Student Performance Monitor

Muhammad Fahimul Huq ■ Shoban Bhowmik

Airin Sikder Onamika ■ Kazi Tokir Ahmed Shawon

Md Takbir

CSE303

Database Management System

Group 02

Muhammad Fahimul Huq	1610044
Shoban Bhowmik	1930533
Airin Sikder Onamika	1822098
Kazi Tokir Ahmed Shawon	1821890
Md Takbir	1822068

Contents

CHAPTER 1

INTRODUCTION

- BACKGROUND OF THE PROJECT
- OBJECTIVE OF THE PROJECT
- SCOPE OF THE PROJECT

BACKGROUND OF THE PROJECT

The academic process of higher education must achieve something that improves the learning process. To make this happen, universities must monitor and evaluate the results of the teaching process by looking at student performance. We are going to design, create and deliver software that will help universities promote a more productive and effective way to assess students everywhere and that is the goal of our project. Performance monitoring includes assessments that play an important role in providing students, teachers, administrators, and policymakers with the information they need to make decisions. If we talk about the main part of our project, here is the concept of Course Outcome (CO) and Program Learning Outcomes (PLO), each CO is mapped to a PLO, and through each PLO students are expected to learn from the course 'problem analysis, design, Skill implementation, etc. 'will be known. The system allows input from IEB to determine PLO requirements. The project will evaluate to see if the COS mapped PLOs are met for each student to assess student proficiency. Faculties then input COs for each of their students so that the system can map the COs to the PLO accordingly. PLOs are carefully and specially selected to ensure that students achieve the most in a course so that students can monitor their progress in each sector and pinpoint the areas where self-improvement and self-development are needed. We are hopeful that our software will help institutional students' progress, departmental performance and assist in the distribution and allocation of their improved resources.

OBJECTIVE OF THE PROJECT

Student progress monitoring is a practice that helps teachers continually evaluate the effectiveness of their learning and use student performance data to make more significant instructional decisions. If the rate at which a particular student is learning seems inadequate, the teacher can adjust the instruction. Our project seeks to create user-friendly software that will serve as a platform for many to improve the quality of education of students, faculty, and other members of the university and in advanced technology in the field of education. We believe that the information we have collected, evaluated, and equipped will lead to opportunities for greater advances in our education and will also make a significant contribution to computer science.

SCOPE OF THE PROJECT

This CHAPTER will discuss falls within the scope of the system. To recognize the importance or quality of the scope, we are to contemplate what the system will accomplish i.e. The purpose of the system and the desired requirements that are to be met.

The main purpose of implementing such a system is to improve and digitalize the old inefficient way. In the exciting system, there exists several human roles (e.g., faculty, students) that are to get the work done manually, not by computers. Therefore, the existing system takes more time to achieve the goals while compared to the newly proposed system. The new system accomplishes this by reducing the human roles of the system and making a computer which helps us to run the system and do the work of organizing, storing and instantly querying the essential data.

- Storing useful data
- Securing the data by restricting access to the system.
- Instantly insert/ update/ delete data from the database
- Generate dynamic charts from the data using defined or dynamic parameters.
- Reporting
- Project management

CHAPTER 2

REQUIREMENT ANALYSIS

- RICH PICTURE AS-IS
- SIX ELEMENTS AS IS
- PROCESS DIAGRAM AS-IS
- PROBLEM ANALYSIS
- RICH PICTURE TO-BE
- SIX ELEMENTS TO-BE
- PROCESS DIAGRAM TO BE

RICH PICTURE (AS-IS)

