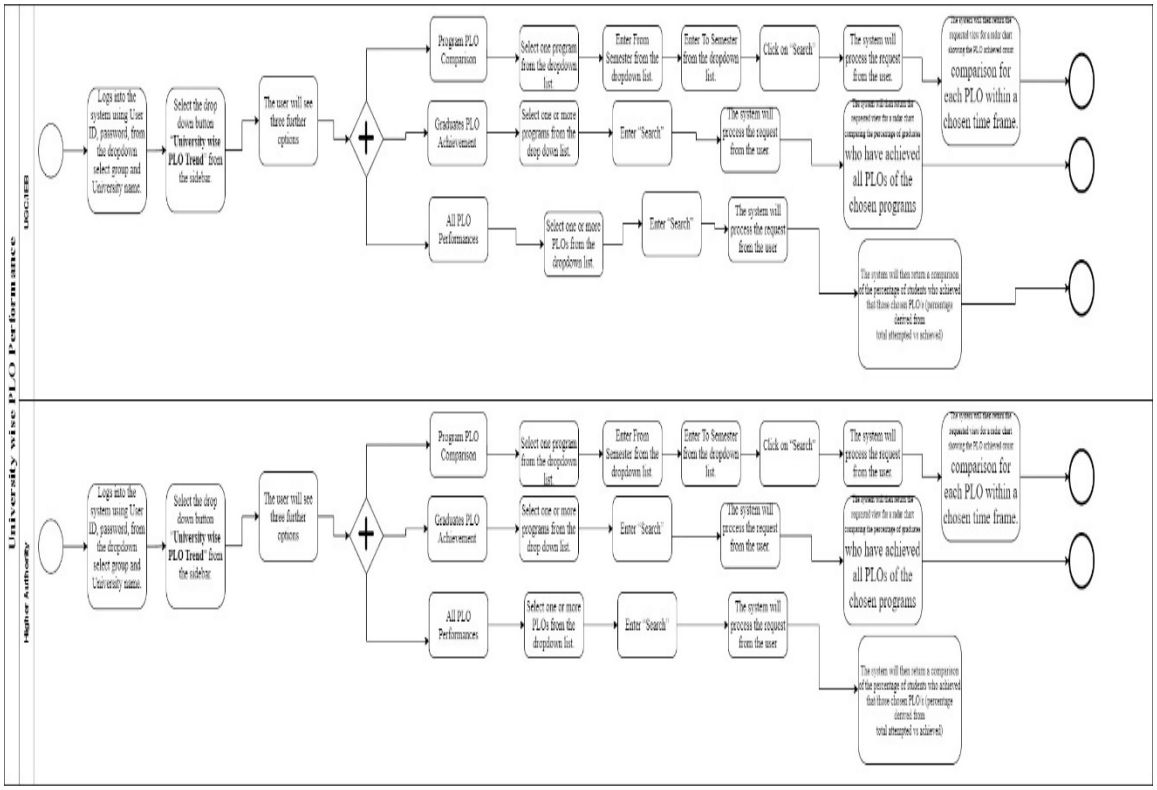


Figure 11: University Wise PLO Performance

CHAPTER 3 – LOGICAL SYSTEM DESIGN:

In this chapter, we will create a data model for the data that will be stored in a database using our proposed system. Essentially, the data model is a conceptual representation of Data objects, the associations between them, and the rules. The purpose of data modelling is to visualize data and enforce business rules, regulatory compliances, and government policies on the data. While ensuring the quality of the data, Data Models ensure consistency in naming conventions, default values, semantics, and security. As part of our proposal, we will be designing a better representation of the data.

A. BUSINESS RULE [SPMS]:

The objective of the system is to expand productivity in monitoring the student's performance. The framework is the place where all the PLO (Program Learning Outcome) and CO (Course Outcome) are put away. In this Student monitoring requirement system-centred approach at learning and teaching, assessment shapes the curriculum. Formulated learning outcomes should cover what matters in terms of achievements that can be assessed and demonstrated by students in the program and practice.

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 University contains UniversityID, UniversityName, UniversityAddress.
2. A department must contain one school. A School must contain one or many departments. A school includes SchoolID, SchoolName.
3. A Department must manage one or many Department heads. A department head must manage one department. A department head includes DepartmentID, StartDate, EndDate.
4. A program must belong to one department. A department must belong to one or many programs. A department contain DepartmentID, DepartmentName, SchoolID.
5. An Account has four sub-types (Student, Faculty, Head). An Account includes AccountID, Name, Gender, Email, Phone, AccountAddress, AccountType.
6. A student must have one department. A STUDENT has StudentID, FirstName, LastName, DateofBirth, Gender, Email, Phone, Address, EnrollmentDate. A department must have many students.
7. AnInstructor must have one Department. A department must have one or many Instructors. AnInstructor includes DepartmentID, Rank, JoinDate.An Instructor may teach many sections. A section must be taught by one Instructor.

8. Students may perform many registrations. A REGISTRATION includes RegistrationID, Semester, Year, Section Id, StutendID. Registration must be performed by at least one student.
9. A section mandatorily has many registrations. A registration has at least one section. A section includes SectionID, SectionNum, CourseId, FacultyID, Semester, Year.
10. A registration may belong to many EVALUATIONS. An evaluation mandatorily belongs to one registration. An evaluation contains EvaluationID, ObtainedMarks, AssessmentID, RegistrationID.
11. An evaluation must have one assessment. An Assessment must have many evaluations. Assessments contains AssesmentsID, AssessmentName, TotalMarks, SectionID, COID. An assessment must contain one section. A section contains one or many assessments.
12. An assessment must map with one CO's. A CO's maps with one or many assessments. A CO's includes COID, CourseID, 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.
13. 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.
14. 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.

B. ENTITY RELATIONSHIP DIAGRAM (ERD):

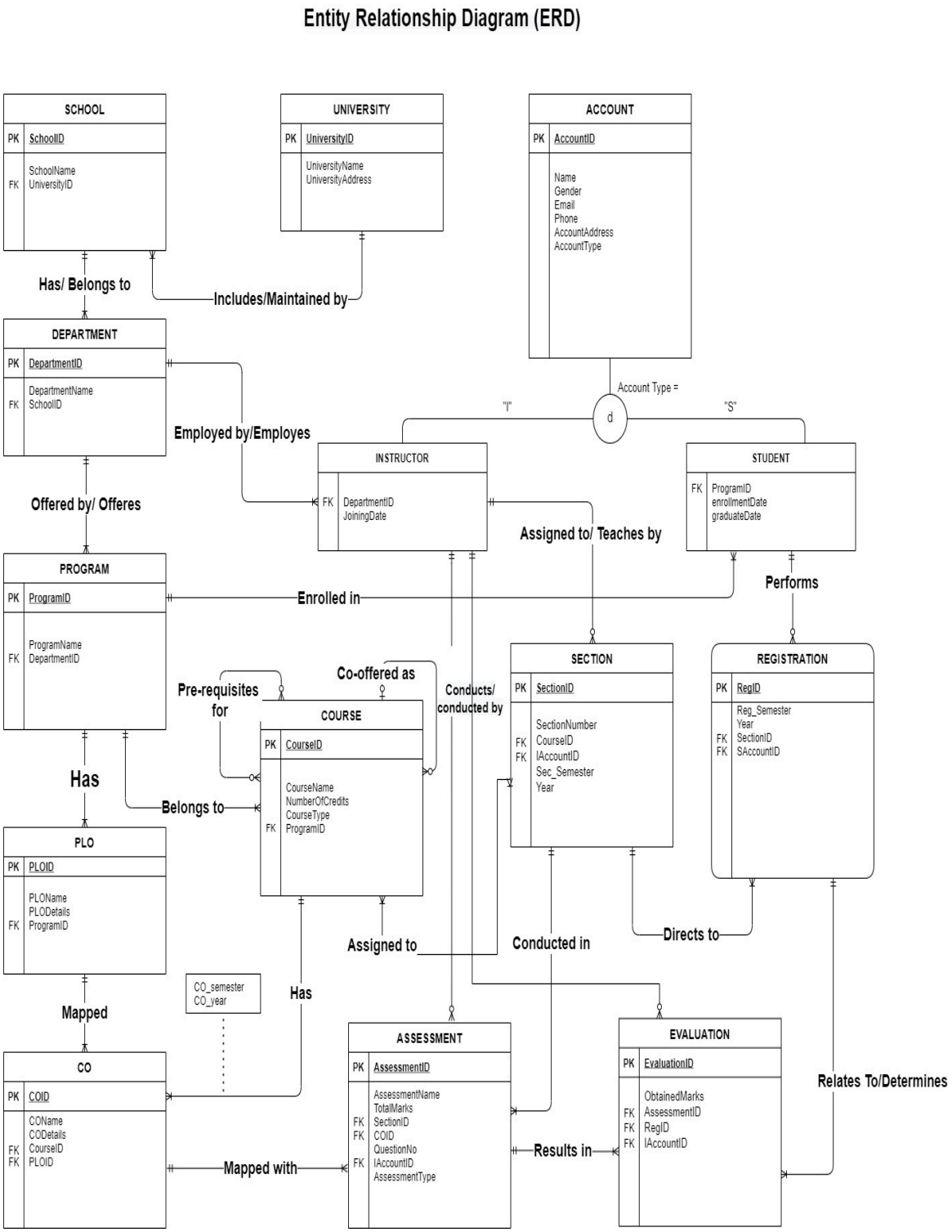


Figure 12: Entity Relationship Diagram

C. ENTITY RELATIONSHIP DIAGRAM TO RELATIONAL SCHEMA:

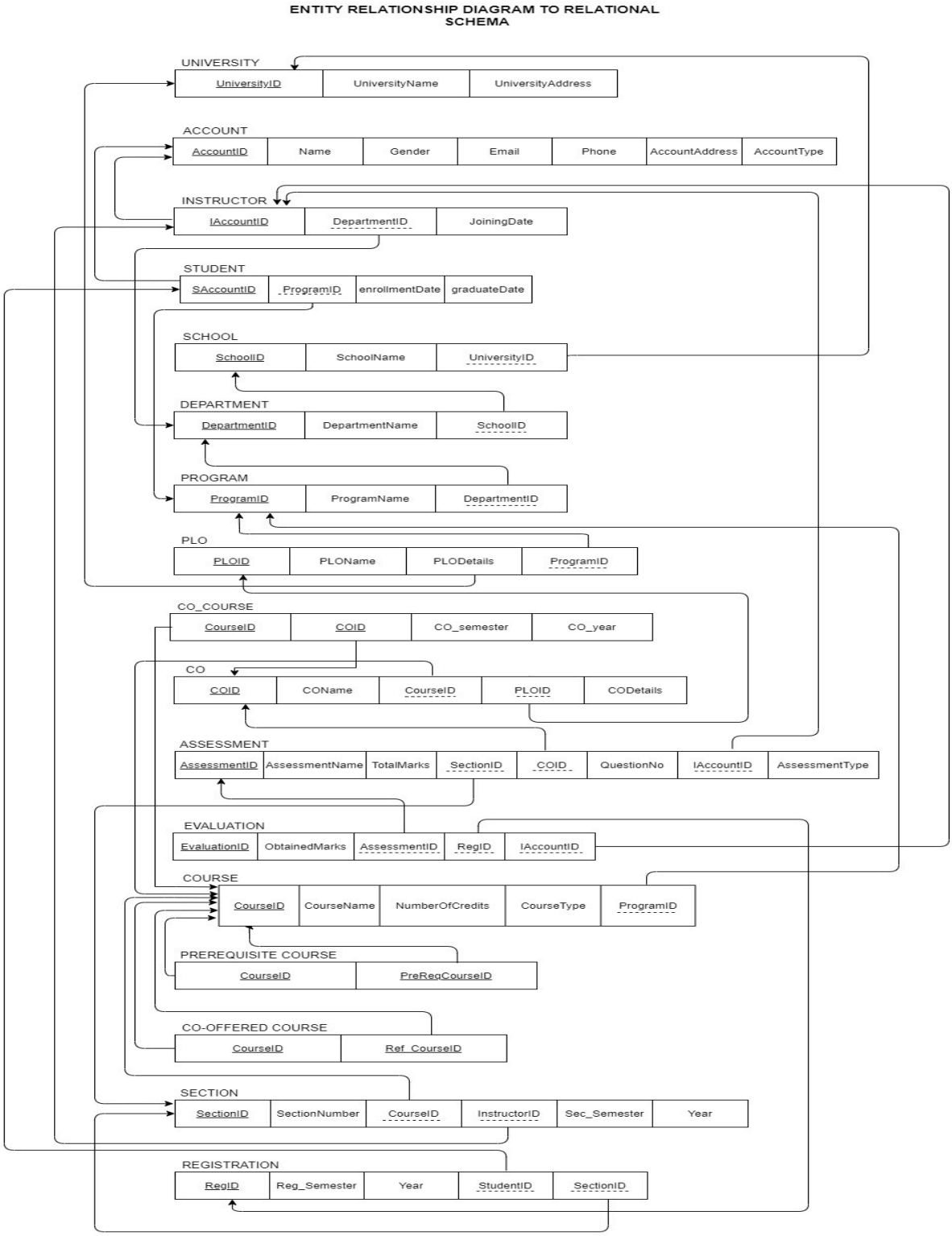


Figure 13: Entity Relationship Diagram to Relational Schema

D. NORMALIZATION:

Account	AccountID	a1	Program	ProgramID	p1
	Name	a2		ProgramName	p2
	Gender	a3		DepartmentID	d1
	Email	a4	PLO	PLOID	pl1
	Phone	a5		PLOName	pl2
	AccountAddress	a6		PLODetails	pl3
	AccountType	a7		ProgramID	p1
Instructor	IAccountID	i1	CO	COID	co1
	DepartmentID	d1		CONum	co2
	JoiningDate	i2		CODetails	co3
Student	SAccountID	st1		CourseID	c1
	ProgramID	p1		PLOID	pl1
	enrollmentDate	st2			
	graduateDate	st3			
University	UniversityID	u1	Assessment	AssessmentID	as1
	UniversityName	u2		AssessmentName	as2
	UniversityAddress	u3		TotalMarks	as3
School	SchoolID	sc1		SectionID	s1

	SchoolName	sc2		COID	co1
	UniversityID	u1		QuestionNo	as4
				IAccountID	i1
				AssessmentType	as5
Department	DepartmentID	d1	Evaluation	EvaluationID	ev1
	DepartmentName	d2		ObtainedMarks	ev2
	SchoolID	sc1		AssessmentID	as1
Course	CourseID	c1		RegID	rg1
	CourseName	c2		IAccountID	i1
	NumberOfCredits	c3	Prerequisite Course	CourseID	c1
	CourseType	c4		PreReqCourseID	pr1
	ProgramID	p1	Co-Offered Course	CourseID	c1
Section	SectionID	s1		Ref_CourseID	ff1
	SectionNumber	s2	Registration	RegID	rg1
	CourseID	c1		Reg_Semester	rg2
	InstructorID	i1		Year	rg3
	Sec_Semester	s3		StudentID	st1
	Year	s4		SectionID	s1
Co_Course	CourseID	c1			

	COID	co1			
	CO_semester	cc1			
	CO_year	cc2			

a1 →	a2, a3, a4, a5, a6, a7	pl1→	pl2, pl3, p1
i1 →	d1, i2	co1→	co2, co3, c1, pl1
st1→	p1, st2, st3	as1→	as2, as3, s1, co1, as4, i1, as5
u1→	u2, u3	ev1→	ev2, as1, rg1, i1
sc1→	sc2, u1	c1→	pr1
d1→	d2, sc1	c1→	ff1
c1→	c2, c3, c4, p1	rg1→	rg2, rg3, st1, s1
p1→	p2, d1	c1, co1→	cc1, cc2
s1→	s2, c1, i1, s3, s4		

AccountID→	Name, Gender, Email, Phone, AccountAddress, AccountType
IAccountID →	DepartmentID, JoiningDate
SAccountID→	ProgramID
UniversityID→	UniversityName, UniversityAddress
SchoolID→	SchoolName, UniversityID
DepartmentID→	DepartmentName, SchoolID
CourseID→	CourseName, NumberOfCredits, CourseType, ProgramID
ProgramID→	ProgramName, DepartmentID
SectionID→	SectionNumber, CourseID, InstructorID, Sec_Semester, Year
PLOID→	PLOName, PLODetails, ProgramID

COID→	COName, CODetails, CourseID, PLOID
AssessmentID→	AssessmentName, TotalMarks,SectionID, COID, QuestionNo
EvaluationID→	ObtainedMarks, AssessmentID, RegID
CourseID→	PreReq, CourseID
CourseID→	CoOffered, CourseID
RegID→	Reg_Semester,Year, StudentID, SectionID

1NF:

primary key => ev1

1NF:

ev1	pr1	ff1	a4	a5	a6	a7	i1	i2	st1	st2	st3	u1	u2	u3	p1	p2	pl1	pl2	pl3
co1	co2	co3	as1	as2	as3	as4	as5	sc1	sc2	d1	d2	c1	c2	c3	c4	s1	s2	s3	s4
a1	ev2	a2	a3	rg1	rg2	rg3	cc1	cc2											

Figure 14: 1NF

2NF:

As we have only one primary key which is 'ev1', we don't need to apply 2NF.

3NF:

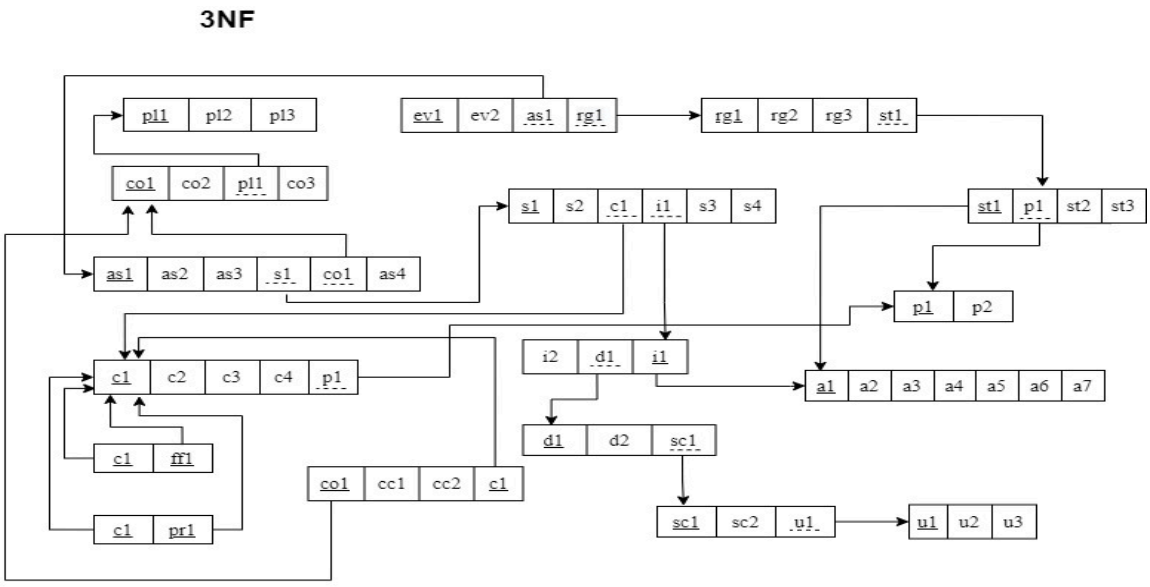


Figure 15: 3NF

BCNF:

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

E. DATA DICTIONARY:

Account_T

Name	Data Type	Size	Remarks
AccountID	VARCHAR	5	This is the primary key of the Account table. For example: “1234567”
CName	VARCHAR	50	This is for the name of the account holder in the Account table. For example: “Brad Pitt”.
Gender	CHAR	50	This is for the gender of the account holder in the Account table. For example: “Male”
CEmail	VARCHAR	50	This is the email address of the account holder in the Account table. For example: “acotorBrad@gmail.com”.
NPhone	INTEGER	20	This is the phone number of the account holder in the Account table. For example: “+88-01700000”.
CAccountAddress	VARCHAR	50	This is the account address of the account holder in the Account table.
CAccountType	VARCHAR	45	This is the account type of the account holder in the Account table. For example: “Dean”

University_T

Name	Data Type	Size	Remarks
cUniversityID	VARCHAR	5	This is the primary key of University. E.g.: "IUB"
cUniversityName	VARCHAR	50	This is the name of the University. E.g.: "Independent University, Bangladesh".
cUniversityAddress	VARCHAR	50	This is the University address. E.g.: "123-Aftab Road, Bashundhara"

School_T

Name	Data Type	Size	Remarks
cSchooID	VARCHAR	5	This is the primary key of School. E.g.: "SETS"
cSchoolName	VARCHAR	50	This is the name of the School. E.g.: "School of Engineering, Technology & Science".
cUniversityID	VARCHAR	5	This is the foreign key from the University table. E.g.: "IUB"

Program_T

Name	Data Type	Size	Remarks
cProgramID	INTEGER		This is the primary key for a program. E.g.: "1"

cProgramName	VARCHAR	50	This is the name of the program. E.g.: "Bachelor of Science"
cDepartmentID	VARCHAR	3	This is the foreign key from the Department table. E.g.: "CSE"

Department_T

Name	Data Type	Size	Remarks
cDepartmentID	VARCHAR	3	This is the primary key for the Department table. E.g.: "CSE"
cDepartmentName	VARCHAR	50	This is the name of the department. E.g.: "Computer Science and Engineering".
cSchoolID	VARCHAR	5	This is a foreign key from the School table. E.g.: "SETS".

Student_T

Name	Data Type	Size	Remarks
SAccountID	INTEGER		This is the primary key for the student table. E.g.: "1921834".
ProgramID	INTEGER		This is the foreign key from the Program table. E.g.: "1"

enrollmentDate	DATETIME	3	This is the enrollment date for students in the student table. E.g.: “12-Sep-2018”
graduateDate	DATETIME	3	This is the graduation date for students in the student table. E.g.: “12-Sep-2021”

CO_T

Name	Data Type	Size	Remarks
COID	VARCHAR	9	This is the primary key for the CO table. E.g.: “CO1”.
CONum	VARCHAR	11	This is the CO name. E.g.: co1
CODetails	VARCHAR	50	This is the CO details in the CO table. E.g.: “CO1 mapping PLO”
CourseID	VARCHAR	6	This is the foreign key from the Course table. E.g.: “CSE303”
PLOID	VARCHAR	5	This is the foreign key from the PLO table. E.g.: “PLO1”

PLO_T

Name	Datatype	Size	Remarks
cPLOID	VARCHAR	5	This is the primary key for Program Learning Outcome. E.g.: “PLO1”

cPLOName	VARCHAR		This is the PLO name. E.g.: “PLO1”
cPLODetails	VARCHAR	50	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	INTEGER		This is a foreign key from the Program table. E.g.: “1”

Course_T

Name	Datatype	Size	Remarks
cCourseID	VARCHAR	6	This is the Primary Key for the Course. E.g.: “CSE203”
cCourseName	VARCHAR	40	This is the name of the Course. E.g.: “Discrete Mathematics”
nNumberOfCredits	INTEGER		This is the number of credits for the Course. E.g.: “3”
cCourseType	VARCHAR	10	This is the type of the Course. E.g.: “Core”
cProgramID	INTEGER		This is the foreign key from the program table. E.g.: “1”

Section_T

Name	Datatype	Size	Remarks
nSectionID	INTEGER		This is the Primary Key for Section. E.g.: “1”
nSectionNumber	INTEGER		This is the section number. E.g.: “1”
cCourseID	VARCHAR	6	This is the foreign key from the Course table. E.g.: “CSE101”
cInstructorID	NUMERIC	4	This is the foreign key from the Instructor table. E.g.: “1801”
cSec_Semester	VARCHAR	6	This is the semester of the section. E.g.: “Summer”
cYear	INTEGER		This is the year for the semester of the section. E.g.: “2021”

Registration_T

Name	Datatype	Size	Remarks
nRegID	INTEGER		This is the Primary Key for Registration. E.g.: “0101010101”
cReg_Semester	VARCHAR	6	This is the semester of registration. E.g.: “Spring”
dYear	YEAR	yyyy	This is the year of registration. E.g.: “2019”

nStudentID	INTEGER		This is the Foreign key from the Student Table. E.g.: “1800001”
nSectionID	INTEGER		This is the Foreign Key from Section table E.g.: “1”

Assessment_T

Name	Datatype	Size	Remarks
nAssessmentID	INTEGER		This is the Primary Key for Assessment.
cAssessmentName	VARCHAR	30	This is the name of the assessment. E.g.: “Mid”
cTotalMarks	NUMBER		This is the total marks of the assessment. E.g.: “30”
nSectionID	INTEGER		This is the Foreign Key from Section table.
nCOID	INTEGER		This is the Foreign Key from the Course Outcome table.
nQuestionNo	INTEGER		This is the question number for assessment. E.g.: “1,2,3….”
IAccountID	CHAR	11	This is the Foreign Key from the Instructor table.