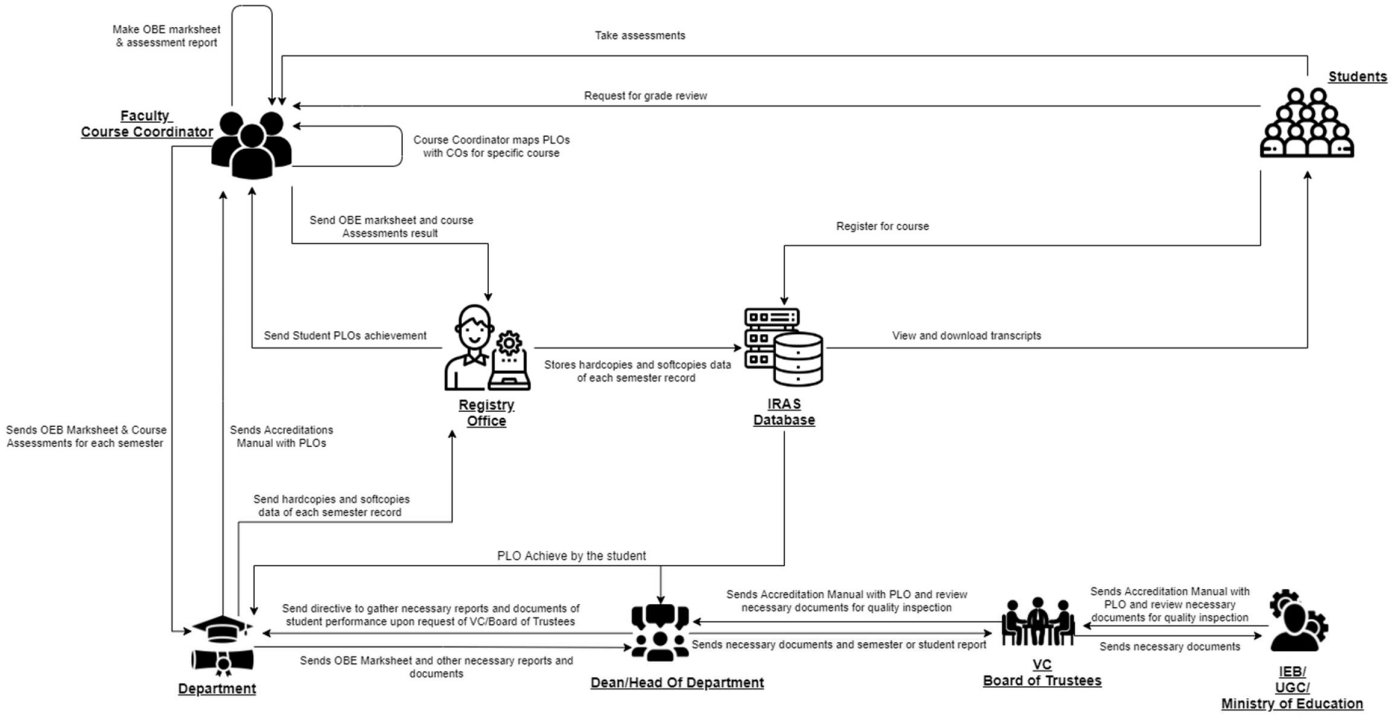


Figure 2.1: Rich Picture (As-Is)

SIX ELEMENTS (AS-IS)

Process	System Roles					
	Human	**Non-Comp Hardware**	**Computing Hardware**	**Software**	**Database**	**Network & Communication**
Map Course Outcomes (COs) to Program Learning Outcomes (PLOs)	**IEB/UGC/ Ministry of Education:** 1. Send Accreditation Manual with PLOs defined to VC/ Board Of trusties. **VC/ Board Of trusties** 1. Receive Accreditation Manual from IEB. 2. Send the Accreditation manual to Department Staff. **Head of Department / Dean of School:** 1. Send the Accreditation manual to Department Staff. 2. Direct Department Staff to tell Course Instructors and Coordinators to design Course Outline and Course Assessment Reports. **Department:** 1. Send Course Instructors the Accreditation Manual with defined PLOs. **Course Instructor:** 1.Check if previous	**Pen and paper:** 1. Is used for noting down intermediate brainstorming ideas. **Board and marker:** 1. Is used for noting down intermediate brainstorming ideas.	**Computer:** 1. Course Coordinators use computers to make softcopies of Course Outcomes (COs) of the specific courses they are experts in. **Printer:** 1. To print out hardcopies of Course Outcomes (COs).	**MS Word:** 1. Course Coordinators use MS Word to make a detailed course outline and Course Assessment Reports with Course Outcomes (COs) mapping to Program Learning Outcomes (PLOs). **Excel Sheet:** 1. Excel Sheet is used by Course Coordinators to map specific questions in the Midterm, Final exams and Project work to specific Course Outcomes (COs).	**IRAS Database server:** 1. IRAS uses a database server to store and maintain student grades' information.	1. Use the internet and emails to communicate with UGC/IEB or other stakeholders to discuss important topics related to mapping Course Outcomes to Program Learning Outcomes. **Others:** 1. Use phones or physical means with stakeholders to discuss important topics related to mapping Course Outcomes to Program Learning Outcomes.

	course content is present form register office, otherwise make new course content. 2. List COs. 3. Map Course Content to Course Outcomes (COs). 4. Map COs to PLOs. 5. Map COs to specific questions of Mid-term, Final Exams questions and Project Work. 6. Starting to design course assessment report using course outline, Course Content and COs. **Register Office:** 1.Send course content to course instructor if available otherwise send negative message.					
Check Number of student enrollment in a department	**Student:** 1. Student enroll in a specific Degree program. 2. Student information is sent to register's office. **Register Office:**	**Pen and Paper** 1. Sheet of number of students in a department is made along with student's information.	**Computer/ Phone:** 1. Uses computers to make softcopies of report or sheet of student information in departments.	**Coded Excel sheet:** 1.Department head or dean uses automated excel sheets to calculate	**Department Storage:** 1. Records of students' enrollment in the department.	**Internet/Mail:** **1.** An Online platform (such as Google Sheets) may be used for processing the student information data spreadsheet.

	1.Gather all the new student's information. 2. Assign the data in sheet of student information of designated departments. 3.Send the new update data to each department. **Department:** 1.Recieve the data of new student. 2.Update it in the existing database 3. Send the data to department heads or deans for further inspection **Department Head/Dean:** 1.Recieve the data from department. 2.Make calculation of number of new student enrollment comparing to previous cases. 3. Make calculation number of categorize students, such as merit base, physical aid and others		**Printer:** 1. Print hardcopies of report and sheet	the number student's in the departme nt. **MS Word:** 1. Used to make report softcopie s.	**Registr ar's Office Storage :** 1. Records of students ' enrollme nt for all the departm ents.	
Register for course	**Student:** 1. Login to IRAS 2. Student enroll in specific courses if all the pre requisite	**Pen and Paper** 1. Sheet of number of students enrolled for the course.	**Computer/ Phone:** 1. Uses computers to make softcopies of report or sheet of	**Coded Excel sheet:** 1.Instruct or uses automate d excel sheets for the	**Depart ment Storage :** 1. Records of students '	**Internet/Mail:** **1.** An Online platform (such as Google Sheets) may be used for processing the student