AssessmentType	VARCHAR		This is the assessment type in the Assessment table. E.g.: “Sec

Evaluation_T

Name	Datatype	Size	Remarks
nEvaluationID	INTEGER		This is the Primary Key for Enrollment.
cObtainedMarks	NUMBER		This is the obtained marks of the student. E.g.: “26.9”
cAssessmentID	INTEGER		This is the foreign key from the assessment table.
nRegID	INTEGER		This is the Foreign Key from Registration table.
IAccountID	VARCHAR	11	This is the primary key for the Instructor table. E.g: “1234562”

Instructor_T

Name	Datatype	Size	Remarks
IAccountID	CHAR	11	This is the primary key for the instructor table. E.g.: “1234562”
DDepartmentID	VARCHAR		This is the DepartmentID of the department the faculty belongs to. E.g.: “CSE”
dJoiningDate	DATE	dd-mm yyyy	This is the starting date. E.g.: “01-07-2021”

PreRequisiteCourse_T

Name	Datatype	Size	Remarks
cCourseID	INTEGER	6	This is the foreign key from the Course table. E.g.: “CSE303”
cPreReqCourseID	INTEGER	6	This is the foreign key from the Course table. E.g.: CSE301

CoOfferedCourse_T

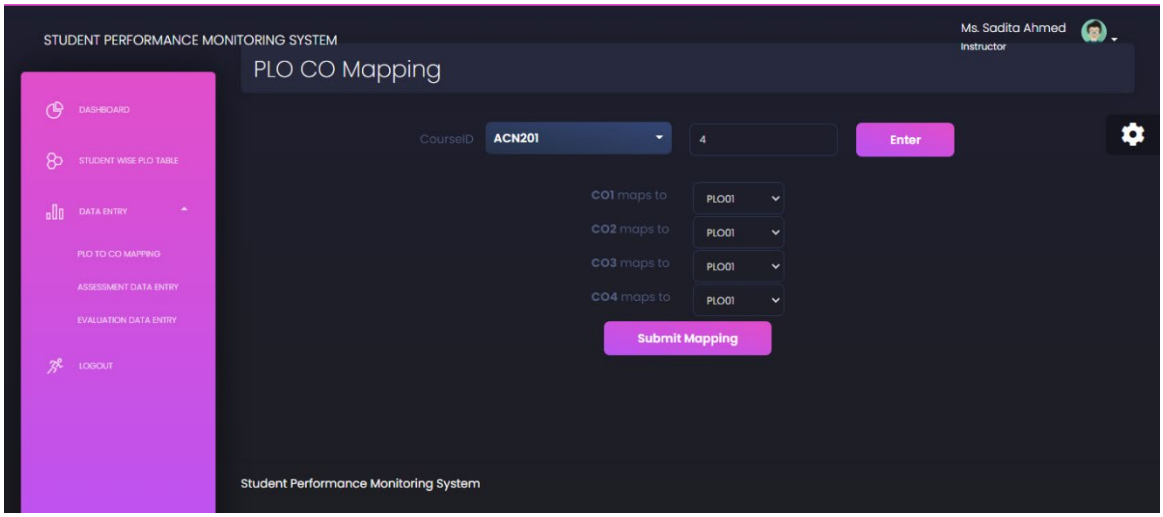
Name	Datatype	Size	Remarks
cCourseID	INTEGER	6	This is the foreign key from the Course table. E.g.: “CSE303”
cRef_CourseID	INTEGER	6	This is the foreign key from the Course table. E.g.: CSE301

CoCourse_T

Name	Datatype	Size	Remarks
cCourseID	INTEGER	6	This is the foreign key from the Course table. E.g.: “CSE303”
cCOID	INTEGER	6	This is the foreign key from the Course table. E.g.: CSE301
CO_semester	VARCHAR	40	This is the attribute stating CO in a semester for the relation in association with CO table and Course table. E.g.: “Spring 21”
CO_year	DATETIME	3	This is the attribute stating CO year for the relation in association with CO table and Course table. E.g.: “2021”

CHAPTER 4 – PHYSICAL SYSTEM DESIGN:

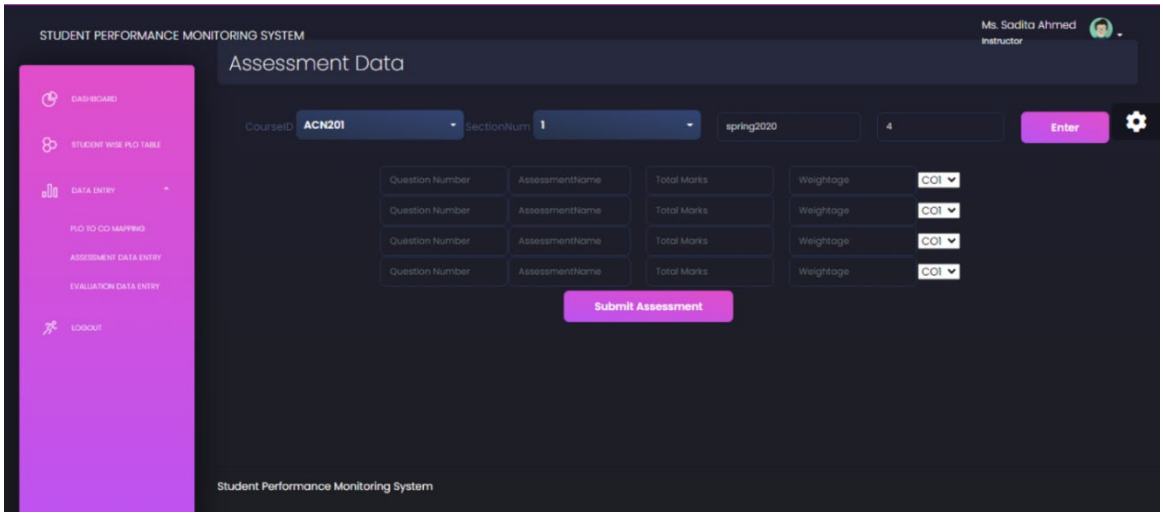
A. INPUT FORM:




```
def dataentry(request):
    name = request.user.get_full_name()
    type = request.user.groups.all()[0].name

    courselist = Course_T.objects.all()

    courses = []
    for c in courselist:
        courses.append(c.courseID)

    semesters = ["Spring", "Summer", "Autumn"]

    sections = [1, 2, 3]
    year = [2019, 2020]

    return render(request, 'dataentry.html', {
        'name': name,
        'usertype': type,
        'courses': courses,
        'semesters': semesters,
        'sections': sections,
        'year': year,
    })
def courseinfoentry(request):
    name = request.user.get_full_name()
    type = request.user.groups.all()[0].name
```

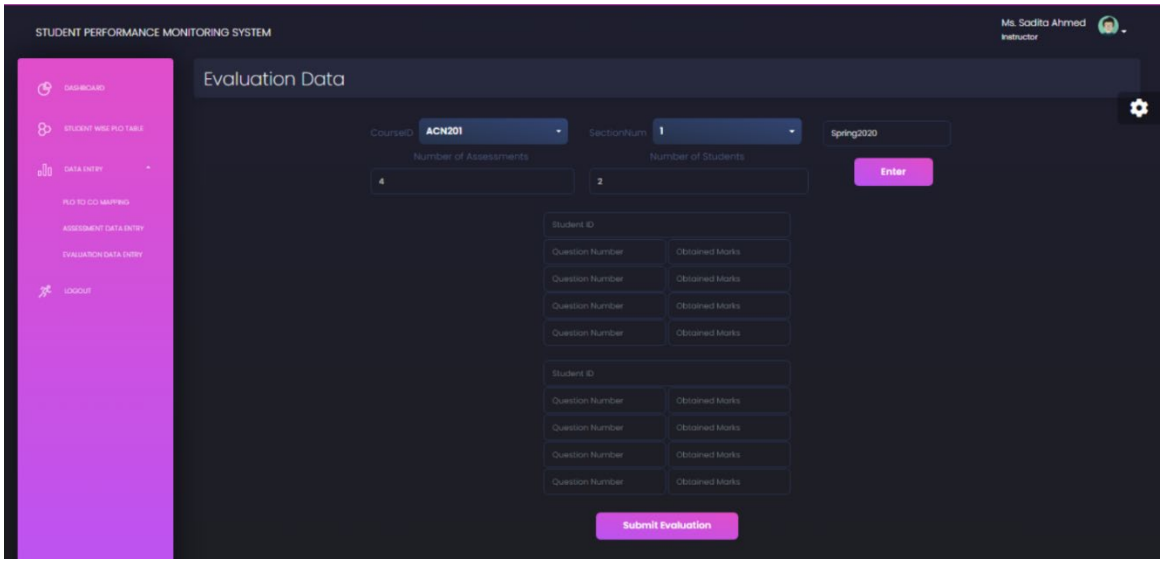
```
    courselist = Course_T.objects.all()
    courses = []
    for c in courselist:
        courses.append(c.courseID)
    semesters = ["Spring", "Summer", "Autumn"]
    sections = [1, 2, 3]
    year = [2019, 2020]
    return render(request, 'courseinfoentry.html', {
        'name': name,
        'usertype': type,
        'courses': courses,
        'semesters': semesters,
        'sections': sections,
        'year': year,
    })

def plocomapping(request):
    usertype = request.user.groups.all()[0].name
    if request.method == 'POST':
        courseID = request.POST.get('course')
        coMaps = request.POST.getlist('coMaps')

        course = Course_T.objects.get(pk=courseID)

        plist = PLO_T.objects.all()

        plolist = []

        for p in plist:
            if p.program_id == course.program_id:
                plolist.append(p)

        for i in range(len(coMaps)):
            k = -1

            for p in range(len(plolist)):
                if plolist[p].ploNum == coMaps[i]:
                    k = p
            co = CO_T(coNum="CO" + str(i + 1), course=course, plo=plolist[k
])
            co.save()
```

```
    return redirect('plocomapping')
    else:
        return render(request, 'plocomapping.html', {
            'usertype': usertype,
            'clist': courselist,

        })
```




```
def assessmentdataentry(request):
    usertype = request.user.groups.all()[0].name

    sections = [1, 2, 3]

    if request.method == 'POST':
        faculty_id = request.user.username
        course_id = request.POST.get('course')
        sectionNo = request.POST.get('section')
        semester = request.POST.get('semester')
        totalMarks = request.POST.getlist('totalMarks')
        weightage = request.POST.getlist('weightAge')
        assessmentName = request.POST.getlist('assessmentName')
        questions = request.POST.getlist('questions')
        cos = request.POST.getlist('co')
        section_id = None
        try:
            sections = Section_T.objects.raw('''
                SELECT *
                FROM spmapp_section_t s,
                    spmapp_course_t c
                WHERE s.course_id = c.courseNum
```

```
                    AND c.courseID = '{}'
                    AND sectionNum = '{}'
                    AND sec_semester = '{}'

            '''.format(course_id, sectionNo, semester))
            section_id = sections[0].sectionID
        except:
            section_id = None

        if section_id is None:
            section = Section_T(sectionNum=sectionNo, course_id=course_id,
                                instructor_id=faculty_id, sec_semester=seme
ster)
            section.save()
            section_id = section.sectionID

        for j in range(1, len(totalMarks) + 1):
            conum = cos[j-1]
            co_id = CO_T.objects.raw('''
                SELECT *
                FROM spmapp_co_t co,
                spmapp_course_t c
                WHERE co.course_id = c.courseNum
                    AND c.courseID = '{}'
                    AND co.coNum = '{}'
            '''.format(course_id, conum))

            assessment = Assessment_T(sectionID_id=section_id, coID_id=co_i
d[0].coID, totalMarks=totalMarks[j - 1],
                                      assessmentName=assessmentName[j-
1], questionNum=questions[j-1], weight=weightage[j-1])
            assessment.save()

        return redirect('assessmentdataentry')

    else:
        return render(request, 'assessmentdataentry.html', {
            'usertype': usertype,
            'clist': courselist,
            'semesters': semesters,
            'sections': sections,

        })
```

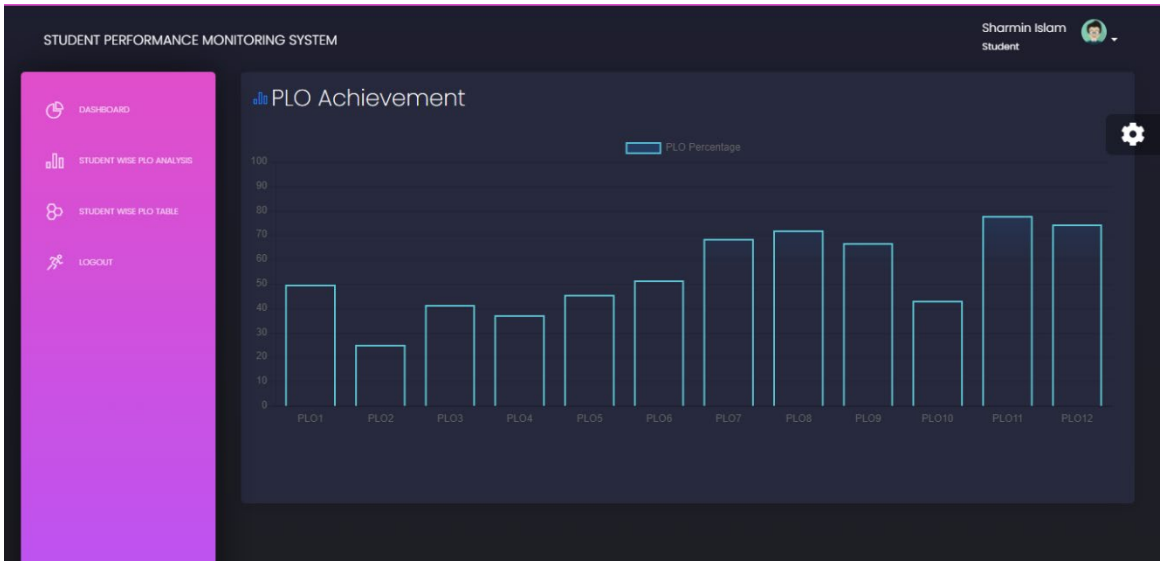



```python
def evaluationdataentry(request):
    usertype = request.user.groups.all()[0].name
    section = [1, 2, 3]

    if request.method == 'POST':
        course_id = request.POST.get('course')
        section = request.POST.get('section')
        semester = request.POST.get('semester')

        print(course_id)
        print(section)
        print(semester)

        student_id = request.POST.getlist('student')
        obtainedMarks = []
        questions = []
        for i in range(len(student_id)):
            obtainedMarks.append(request.POST.getlist(f'obtainedMarks{i}'))
            questions.append(request.POST.getlist(f'questions'))

        section_id = None
        try:
            section_id = Section_T.objects.raw('''
                 SELECT *
                FROM spmapp_section_t s,
                    spmapp_course_t c
                WHERE s.course_id = c.courseNum
                    AND c.courseID = '{}'
                    AND sectionNum = '{}'
```

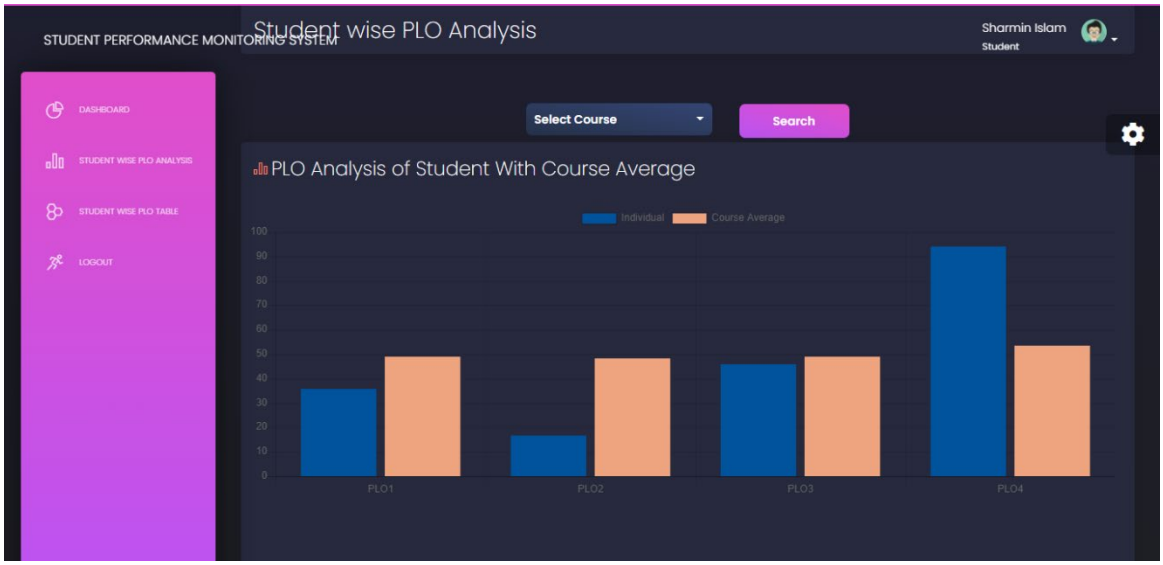
```
                AND sec_semester = '{}'
        '''.format(course_id, section, semester))
        section_id = section_id[0].sectionID
        print(section_id)
    except:
        section_id = None
    assessment_list = []
    coLength = 0
    try:
        col = CO_T.objects.raw('''
            SELECT count(*)
            FROM spmapp_co_t co,
                spmapp_course_t c
                WHERE co.course_id = c.courseNum
                    AND c.courseID = '{}'
            '''.format(course_id))
        coLength = col[0][0]+1
    except:
        coLength = 0
    for j in range(1, len(questions[0])+1):
        assessment_id = None
        try:
            assessment_id = Assessment_T.objects.raw('''
                SELECT *
                FROM spmapp_assessment_t
                WHERE sectionID_id = '{}'
                    AND coID_id IN (
                    SELECT coID
                    FROM spmapp_co_t co,
                        spmapp_course_t c
                    WHERE co.course_id = c.courseNum
                        AND c.courseID = '{}'
                        AND questionNum = '{}'
            )
            '''.format(section_id, course_id, j))
            assessment_list.append(assessment_id[0].assessmentID)
        except:
            assessment_id = None
            assessment_list.append(assessment_id)

    for i in range(len(student_id)):

        registration_id = None
```

```
            try:

                registration_id = Registration_T.objects.raw('''
                    SELECT *
                    FROM spmapp_registration_t
                    WHERE student_id = '{}'
                        AND section_id = '{}'
                '''.format(student_id[i], section_id))
                registration_id = registration_id[0].registrationID
            except:
                registration_id = None

            if registration_id is None:
                print(section_id)
                print(student_id[i])
                registration = Registration_T(
                    student_id=student_id[i], section_id=section_id, reg_se
mester=semester)

                registration.save()
                registration_id = registration.registrationID

            for j in range(len(assessment_list)):
                evaluation = Evaluation_T(registration_id=registration_id,
assessment_id=assessment_list[j],
                                          obtainedMarks=obtainedMarks[i][j]
)

                evaluation.save()
        return redirect('evaluationdataentry')

 else:

        return render(request, 'evaluationdataentry.html', {
            'usertype': usertype,
            'clist': courselist,
            'semesters': semesters,
            'sections': section,

        })
```

B. OUTPUT FORM:

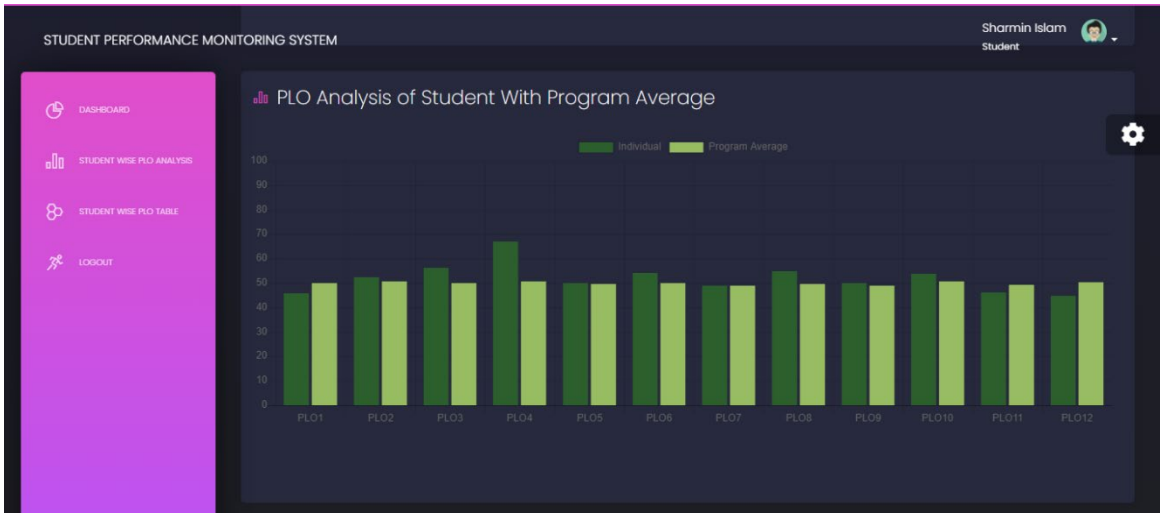



```
def getStudentWisePLO_program(studentID):
    with connection.cursor() as cursor:
        cursor.execute('''
                SELECT p.ploNum as plonum,100*(sum( e.obtainedMarks)/sum( a
.totalMarks)) as plopercent
                FROM spmapp_registration_t r,
                    spmapp_assessment_t a,
                    spmapp_evaluation_t e,
                    spmapp_co_t co,
                    spmapp_plo_t p
                WHERE  r.regID = e.reg_id
                    and e.assessment_id = a.assessmentID
                    and a.coID_id=co.coID
                    and co.plo_id = p.ploID
                    and  r.student_id = '{}'
                GROUP BY  p.ploID
                '''.format(studentID))
        row = cursor.fetchall()
    return row
```




```
def getStudentWiseePLO_course(studentID, course):
    with connection.cursor() as cursor:
        cursor.execute('''
                SELECT p.ploNum as plonum,100*(sum( e.obtainedMarks)/sum( a
.totalMarks)) as plopercent
                FROM spmapp_registration_t r,
                    spmapp_assessment_t a,
                    spmapp_evaluation_t e,
                    spmapp_co_t co,
                    spmapp_plo_t p,
                    spmapp_course_t c
                WHERE  r.regID = e.reg_id
                    and e.assessment_id = a.assessmentID
                    and a.coID_id = co.coID
                    and co.plo_id = p.ploID
                    and co.course_id = c.courseNum
                    and  r.student_id = '{}'
                    and c.courseID = '{}'
                GROUP BY  p.ploID

                '''.format(studentID, course))
        row = cursor.fetchall()
    return row

def getCourseWisePLO(course, uni):
    with connection.cursor() as cursor:
        cursor.execute('''
             SELECT derived.plonum, avg(per)
             FROM(
```

```
                SELECT p.ploID as PLOID,p.ploNum as ploNum, 100*sum(e.obtai
nedMarks)/sum(a.TotalMarks) as per
                FROM spmapp_registration_t r,
                    spmapp_evaluation_t e,
                    spmapp_section_t s,
                    spmapp_course_t cr,
                    spmapp_program_t prog,
                    spmapp_department_t d,
                    spmapp_school_t sch,
                    spmapp_university_t u,
                    spmapp_assessment_t a,
                    spmapp_co_t c,
                    spmapp_plo_t p
                WHERE r.section_id = s.sectionID
                    and e.reg_id = r.regID
                    and a.assessmentID = e.assessment_id
                    and a.coID_id = c.coID
                    and c.plo_id = p.ploID
                    and c.course_id = cr.courseNum
                    and cr.program_id = prog.programID
                    and prog.department_id = d.departmentNum
                    and d.school_id = sch.schoolNum
                    and sch.university_id = u.universityID
                    and cr.courseID = '{}'
                    and u.universityID = '{}'
                    GROUP BY p.ploNum,r.student_id) derived
            GROUP BY derived.ploNum

  '''.format(course, uni))

 row = cursor.fetchall()
        row.sort(key=len)

 return row
```




```
def getStudentWisePLO_program(studentID):
    with connection.cursor() as cursor:
        cursor.execute('''
                SELECT p.ploNum as plonum,100*(sum( e.obtainedMarks)/sum( a
.totalMarks)) as plopercent
                FROM spmapp_registration_t r,
                    spmapp_assessment_t a,
                    spmapp_evaluation_t e,
                    spmapp_co_t co,
                    spmapp_plo_t p
                WHERE  r.regID = e.reg_id
                    and e.assessment_id = a.assessmentID
                    and a.coID_id=co.coID
                    and co.plo_id = p.ploID
                    and  r.student_id = '{}'
                GROUP BY  p.ploID

                '''.format(studentID))
        row = cursor.fetchall()
    return row


def getProgramWisePLOpp(program):
    with connection.cursor() as cursor:
        cursor.execute('''
             SELECT derived.plonum, avg(per)
             FROM(
                SELECT p.ploID as PLOID, p.ploNum as ploNum, 100*sum(e.obta
inedMarks)/sum(a.TotalMarks) as per
                FROM spmapp_registration_t r,
                    spmapp_evaluation_t e,
```