	courses are completed otherwise can't process end. 3. Request for bill 4. Receive for bill 5. Pay the bill **Register Office:** 1.Store request asked by the student and send the billing date 2. Receive billing data 3. If bill paid stored data is updated to database otherwise process end and student had to drop the course. 4. Send student information to Department. **Department:** 1.Recieve the data of enroll student. 2. Send the data of enroll student to course instructor. **Instructor** 1. Receive data of enrolled student. 2. Allocate space for the new student data in OEB marksheet.		student information enrolled for the course. **Printer:** 1. Print hardcopies of report and sheet	semester OEB markshe et. **MS Word:** 1. Used to make report softcopie s.	enrollme nt in the course. **Registr ar's Office Storage :** 1. Records of students ' enrollme nt in the course.	information data spreadsheet.
Record Student Assess ment Data	**Faculty/ Course Coordinator:** 1. Assign project work and	**Pen & Paper:** 1. Use pen & paper to	**Computer:** 1. Creating softcopies of records	**Excel Sheet:** 1. Record necessar y	**Depart ment Storage :**	**Internet:** 1. The Internet is used to communicate with IRAS to

	assignments according to course outline. 2. Take quizzes and exams throughout the semester according to course outline. 3. Record assessment data of students throughout the semester of each student for every assessment (quizzes, assignments, project, exams) on softcopies and hardcopies. 4. Record marks for each specific question in the midterms and final exams. 5. Calculate total marks of quizzes, assignments and midterm and final exams and assign final grades to each student of specific courses. 6. Convert finals and midterms marks. 7. Bring all the marks of every student for a course into a Marksheet. 8. Grade the student	record assessment data and marks obtained on physical paper in tabular format(hardcopies).	of all assessment data for specific courses are done on computers.	assessment data and final grades on Excel Sheets. **IRAS:** 1. Upload students' final grades to IRAS for viewing by students or the registrar's office.	1. Records of students' assessment data and final grades may be saved in the department office and registrar's office for future reference. **IRAS Database server:** 1. IRAS uses a database server to store and maintain student grades' information.	store final grades of students.

	according to current mark distribution if no change is needed else adjustment has been made. 9. Upload students' final grades on IRAS. 10. Send the Marksheet to the Department. 11. Send the Marksheet to admin to store in the database					
Produce OBE Marksheet & Course Assessment Report	**Faculty:** 1. Calculate total marks received for each CO by calculating the marks received for questions and/or other assessments mapped to COs. 2. Calculate total percentages received for each COs on the OBE Marksheet. 3. Declare if a student has achieved a specific CO (if CO percentage is greater than or equal to 40). 4. Declare if a student has received a PLO for a related CO. 5. Make a table giving the verdict and	**Pen and Paper** 1. OBE marksheet stored in hardcopy. Additional markings may be made to further separate between students.	**Computer/ Phone:** 1. Uses computers to make softcopies of the OBE Marksheet and Course Assessment Reports. **Printer:** 1. Print hardcopies of final versions of the OBE Marksheets and Course Assessment Reports.	**Coded Excel sheet:** 1.Faculty /Course Coordinator uses automated excel sheets to calculate the student's success/ failure in achieving PLOs. **MS Word:** 1. Used to make Course Assessment Report softcopies.	**Department Storage:** 1. Records of students' assessment data and final grades will be saved in the department for future reference. **Registrar's Office Storage:** 1. OBE Marksheets, Course Assessment Reports and other documents	**Internet/Mail:** **1.** An Online platform (such as Google Sheets) may be used for processing the OBE assessment data spreadsheet.

	analysis of how many students were able to receive a certain CO and PLO and other documents containing necessary information and data. 6. Design Course Assessment Report using Course Outline, Course Content and Course Outcomes. 7. Send the final version of the OBE Marksheet to the Dept. Office. **Department Office:** 1. Send the OBE marksheet, Course Assessment Report and others to the Registrar's Office. 2. Store the OBE Marksheet and Course Assessment Report in the department. **Registry Office:** 1. Stores the OBE Marksheet and Course Assessment Reports and other				submitted by the department is stored for future reference.	

	documents and reports in the Registrar's Office.					
View grades and download Transcripts	**Students**: 1. Log into IRAS. 2. Search semester wise result for intended semester. 3. See grades for specific semesters. 4. Download transcript through browser into hard disk. **Dean/DOH:** 1. Log into IRAS. 2. Search semester wise result for intended semester for a specific student. 3. See grades for specific semesters. 4. Download transcript through browser into hard disk. **Faculty/Higher Officials:** 1. Request register office for transcript of particular student or semester of a particular course. 2. Receive transcript of particular student or semester of a particular course.	**Pen and Paper** 1. Tabulated transcripts may be printed onto paper. Hardcopy is used as the primary source of truth during applications and other paperwork.	**Computer/ Phone:** 1. Used for accessing IRAS. **Printer:** 1. Used to print the tabulated transcript. Prints tabulated transcripts.	**IRAS:** 1. **Store's** letter grades of each completed course 2. Provides the online user **interface** for viewing grades and transcripts.	**Registrar's Office Storage:** 1. Student information is kept in admin in hardcopies for future reference. **IRAS Database Server:** 1. A Database Management Service is used to store, maintain, edit and receive student grades information in IRAS. **Web Server:** 1. User interface and website pages are served using a remote web server.	**Internet/ Email** 1. The **Internet** is used to communicate with IRAS to store final grades of students. 2. Softcopies may be **mailed**.