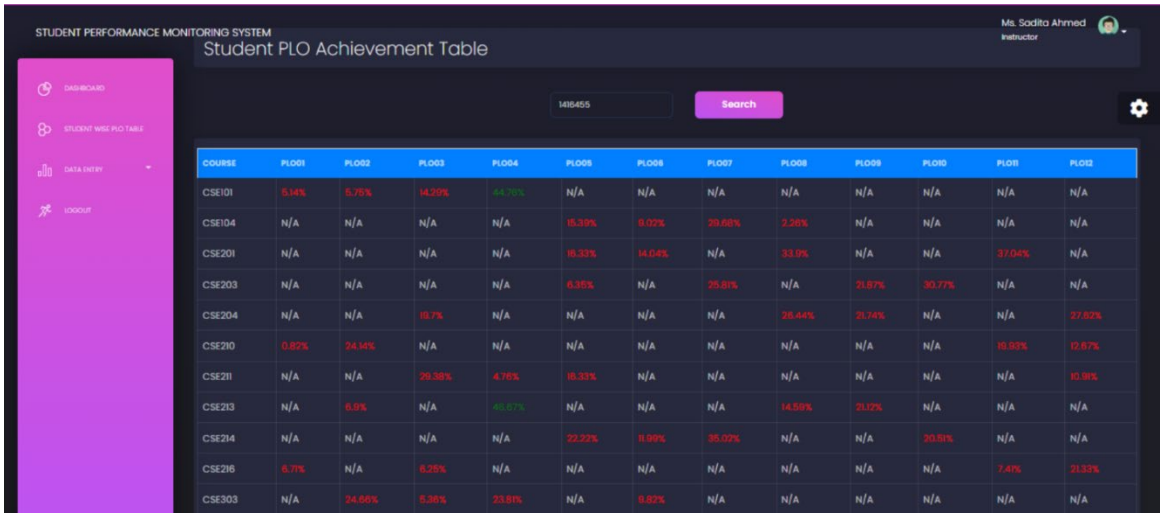
```
                    spmapp_program_t pr,
                    spmapp_assessment_t a,
                    spmapp_co_t c,
                    spmapp_plo_t p
                WHERE e.reg_id = r.regID
                    and a.assessmentID = e.assessment_id
                    and a.coID_id = c.coID
                    and c.plo_id = p.ploID
                    and pr.programID = p.program_id
                    and pr.programID = '{}'
                    GROUP BY p.ploID, r.student_id) derived
            GROUP BY derived.PLOID
                ''').format(program))
        row = cursor.fetchall()

    return row
```


```
def getCourseWiseStudentPLO(studentID, cat):
    with connection.cursor() as cursor:
        cursor.execute('''
   SELECT p.ploNum as ploNum,c.courseID, (sum(e.obtainedMarks)/sum(a.totalM
arks))*100, derived.Total
               FROM spmapp_registration_t r,
                   spmapp_assessment_t a,
                   spmapp_evaluation_t e,
                   spmapp_co_t co,
                   spmapp_plo_t p,
                   spmapp_course_t c,
                   (
```

```
                        SELECT p.ploNum as ploNum,sum(a.totalMarks) as Tota
l, r.student_id as StudentID
                        FROM spmapp_registration_t r,
                            spmapp_assessment_t a,
                            spmapp_evaluation_t e,
                            spmapp_co_t co,
                            spmapp_plo_t p
                        WHERE r.regID = e.reg_id
                            and e.assessment_id = a.assessmentID
                            and a.coID_id=co.coID
                            and co.plo_id = p.ploID
                            and r.student_id = '{}'
                        GROUP BY  r.student_id,p.ploID) derived
                WHERE r.student_id = derived.StudentID
                     and e.reg_id = r.regID
                     and e.assessment_id = a.assessmentID
                     and a.coID_id=co.coID
                     and co.plo_id = p.ploID
                     and p.ploNum = derived.ploNum
                     and c.courseNum = co.course_id

                GROUP BY  p.ploID,co.course_id

                '''.format(studentID))
        row = cursor.fetchall()

    table = []
    courses = []

    for entry in row:
        if entry[1] not in courses:
            courses.append(entry[1])
    courses.sort()
    plo = ["PLO1", "PLO2", "PLO3", "PLO4", "PLO5", "PLO6",
           "PLO7", "PLO8", "PLO9", "PLO10", "PLO11", "PLO12"]
    for i in courses:
        temptable = []
        if cat == 'report':
            temptable = [i]

        for j in plo:
            found = False
            for k in row:
```

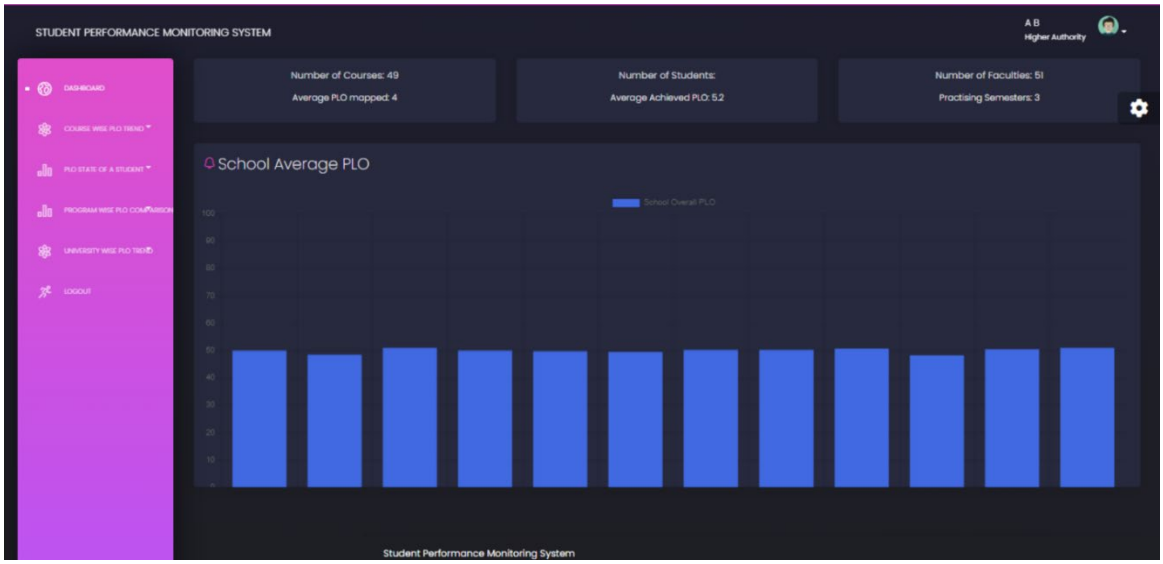
```
                if j == k[0] and i == k[1]:
                    if cat == 'report':
                        temptable.append(np.round(100 * k[2] / k[3], 2))
                    elif cat == 'chart':
                        temptable.append(np.round(100 * k[2] / k[4], 2))
                    found = True
            if not found:
                if cat == 'report':
                    temptable.append('N/A')
                elif cat == 'chart':
                    temptable.append(0)
        table.append(temptable)
    return plo, courses, table
```


```
def getProgramIDOfAUniversity(progName, uniID):
    with connection.cursor() as cursor:
        cursor.execute('''
            SELECT p.programID
            FROM spmapp_program_t p,
                spmapp_department_t d,
                spmapp_school_t s,
                spmapp_university_t u
            WHERE p.department_id = d.departmentNum
                and d.school_id = s.schoolNum
                and s.university_id = u.universityID
                and p.programName = '{}'
                and u.universityID = '{}'
        '''.format(progName, uniID))
        row = cursor.fetchall()
    return row
```




```
def getCourseWiseStudentPLO(studentID, cat):
    with connection.cursor() as cursor:
        cursor.execute('''
                SELECT p.ploNum as ploNum,c.courseID, (sum(e.obtainedMarks)/
sum(a.totalMarks))*100, derived.Total
                FROM spmapp_registration_t r,
                    spmapp_assessment_t a,
                    spmapp_evaluation_t e,
                    spmapp_co_t co,
                    spmapp_plo_t p,
                    spmapp_course_t c,
                    (
          SELECT p.ploNum as ploNum,sum(a.totalMarks) as Total, r.student_i
d as StudentID
                        FROM spmapp_registration_t r,
                            spmapp_assessment_t a,
                            spmapp_evaluation_t e,
                            spmapp_co_t co,
                            spmapp_plo_t p
                        WHERE r.regID = e.reg_id
                            and e.assessment_id = a.assessmentID
                            and a.coID_id=co.coID
                            and co.plo_id = p.ploID
                            and r.student_id = '{}'
                        GROUP BY  r.student_id,p.ploID) derived
                WHERE r.student_id = derived.StudentID
                     and e.reg_id = r.regID
                     and e.assessment_id = a.assessmentID
                     and a.coID_id=co.coID
                     and co.plo_id = p.ploID
```

```
                and p.ploNum = derived.ploNum
                and c.courseNum = co.course_id

            GROUP BY  p.ploID,co.course_id

            '''.format(studentID))
    row = cursor.fetchall()

table = []
courses = []

for entry in row:
    if entry[1] not in courses:
        courses.append(entry[1])
courses.sort()
plo = ["PLO1", "PLO2", "PLO3", "PLO4", "PLO5", "PLO6",
       "PLO7", "PLO8", "PLO9", "PLO10", "PLO11", "PLO12"]

for i in courses:
    temptable = []
    if cat == 'report':
        temptable = [i]

    for j in plo:
        found = False
        for k in row:
            if j == k[0] and i == k[1]:
                if cat == 'report':
                    temptable.append(np.round(100 * k[2] / k[3], 2))
                elif cat == 'chart':
                    temptable.append(np.round(100 * k[2] / k[4], 2))
                found = True
        if not found:
            if cat == 'report':
                temptable.append('N/A')
            elif cat == 'chart':
                temptable.append(0)
    table.append(temptable)
return plo, courses, table
```

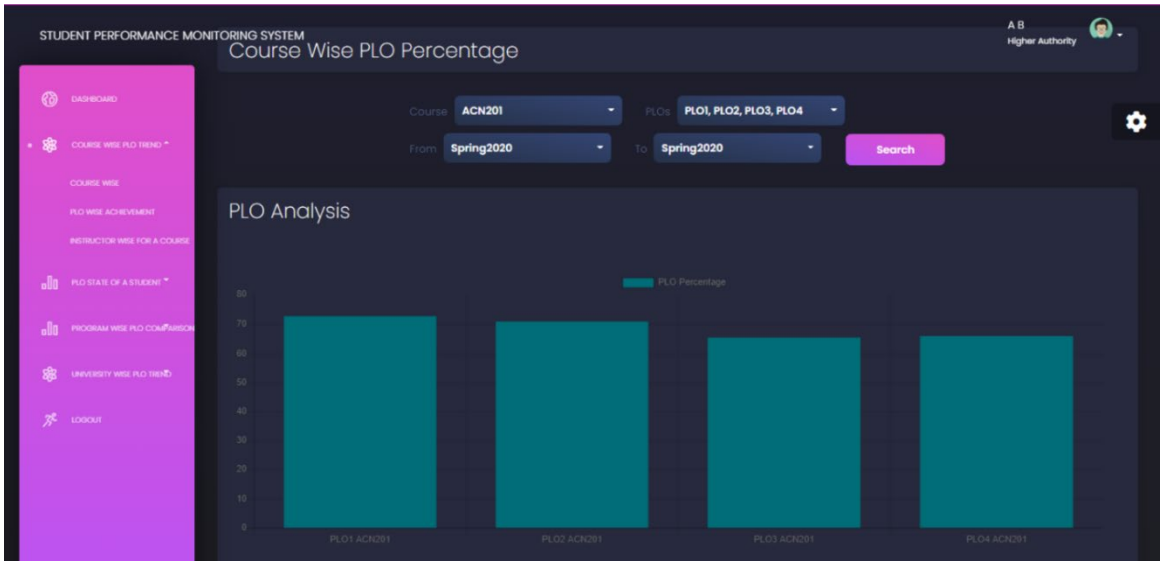



```
def getSchoolWiseplo(sch):
    with connection.cursor() as cursor:
        cursor.execute('''
             SELECT derived.plonum, avg(per)
             FROM(
                SELECT p.ploID as PLOID,p.ploNum as ploNum, 100*sum(e.obtai
nedMarks)/sum(a.TotalMarks) as per
                FROM spmapp_registration_t r,
                    spmapp_evaluation_t e,
                    spmapp_program_t pr,
                    spmapp_department_t d,
                    spmapp_school_t sch,
                    spmapp_assessment_t a,
                    spmapp_co_t c,
                    spmapp_plo_t p
                WHERE r.regID = e.reg_id
                    and pr.department_id = d.departmentNum
                    and a.assessmentID = e.assessment_id
                    and a.coID_id = c.coID
                    and c.plo_id = p.ploID
                    and p.program_id = pr.programID
                    and d.school_id = sch.schoolNum
                    and sch.schoolNum = '{}'
                    GROUP BY p.ploNum,r.student_id) derived
             GROUP BY derived.ploNum
                  '''.format(sch))
        row = cursor.fetchall()
    plo = []
    avg = []
```

```
    for r in row:
        plo.append(r[0])
        avg.append(r[1])

    return plo, avg
```


```
def getCourseWisePLOComp(course, semester):
    cursor = connection.cursor()
    cursor.execute('''
        SELECT ploNum,COUNT(*)
        FROM(
            SELECT p.ploNum as ploNum, c.course_id, r.student_id, 100*(sum(
e.obtainedMarks)/sum(a.totalMarks))
            FROM spmapp_registration_t r,
                spmapp_evaluation_t e,
                spmapp_assessment_t a,
                spmapp_co_t c,
                spmapp_plo_t p,
                spmapp_course_t cr
            WHERE e.reg_id = r.regID
                and a.assessmentID = e.assessment_id
                and a.coID_id = c.coID
                and c.plo_id = p.ploID
                and c.course_id = cr.courseNum
                and cr.courseNum = '{}'
                and r.reg_semester = '{}'
            GROUP BY p.ploNum, c.course_id, r.student_id) derived
        GROUP BY  derived.ploNum
```

```
    '''.format(course, semester))

    row1 = cursor.fetchall()
    row1.sort(key=lambda t: len(t[0]))

    cursor.execute('''
  SELECT ploNum,COUNT(*)
            FROM(
                SELECT p.ploNum as ploNum, c.course_id, r.student_id, 100*(
sum(e.obtainedMarks)/sum(a.totalMarks))
                FROM spmapp_registration_t r,
                    spmapp_course_t cr,
                    spmapp_evaluation_t e,
                    spmapp_assessment_t a,
                    spmapp_co_t c,
                    spmapp_plo_t p
                WHERE e.reg_id = r.regID
                    and a.assessmentID = e.assessment_id
                    and a.coID_id = c.coID
                    and c.plo_id = p.ploID
                    and c.course_id = cr.courseNum
                    and cr.courseNum = '{}'
                    and r.reg_semester = '{}'
                GROUP BY p.ploNum, c.course_id, r.student_id
                HAVING  100*(sum(e.obtainedMarks)/sum(a.totalMarks))>=40) d
erived
            GROUP BY  derived.ploNum

        '''.format(course, semester))
    row2 = cursor.fetchall()
    row2.sort(key=lambda t: len(t[0]))
    plo = []
    expected = []
    actual = []

    for r in row1:
        plo.append(r[0])
        expected.append(r[1])

    for r in row2:
        actual.append(r[1])

    return plo, expected, actual
```



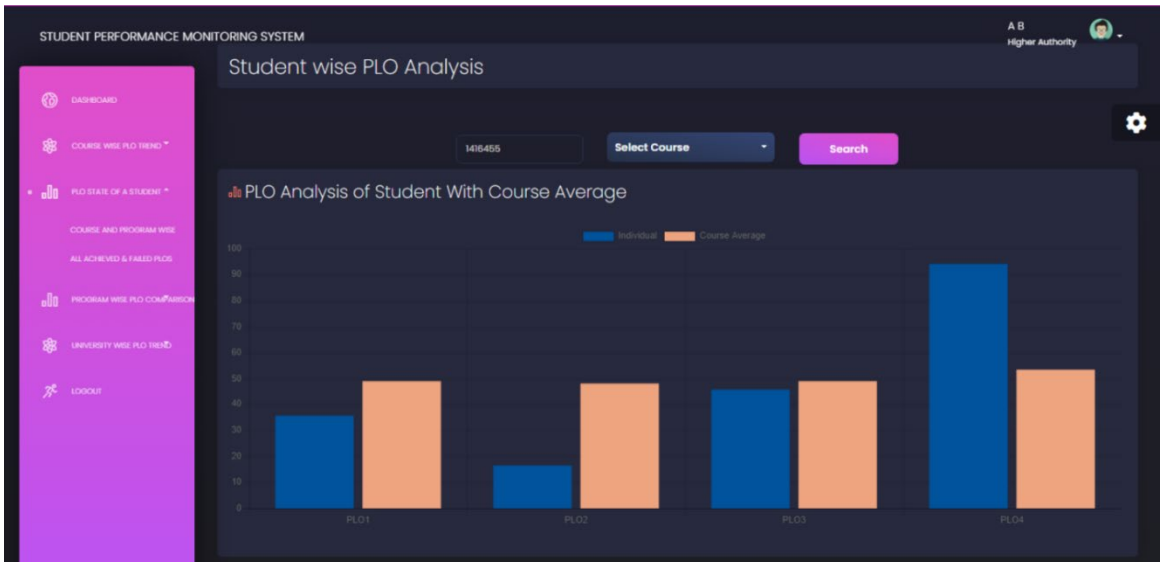

```
def getCourseWisePLOC(course, semester, plo):
    cursor = connection.cursor()

    cursor.execute('''
        SELECT  ploNum, COUNT(*)
        FROM(
            SELECT p.ploNum as ploNum,100*sum(e.obtainedMarks)/sum(a.totalM
arks) as marks
            FROM spmapp_registration_t r,
                spmapp_evaluation_t e,
                spmapp_assessment_t a,
                spmapp_co_t c,
                spmapp_course_t cr,
                spmapp_plo_t p
            WHERE r.regID = e.reg_id
                and e.assessment_id = a.assessmentID
                and a.coID_id = c.coID
                and c.plo_id = p.ploID
                and c.course_id = cr.courseNum
                and cr.courseID = '{}'
                and r.reg_semester ='{}'
                and p.ploNum = '{}'
            GROUP BY p.ploNum,r.student_id) derived1
        GROUP BY ploNum

    '''.format(course, semester, plo))

    temp1 = cursor.fetchall()
    temp1.sort(key=lambda t: len(t[0]))
```

```
   cursor.execute('''
            SELECT ploNum, COUNT(*)
            FROM(
                SELECT p.ploNum as ploNum,100*sum(e.obtainedMarks)/sum(a.tot
alMarks) as marks
                FROM spmapp_registration_t r,
                 spmapp_evaluation_t e,
                 spmapp_assessment_t a,
                 spmapp_co_t c,
                 spmapp_course_t cr,
                 spmapp_plo_t p
                WHERE r.regID = e.reg_id
                 and e.assessment_id = a.assessmentID
                 and a.coID_id = c.coID
                 and c.plo_id = p.ploID
                 and c.course_id = cr.courseNum
                 and cr.courseID = '{}'
                 and r.reg_semester ='{}'
                 and p.ploNum = '{}'
                GROUP BY p.ploNum,r.student_id
                HAVING 100*sum(e.obtainedMarks)/sum(a.totalMarks)>=40) deriv
ed1
            GROUP BY ploNum
        '''.format(course, semester, plo))

    temp2 = cursor.fetchall()
    temp2.sort(key=lambda t: len(t[0]))
    tt = []
    tt2 = []
    for t in temp1:
        tt.append(t[1])
    for t in temp2:
        tt2.append(t[1])

    return tt2, tt
```


```
def getInstructorWisePLOForCourse(course, semester, uni):
    cursor = connection.cursor()
    cursor.execute('''
        SELECT p.ploNum, COUNT(*)
        FROM
            spmapp_program_t pr,
            spmapp_course_t cr,
            spmapp_section_t sec,
            spmapp_registration_t r,
            spmapp_plo_t p,
            spmapp_co_t c,
            spmapp_department_t d,
            spmapp_school_t sch,
            spmapp_university_t u
        WHERE c.plo_id = p.ploID
            and c.course_id = cr.courseNum
            and cr.courseNum = sec.course_id
            and sec.sectionID = r.section_id
            and p.program_id = pr.programID
            and pr.department_id = d.departmentNum
            and d.school_id = sch.schoolNum
            and sch.university_id = u.universityID
            and cr.courseID = '{}'
            and sec.sec_semester = '{}'
            and u.universityID = '{}'
        GROUP BY p.ploNum

    '''.format(course, semester, uni))

    row1 = cursor.fetchall()
```

```
   row1.sort(key=lambda t: len(t[0]))
    cursor.execute('''
            SELECT derived.ins, derived.ploNum, COUNT(*)
            FROM(
                SELECT p.ploNum as ploNum, i.name as ins, 100*(sum(e.obtain
edMarks)/sum(a.totalMarks))
                FROM spmapp_registration_t r,
                spmapp_program_t pr,
                spmapp_evaluation_t e,
                spmapp_assessment_t a,
                spmapp_co_t c,
                spmapp_instructor_t i,
                spmapp_plo_t p,
                spmapp_course_t cr,
                spmapp_section_t sec,
                spmapp_department_t d,
                spmapp_school_t sch,
                spmapp_university_t u
            WHERE e.reg_id = r.regID
                and a.assessmentID = e.assessment_id
                and c.course_id = cr.courseNum
                and cr.courseNum = sec.course_id
                and sec.instructor_id = i.instructorID
                and sec.sectionID = r.section_id
                and a.coID_id = c.coID
                and c.plo_id = p.ploID
                and p.program_id = pr.programID
                and pr.department_id = d.departmentNum
                and d.school_id = sch.schoolNum
                and sch.university_id = u.universityID
                and cr.courseID = '{}'
                and sec.sec_semester = '{}'
                and u.universityID = '{}'
                GROUP BY  i.instructorID, p.ploNum,r.student_id
                HAVING  100*(sum(e.obtainedMarks)/sum(a.totalMarks))>=40) d
erived
            GROUP BY  derived.ins, derived.ploNum
            ORDER BY derived.ins, derived.ploNum
        '''.format(course, semester, uni))
    row2 = cursor.fetchall()
    row2.sort(key=lambda t: len(t[1]))
    plo = []
    fac = []
```

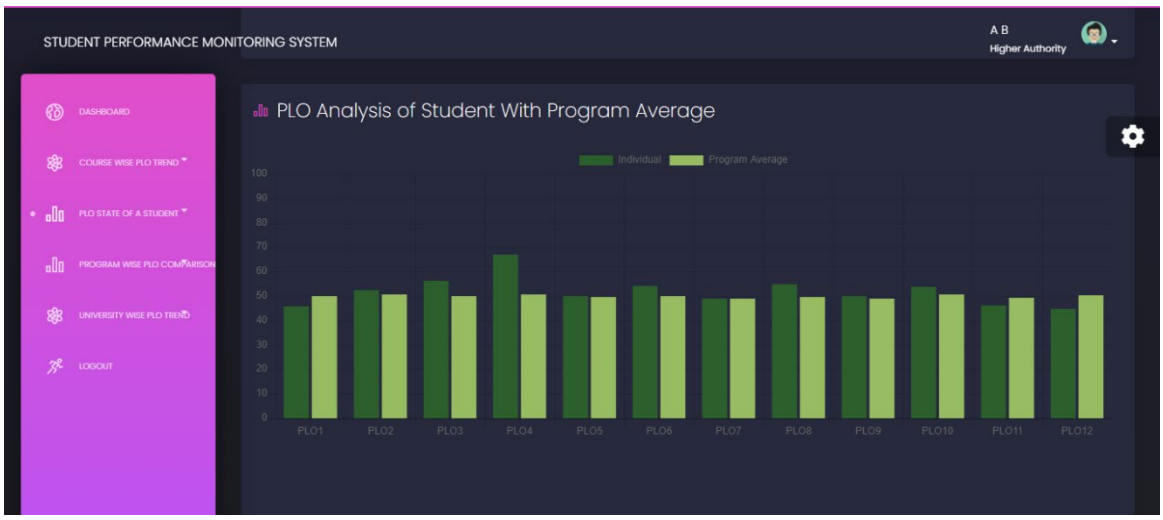
```
    table = []

    for i in row1:
        total = i[1]

    for r in row2:
        if r[0] not in fac:
            fac.append(r[0])
        if r[1] not in plo:
            plo.append(r[1])

    for f in fac:
        p = []
        for r in row2:
            if r[0] == f:
                p.append((r[2]/total)*100)
        table.append(p)

    return plo, fac, table
```




```
def getCourseListOfAStudent(student):
    with connection.cursor() as cursor:
        cursor.execute('''
   SELECT c.courseID
            FROM spmapp_course_t c,
                spmapp_section_t s,
                spmapp_registration_t r
            WHERE c.courseNum = s.course_id
                and s.sectionID = r.section_id
```