	Registry Office: 1. Access IRAS. 2. View students' grades if and when it's necessary. 3. Download their transcripts. 4. Send transcript					
View Records OBE Marksheets, Course Assessment Reports over a time period for inspection and analysis of student performance trend	**IEB/ UGC:** 1. Inform the VC of a deadline within which OBE Marksheets, Course Assessment Reports and other documents are needed for quality inspection to make necessary improvements to degree programs. 2. Inform the university head if govt. official will visit the campus. 3. Visit university and relevant depts to receive the necessary documents and reports. **Head of Dept/Dean of School:** 1. Request to view records of OBE Marksheets, Assessment Reports to analyze students'	**Pen and Paper:** 1. May be used for noting/marking down key points of the report. 2. Hardcopies of reports may be used.	**Computer:** 1. Used to display OBE Marksheet and Course Assessment Report softcopies. 2. Send OBE and Course Assessment Reports to other computers.		**Department Records** 1. Retrieval of OBE marksheets and Course Assessment reports when needed. 2. Stores records on stakeholders' interpretation of student performance trends.	**The internet:** 1. OBE marksheets and course assessment reports may be **mailed** online. 2. Online platforms such as Google Docs/Sheets display reports of softcopies.

	performance trends. 2. Direct Department Staff to gather necessary documents, OBE Marksheets, Assessment report for a given time-period specified by govt. officials. 3. Receive the necessary documents gathered by the dept. 4. Evaluate the need to change/ improve the department's educational resources based on students' performance trends. **VC/Board of Trustees:** 1. Request to view records of OBE Marksheets, Assessment Reports to analyze students' performance trends. **Departmental :** 1. Gather necessary OBE Marksheets, Assessment Reports & other documents. 2. Provide all the necessary documents to govt. officials.					

	Faculty/Higher Officials: 1. Request register office for OBE marksheet semester of a particular course. 2. Receive OBE marksheet semester of a particular course. **Registry Office:** 1. Access IRAS. 2. Gather OBE marksheet from database. 3. Send OBE marksheet.					
Request for review and change of grades	**Students:** 1. Request for grade change and review to faculty. **Faculty/ Course Coordinator:** 1. Check exam papers and other assessments upon request. 2. If change needs to be made, send a grade change request of a specific student to register office. If not, end the process. **Register Office:** 1. Receive a request to change the grade of a specific student.	**Pen and Paper:** 1. May be used to note down key points or marks on the students' answer sheets.	**Computer/ Phone:** 1. Used for communicating with the faculty.	**IRAS**: 1. Used by the admin for changing the grade.	**IRAS server:** 1. Update student grade data. **Department Storage:** 1. Update student grade data. **Registrar's Office Storage:** 1. Update student grade data.	**Internet:** 1. Email is primarily used for communication. **Phone:** 1. May be used for communication.

	2. Change grade of student based on Faculty request.					

PROCESS DIAGRAM (AS-IS)

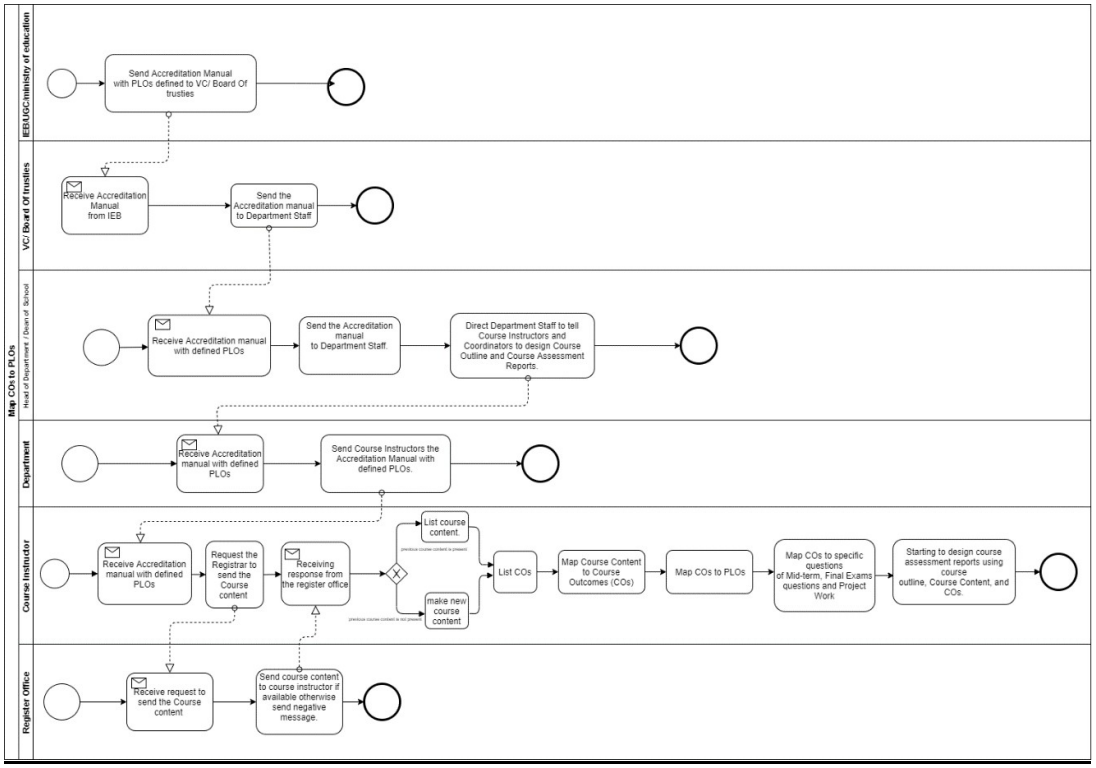


Figure 2.3: Process Diagram for Map COs to PLOs

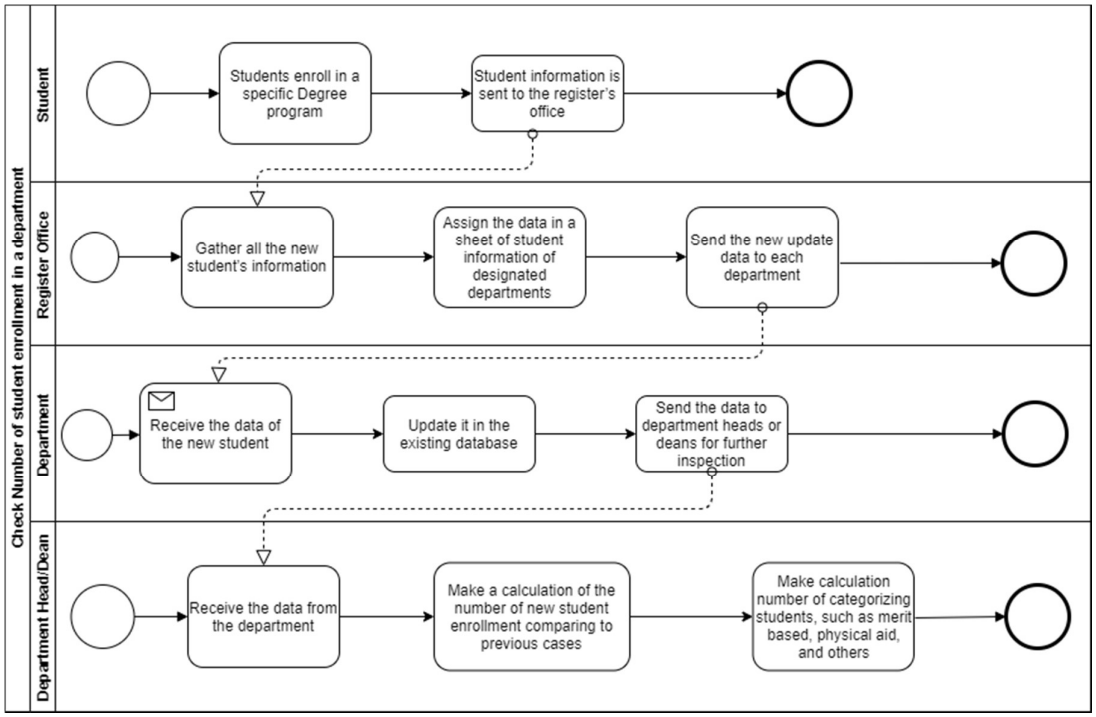


Figure 2.4: Process Diagram for Check Number of student enrollment in a department