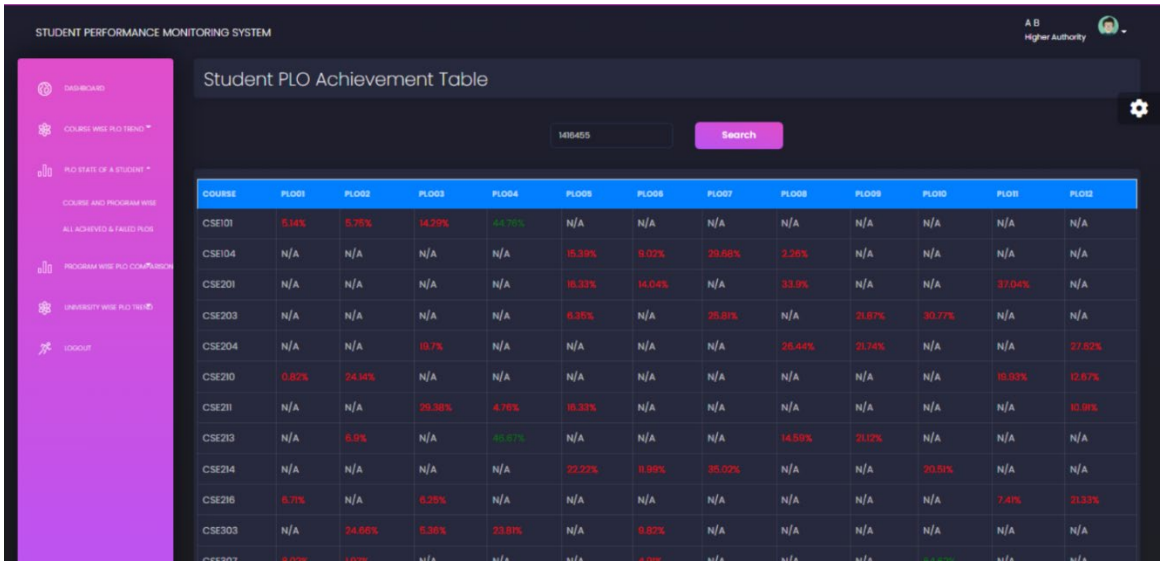
```
                and r.student_id = '{}'
        '''.format(student))
        row = cursor.fetchall()
    return row

def getStudentWiseePLO_course(studentID, course):
    with connection.cursor() as cursor:
        cursor.execute('''
                SELECT p.ploNum as plonum,100*(sum( e.obtainedMarks)/sum( a
.totalMarks)) as plopercent
                FROM spmapp_registration_t r,
                    spmapp_assessment_t a,
                    spmapp_evaluation_t e,
                    spmapp_co_t co,
                    spmapp_plo_t p,
                    spmapp_course_t c
                WHERE  r.regID = e.reg_id
                    and e.assessment_id = a.assessmentID
                    and a.coID_id = co.coID
                    and co.plo_id = p.ploID
                    and co.course_id = c.courseNum
                    and  r.student_id = '{}'
                    and c.courseID = '{}'
                GROUP BY  p.ploID

                '''.format(studentID, course))
        row = cursor.fetchall()
    return row

def getCourseWisePLO(course, uni):
    with connection.cursor() as cursor:
        cursor.execute('''
             SELECT derived.plonum, avg(per)
             FROM(
                SELECT p.ploID as PLOID,p.ploNum as ploNum, 100*sum(e.obtai
nedMarks)/sum(a.TotalMarks) as per
                FROM spmapp_registration_t r,
                    spmapp_evaluation_t e,
                    spmapp_section_t s,
                    spmapp_course_t cr,
                    spmapp_program_t prog,
                    spmapp_department_t d,
                    spmapp_school_t sch,
```

```
                    spmapp_university_t u,
                    spmapp_assessment_t a,
                    spmapp_co_t c,
                    spmapp_plo_t p
                WHERE r.section_id = s.sectionID
                    and e.reg_id = r.regID
                    and a.assessmentID = e.assessment_id
                    and a.coID_id = c.coID
                    and c.plo_id = p.ploID
                    and c.course_id = cr.courseNum
                    and cr.program_id = prog.programID
                    and prog.department_id = d.departmentNum
                    and d.school_id = sch.schoolNum
                    and sch.university_id = u.universityID
                    and cr.courseID = '{}'
                    and u.universityID = '{}'
                    GROUP BY p.ploNum,r.student_id) derived
            GROUP BY derived.ploNum
                ''').format(course, uni))
        row = cursor.fetchall()
        row.sort(key=len)
    return row
```




```
def getStudentWisePLO_program(studentID):
    with connection.cursor() as cursor:
        cursor.execute('''
                SELECT p.ploNum as plonum,100*(sum( e.obtainedMarks)/sum( a
.totalMarks)) as plopercent
                FROM spmapp_registration_t r,
                    spmapp_assessment_t a,
```

```
                    spmapp_evaluation_t e,
                    spmapp_co_t co,
                    spmapp_plo_t p
                WHERE  r.regID = e.reg_id
                    and e.assessment_id = a.assessmentID
                    and a.coID_id=co.coID
                    and co.plo_id = p.ploID
                    and  r.student_id = '{}'
                GROUP BY  p.ploID

                '''.format(studentID))
        row = cursor.fetchall()
    return row


def getProgramWisePLOpp(program):
    with connection.cursor() as cursor:
        cursor.execute('''
             SELECT derived.plonum, avg(per)
             FROM(
                SELECT p.ploID as PLOID, p.ploNum as ploNum, 100*sum(e.obta
inedMarks)/sum(a.TotalMarks) as per
                FROM spmapp_registration_t r,
                    spmapp_evaluation_t e,
                    spmapp_program_t pr,
                    spmapp_assessment_t a,
                    spmapp_co_t c,
                    spmapp_plo_t p
                WHERE e.reg_id = r.regID
                    and a.assessmentID = e.assessment_id
                    and a.coID_id = c.coID
                    and c.plo_id = p.ploID
                    and pr.programID = p.program_id
                    and pr.programID = '{}'
                    GROUP BY p.ploID, r.student_id) derived
             GROUP BY derived.PLOID
                   '''.format(program))
        row = cursor.fetchall()

    return row
```