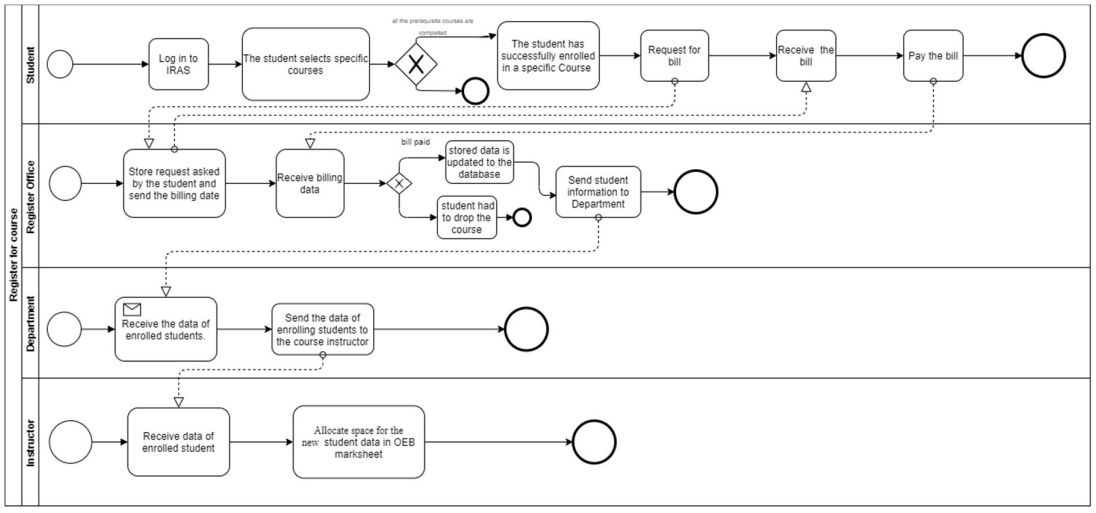


Figure 2.5: Process Diagram for Register for course

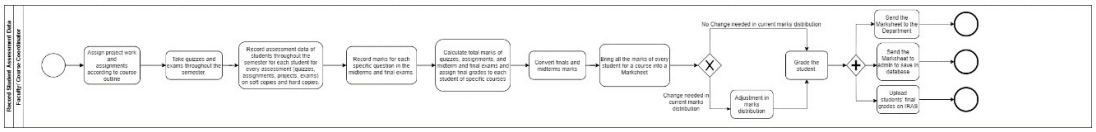


Figure 2.6: Process Diagram for Record Student Assessment Data

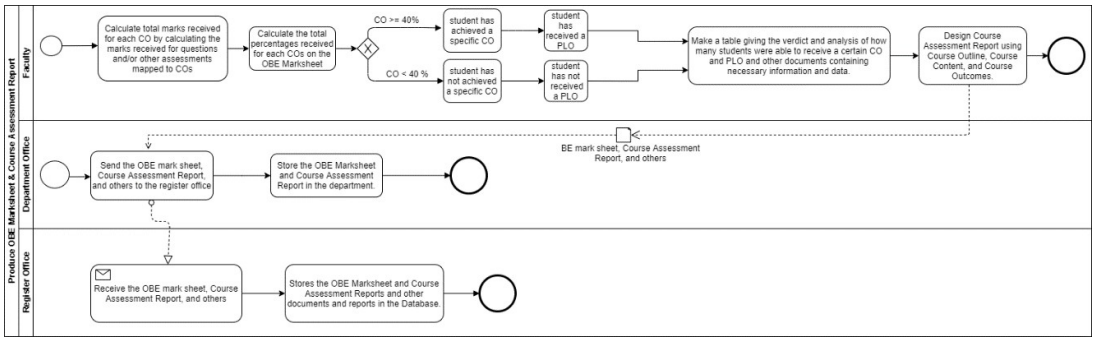


Figure 2.7: Process Diagram for Produce OBE Marksheet & Course Assessment Report

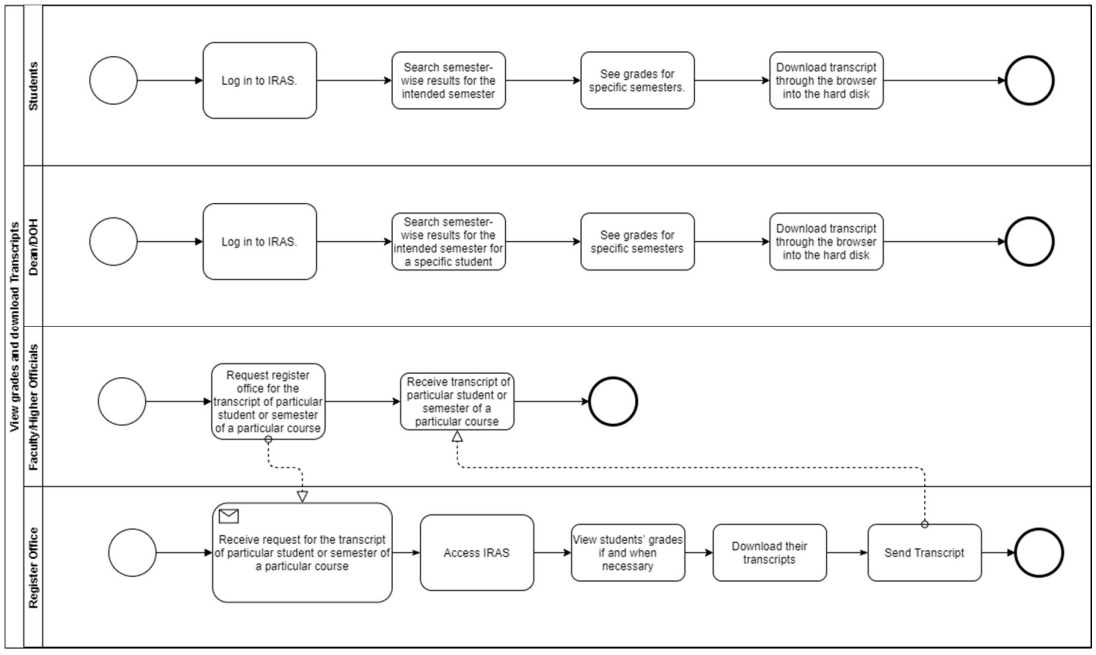


Figure 2.8: Process Diagram for View grades and download Transcripts

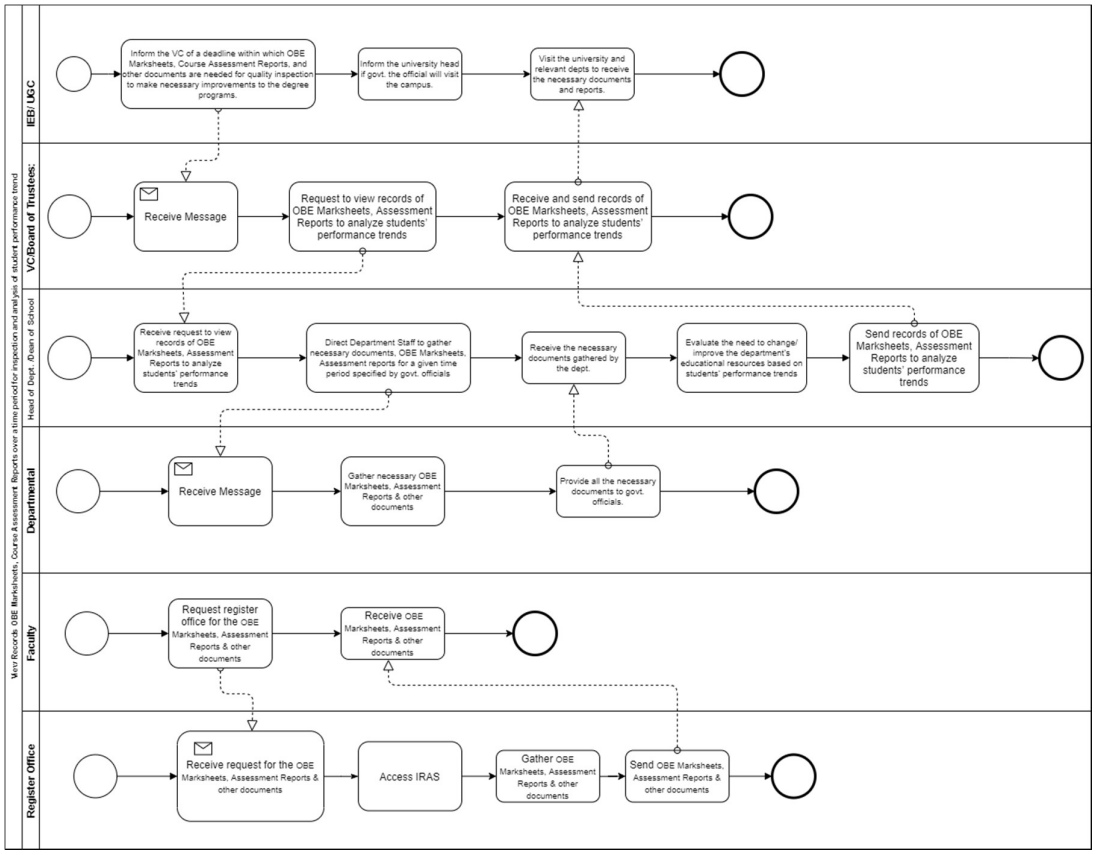


Figure 2.9: Process Diagram for View Records OBE Marksheets, Course Assessment Reports over a time period for inspection and analysis of student performance trend

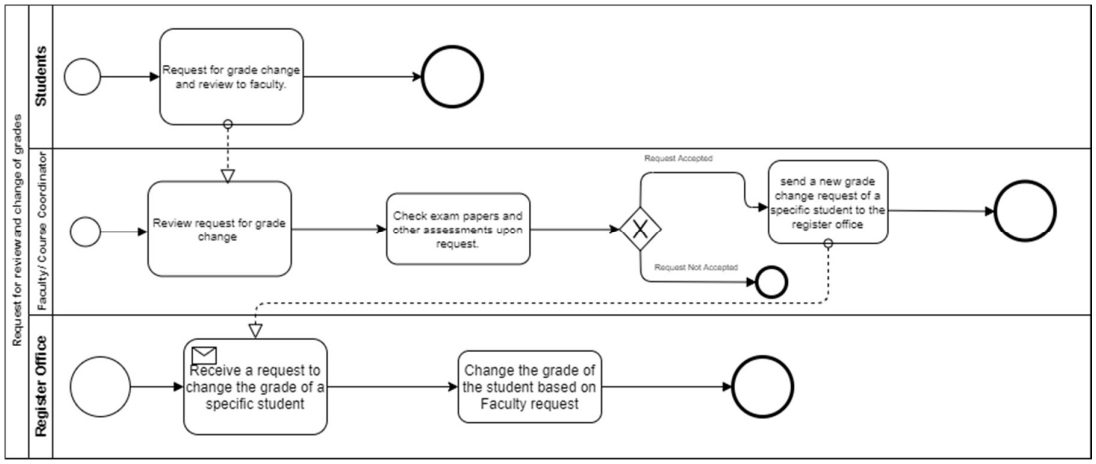


Figure 2.10: Process Diagram for Request for review and change of grades

PROBLEM ANALYSIS

Process Name	Stakeholders	Concerns (Problems)	Analysis (reason of the problem)	Proposed solution
Map course outcomes (COs) to program Learning Outcomes (PLOs)	1.Faculties	1.VC has to collect and send CO and PLO data to the Dean. 2.Dean sends data to department head and then it is passed to department. 3.Course instructor implements CO and PLO in their course.	The process is very complicated and time consuming as faculties must wait for other non-essential stakeholders to implement PLO and CO in their courses.	We can eliminate the involvement of department by giving faculties direct access to update PLOs and COs in our software and department head to update the PLO after further inspection
Check number of student enrollment in a department	1.Department 2.Dean	1.Register office collects all the new student's information. 2.Register office sends updated data to each department. 3.Department updates data to database. Then sends	Same information is being send to different stakeholders individually. Which creates unnecessary repetition. Which makes the overall process	We can make this information centralized, so that all the stakeholders can see latest information any time. We can also generate custom comparison Graphs/charts for any