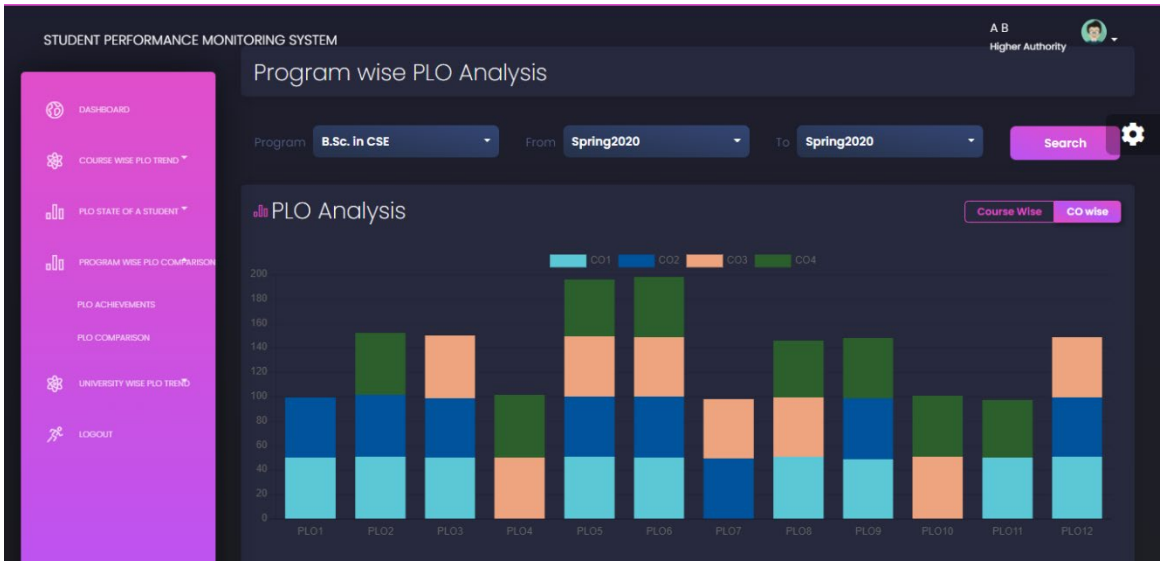



```
def getCourseWiseStudentPLO(studentID, cat):
    with connection.cursor() as cursor:
        cursor.execute('''
                SELECT p.ploNum as ploNum,c.courseID, (sum(e.obtainedMarks)/
sum(a.totalMarks))*100, derived.Total
                FROM spmapp_registration_t r,
                    spmapp_assessment_t a,
                    spmapp_evaluation_t e,
                    spmapp_co_t co,
                    spmapp_plo_t p,
                    spmapp_course_t c,
                    (
                        SELECT p.ploNum as ploNum,sum(a.totalMarks) as Tota
l, r.student_id as StudentID
                        FROM spmapp_registration_t r,
                            spmapp_assessment_t a,
                            spmapp_evaluation_t e,
                            spmapp_co_t co,
                            spmapp_plo_t p
                        WHERE r.regID = e.reg_id
                            and e.assessment_id = a.assessmentID
                            and a.coID_id=co.coID
                            and co.plo_id = p.ploID
                            and r.student_id = '{}'
                        GROUP BY  r.student_id,p.ploID) derived
                WHERE r.student_id = derived.StudentID
                     and e.reg_id = r.regID
                     and e.assessment_id = a.assessmentID
                     and a.coID_id=co.coID
```

```
                and co.plo_id = p.ploID
                and p.ploNum = derived.ploNum
                and c.courseNum = co.course_id

            GROUP BY  p.ploID,co.course_id

            '''.format(studentID))
    row = cursor.fetchall()

table = []
courses = []

for entry in row:
    if entry[1] not in courses:
        courses.append(entry[1])
courses.sort()
plo = ["PLO1", "PLO2", "PLO3", "PLO4", "PLO5", "PLO6",
       "PLO7", "PLO8", "PLO9", "PLO10", "PLO11", "PLO12"]

for i in courses:
    temptable = []
    if cat == 'report':
        temptable = [i]
    for j in plo:
        found = False
        for k in row:
            if j == k[0] and i == k[1]:
                if cat == 'report':
                    temptable.append(np.round(100 * k[2] / k[3], 2))
                elif cat == 'chart':
                    temptable.append(np.round(100 * k[2] / k[4], 2))
                found = True
        if not found:
            if cat == 'report':
                temptable.append('N/A')
            elif cat == 'chart':
                temptable.append(0)
    table.append(temptable)
return plo, courses, table
```


```
def getProgramWisePLO(program, cat):
    with connection.cursor() as cursor:
        cursor.execute('''
                SELECT p.ploNum as ploNum,c.courseID, (sum(e.obtainedMarks)/
sum(a.totalMarks))*100, derived.Total
                FROM spmapp_registration_t r,
                    spmapp_assessment_t a,
                    spmapp_evaluation_t e,
    spmapp_co_t co,
                    spmapp_plo_t p,
                    spmapp_course_t c,
                    (
                        SELECT p.ploNum as ploNum,sum(a.totalMarks) as Tota
l, r.student_id as StudentID
                        FROM spmapp_registration_t r,
                            spmapp_assessment_t a,
                            spmapp_evaluation_t e,
                            spmapp_co_t co,
                            spmapp_plo_t p,
                            spmapp_program_t pr
                        WHERE r.regID = e.reg_id
                            and e.assessment_id = a.assessmentID
                            and a.coID_id=co.coID
                            and co.plo_id = p.ploID
                            and p.program_id =  pr.programID
                            and pr.programID = '{}'
                        GROUP BY  r.student_id,p.ploID) derived
                WHERE r.student_id = derived.StudentID
                     and e.reg_id = r.regID
                     and e.assessment_id = a.assessmentID
```

```
                and a.coID_id=co.coID
                and co.plo_id = p.ploID
                and p.ploNum = derived.ploNum
                and c.courseNum = co.course_id
            GROUP BY  p.ploID,co.course_id

            '''.format(program))
    row = cursor.fetchall()

table = []
courses = []

for entry in row:
    if entry[1] not in courses:
        courses.append(entry[1])
courses.sort()

plo = ["PLO1", "PLO2", "PLO3", "PLO4", "PLO5", "PLO6",
       "PLO7", "PLO8", "PLO9", "PLO10", "PLO11", "PLO12"]

for i in courses:
    temptable = []
    # if cat == 'report':
    #     temptable = [i]

    for j in plo:
        found = False
        for k in row:
            if j == k[0] and i == k[1]:
                # if cat == 'report':
                #     temptable.append(np.round(100 * k[2] / k[3], 2))
                temptable.append(np.round(100 * k[2] / k[3], 2))
                found = True
        if not found:
            # if cat == 'report':
            #     temptable.append('N/A')
            temptable.append(0)
    table.append(temptable)

return plo, courses, table
```




```
def getCOWiseProgramPLO(program, cat):
    with connection.cursor() as cursor:
        cursor.execute('''
                SELECT p.ploNum as ploNum,co.coNum, sum(e.obtainedMarks),sum
(a.totalMarks),derived.Total
                FROM spmapp_registration_t r,
                    spmapp_assessment_t a,
    spmapp_program_t pr,
                    spmapp_evaluation_t e,
                    spmapp_co_t co,
                    spmapp_plo_t p,
                    (
                        SELECT p.ploNum as ploNum,sum(a.totalMarks) as Tota
l, pr.programID as prog
                        FROM spmapp_registration_t r,
                            spmapp_assessment_t a,
                            spmapp_evaluation_t e,
                            spmapp_program_t pr,
                            spmapp_co_t co,
                            spmapp_plo_t p
                        WHERE r.regID = e.reg_id
                            and e.assessment_id = a.assessmentID
                            and a.coID_id=co.coID
                            and co.plo_id = p.ploID
                            and p.program_id = pr.programID
                            and pr.programID = '{}'
                        GROUP BY  r.student_id,p.ploID) derived
                WHERE pr.programID = derived.prog
                     and e.reg_id = r.regID
```

```
                and e.assessment_id = a.assessmentID
                and a.coID_id=co.coID
                and co.plo_id = p.ploID
                and p.ploNum = derived.ploNum
                and p.program_id = pr.programID

            GROUP BY  p.ploID,co.coNum;

            '''.format(program))
    row = cursor.fetchall()

table = []
cos = []

for entry in row:
    if entry[1] not in cos:
        cos.append(entry[1])
cos.sort()
plo = ["PLO1", "PLO2", "PLO3", "PLO4", "PLO5", "PLO6",
       "PLO7", "PLO8", "PLO9", "PLO10", "PLO11", "PLO12"]

for i in cos:
    temptable = []
    if cat == 'report':
        temptable = [i]

    for j in plo:
        found = False
        for k in row:
            if j == k[0] and i == k[1]:
                if cat == 'report':
                    temptable.append(np.round(100 * k[2] / k[3], 2))
                elif cat == 'chart':
                    temptable.append(np.round(100 * k[2] / k[3], 2))
                found = True
        if not found:
            if cat == 'report':
                temptable.append('N/A')
            elif cat == 'chart':
                temptable.append(0)
    table.append(temptable)
return plo, cos, table
```


```
def getProgramWisePLOComp(program, semester):
    cursor = connection.cursor()
    cursor.execute('''
        SELECT ploNum,COUNT(*)
            SELECT p.ploNum as ploNum, c.course_id, r.student_id, 100*(sum(
e.obtainedMarks)/sum(a.totalMarks))
            FROM spmapp_registration_t r,
                spmapp_program_t pr,
                spmapp_evaluation_t e,
                spmapp_assessment_t a,
                spmapp_co_t c,
                spmapp_plo_t p
            WHERE e.reg_id = r.regID
                and a.assessmentID = e.assessment_id
                and a.coID_id = c.coID
                and c.plo_id = p.ploID
                and p.program_id = pr.programID
                and pr.programID = '{}'
                and r.reg_semester = '{}'
            GROUP BY p.ploNum, c.course_id, r.student_id) derived
        GROUP BY  derived.ploNum

    '''.format(program, semester))

    row1 = cursor.fetchall()
    row1.sort(key=lambda t: len(t[0]))

    cursor.execute('''
            SELECT ploNum,COUNT(*)
```

```
            FROM(
                SELECT p.ploNum as ploNum, c.course_id, r.student_id, 100*(
sum(e.obtainedMarks)/sum(a.totalMarks))
                FROM spmapp_registration_t r,
                    spmapp_program_t pr,
                    spmapp_evaluation_t e,
                    spmapp_assessment_t a,
                    spmapp_co_t c,
                    spmapp_plo_t p
                WHERE e.reg_id = r.regID
                    and a.assessmentID = e.assessment_id
                    and a.coID_id = c.coID
                    and c.plo_id = p.ploID
                    and p.program_id = pr.programID
                    and pr.programID = '{}'
                    and r.reg_semester = '{}'
                GROUP BY p.ploNum, c.course_id, r.student_id
                HAVING  100*(sum(e.obtainedMarks)/sum(a.totalMarks))>=40) d
erived
            GROUP BY  derived.ploNum
        '''.format(program, semester))

    row2 = cursor.fetchall()
    row2.sort(key=lambda t: len(t[0]))

    plo = []
    expected = []
    actual = []

    for r in row1:
        plo.append(r[0])
        expected.append(r[1])

    for r in row2:
        actual.append(r[1])

    return plo, expected, actual
```


```
def getUniversityWiseCountStudent_program(semester, program, uni):

    with connection.cursor() as cursor:
        cursor.execute('''
            SELECT ploNum, COUNT(*)
SELECT p.ploNum as ploNum, c.course_id, r.student_id, 100*(sum(e.obtainedMa
rks)/sum(a.totalMarks))
                FROM spmapp_registration_t r,
                    spmapp_program_t pr,
                    spmapp_evaluation_t e,
                    spmapp_assessment_t a,
                    spmapp_co_t c,
                    spmapp_plo_t p,
                    spmapp_department_t d,
                    spmapp_school_t sch,
                    spmapp_university_t u
                WHERE e.reg_id=r.regID
                    and a.assessmentID=e.assessment_id
                    and a.coID_id=c.coID
                    and c.plo_id=p.ploID
                    and p.program_id=pr.programID
                    and pr.department_id=d.departmentNum
                    and d.school_id=sch.schoolNum
                    and sch.university_id=u.universityID
                    and pr.programName='{}'
                    and r.reg_semester='{}'
                    and u.universityID='{}'
                GROUP BY p.ploNum, c.course_id, r.student_id
```

```
                HAVING  100*(sum(e.obtainedMarks)/sum(a.totalMarks)) >= 40)
 derived
            GROUP BY  derived.ploNum

            '''.format(program, semester, uni))

        row1 = cursor.fetchall()

    row1.sort(key=lambda t: len(t[0]))

    plo = []
    actual = []
    total = []
    for r in row1:
        total.append(r[1])

    for r in row1:
        plo.append(r[0])
        actual.append(r[1])

    return plo, actual
```


```
def getUniversityWiseGraduateStudent(program, uni):
    with connection.cursor() as cursor:
        cursor.execute('''
            SELECT ploNum, COUNT(*)
            FROM(
                SELECT p.ploNum as ploNum, c.course_id, r.student_id, 100*(
sum(e.obtainedMarks)/sum(a.totalMarks))
```

```
                FROM spmapp_registration_t r,
                    spmapp_program_t pr,
                    spmapp_evaluation_t e,
                    spmapp_assessment_t a,
                    spmapp_co_t c,
                    spmapp_plo_t p,
                    spmapp_department_t d,
                    spmapp_school_t sch,
                    spmapp_university_t u,
                    spmapp_student_t st
                WHERE e.reg_id=r.regID
                    and a.assessmentID=e.assessment_id
                    and a.coID_id=c.coID
                    and c.plo_id=p.ploID
                    and p.program_id=pr.programID
                    and pr.department_id=d.departmentNum
                    and d.school_id=sch.schoolNum
                    and sch.university_id=u.universityID
                    and pr.programID='{}'
                    and u.universityID='{}'
                    and st.graduateDate IS NOT NULL
                    and st.studentID = r.student_id
                GROUP BY p.ploNum, c.course_id, r.student_id
                HAVING  100*(sum(e.obtainedMarks)/sum(a.totalMarks)) >= 40)
derived
           GROUP BY  derived.ploNum

           '''.format(program, uni))

        row1 = cursor.fetchall()

    row1.sort(key=lambda t: len(t[0]))
    plo = []
    actual = []
    total = []
    for r in row1:
        total.append(r[1])

    for r in row1:
        plo.append(r[0])
        actual.append(r[1])

    return plo, actual
```


```
def getUniversityWisePloPerformance(p, uni):
    cursor = connection.cursor()
    cursor.execute('''
        SELECT ploNum,COUNT(*)
        FROM(
            SELECT p.ploNum as ploNum, c.course_id, r.student_id, 100*(sum(
e.obtainedMarks)/sum(a.totalMarks))
            FROM spmapp_registration_t r,
                spmapp_program_t pr,
                spmapp_evaluation_t e,
                spmapp_assessment_t a,
                spmapp_co_t c,
                spmapp_plo_t p,
                spmapp_department_t d,
                spmapp_school_t sch,
                spmapp_university_t u
            WHERE e.reg_id = r.regID
                and a.assessmentID = e.assessment_id
                and a.coID_id = c.coID
                and c.plo_id = p.ploID
                and p.program_id = pr.programID
                and pr.department_id = d.departmentNum
                and d.school_id = sch.schoolNum
                and sch.university_id = u.universityID
                and p.ploNum = '{}'
                and u.universityID = '{}'
            GROUP BY p.ploNum, c.course_id, r.student_id) derived
        GROUP BY  derived.ploNum
```

```
    '''.format(p, uni))
    row1 = cursor.fetchall()
    row1.sort(key=lambda t: len(t[0]))
    cursor.execute('''
            SELECT ploNum,COUNT(*)
            FROM(
                SELECT p.ploNum as ploNum, c.course_id, r.student_id, 100*(
sum(e.obtainedMarks)/sum(a.totalMarks))
                FROM spmapp_registration_t r,
                spmapp_program_t pr,
                spmapp_evaluation_t e,
                spmapp_assessment_t a,
                spmapp_co_t c,
                spmapp_plo_t p,
  spmapp_department_t d,
                spmapp_school_t sch,
                spmapp_university_t u
            WHERE e.reg_id = r.regID
                and a.assessmentID = e.assessment_id
                and a.coID_id = c.coID
                and c.plo_id = p.ploID
                and p.program_id = pr.programID
                and pr.department_id = d.departmentNum
                and d.school_id = sch.schoolNum
                and sch.university_id = u.universityID
                and p.ploNum = '{}'
                and u.universityID = '{}'
                GROUP BY p.ploNum, c.course_id, r.student_id
                HAVING  100*(sum(e.obtainedMarks)/sum(a.totalMarks))>=40) d
erived
            GROUP BY  derived.ploNum
        '''.format(p, uni))
    row2 = cursor.fetchall()
    row2.sort(key=lambda t: len(t[0]))
    plo = []
    expected = []
    actual = []
    for r in row1:
        plo.append(r[0])
        expected.append(r[1])
    for r in row2:
        actual.append(r[1])
    return plo, expected, actual
```

CHAPTER 5 – CONCLUSION:

A. PROBLEM AND SOLUTION:

Analysis Phase

Building upon project SPM developers, most of the work assumptions and queries were made while working on the rich picture and six element analyses of operations of the organization as there was no discreet data present. For a better understanding scenario and to overcome such confusions, respected faculty members and stake holder interviews were made.

Designing Phase

Based on descriptive research, entities were maintained at significant levels, which was reflected in the Relational Schema schematic. In addition, instructor feedback was extremely valuable and crucial in this part. As a result of the feedback session, we were able to easily develop the design phase, as we gained a clearer understanding of both back-end and front-end development.

The most time was spent on backend language. It was the first time any of us had used a backend server to get data from the database. While many tutorials are available online, implementing them on our project was too difficult, especially when it involved data manipulation, fetching specific data, and placing it on specific areas of the screen.

Implementation Phase

All the Software System Requirements (SSR's) could not satisfy the requirement successfully.

Front-End Developing tools: HTML, CSS, Bootstrap JavaScript, Chart Js

Back End Developing tools: Python, Django

Database-integration: MySQL

At the middle phase of developing the project the results from list queries do not show views from the SQL database. Most of the time was spent on it. Our team was eventually able to resolve the issues from Stack overflow. Mainly displaying the chart was a challenging phase for us, since dynamic and distinct features were present. Charts were an interesting and challenging part of this project.

B. ADDITIONAL FEATURE AND FUTURE DEVELOPMENT:

Additional feature and future Development (This section deals with the additional nonfunctional requirement of the system i.e., the extra features of the software should be provided in this section)

Future Developing Purposes:

- Plans for the project is, to add another feature which can predict A candidate's grade based on his/her past grades and performances.
- Securing database from sql injections
- Feature of sending emails to designation authority through admin
- Save feature as a pdf document of files such as Student Marks, Reports, Overall class progress etc.
- Deployment.

REFERENCES –

[1] Independent University - Bangladesh, "Curriculum for B Sc. in Computer Science and Engineering (Version 2.2)," Independent University - Bangladesh, March 19, 2017.
[2] Independent University - Bangladesh, "www.iub.edu.bd," 2020. [Online]. Available: www.iub.edu.bd.
[3] I. Department of Computer Science and Engineering, "Department of Computer Science and Engineering, IUB," 2020. [Online]. Available: http://www.cse.iub.edu.bd/.
[4] INSTITUTION OF ENGINEERS, BANGLADESH, Accreditation Manual for Undergraduate Engineering Programs, 2019. [Online].
Available: https://baetebangladesh.org/2nd_edi_05.03.2019_F.pdf
[5] D. o. C. S. &. S. Engineering, "The University of Western Australia," 2006. [Online].
Available: https://teaching.csse.uwa.edu.au/units/CITS3240/Lectures/db-er1-nup4.pdf.
[6]K.Stevens,"BetterEvaluation,"2020.[Online].Available:
https://www.betterevaluation.org/en/evaluation-options/richpictures.
[7]Wikipedia,"Wikipedia,"2020.[Online].Available:
https://en.wikipedia.org/wiki/Rich_picture.
[8] Wikipedia, 2020. [Online].
Available: https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation.
[9] An Improved Database System for Program Assessment, 2011. [Online].
Available: https://files.eric.ed.gov/fulltext/EJ1136803.pdf