		new data to Dean. 4. Dean makes calculation to see student enrollment comparison.	time consuming.	individual stakeholder.
Record Student Assessment Data to SPM	1.Faculty	1. Faculty had to calculate the total assessments marks and convert finals and midterms Marks of each student. 2. Bring all the marks of every student for a course into a Marksheet. 3. Grade the student	Making all the calculation manually is too much time consuming and chances human error is greater.	All the calculation can be done in SPM and graded accordingly. Adjustment in marks distribution can easily be made if change is needed
Produce OBE Marksheet & Course Assessment Report	1.Faculty	1. All calculation have to done manually 2. Need to send data to register office to update database	Making all the calculation manually and waiting for register office to update data is too much time consuming and chances	Calculation can be done by the help of the software and OEB marksheet can directly uploaded to database using the software

			human error is greater.	
View records, OBE marksheets and Course assessment report	1.IEB/UGC 2.Dean 3.VC 4.Faculty	1. Faculty can't access the OEB marksheet directly 2. Calculations have to done manually and charts have to make manually to make comparison	Too many manual processes which takes time and resource. Therefore, lowers overall efficiency	We will generate automated charts, graphs and report for relevant stakeholder. we can collect most of the relevant data directly from IRAS, which will eliminate any extra steps. Faculty can view OEB marksheet of past semesters.
Request for review and change grade	1.Faculty	1. Faculty need to send request to register office to change the grade	Grade change could be done by faculty. Sending request to register office for grade upgrade adds extra work.	By giving access to change grade in our system we could eliminate the involvement of register office in our system.

RICH PICTURE (TO-BE)

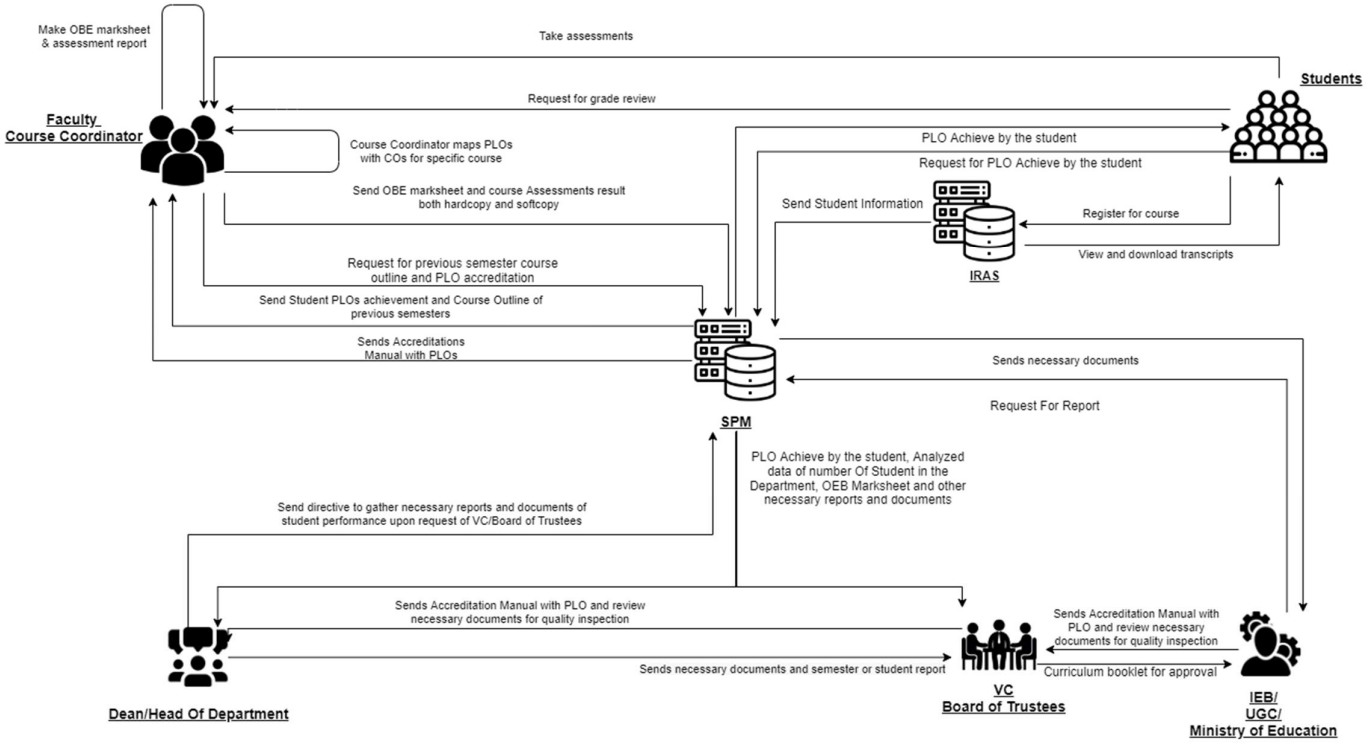


Figure 2.11: Rich Picture (To-Be)

SIX ELEMENTS (TO-BE)

Process	System Roles					
	Human	**Non-Comp Hardware**	**Computing Hardware**	**Software**	**Database**	**Network & Communication**
Map Course Outcomes (COs) to Program Learning Outcomes (PLOs)	**IEB/UGC/ Ministry of Education:** 1. Send Accreditation Manual with PLOs defined to VC/ Board Of trusties. **VC/ Board Of trusties** 1. Receive Accreditation Manual from IEB. 2. Send the Accreditation manual to Department Staff. **Head of Department / Dean of School:**	**Pen and paper:** 1. Is used for noting down intermediate brainstorming ideas. **Board and marker:** 1. Is used for noting down intermediate brainstorming ideas.	**Computer:** 1. Course Coordinators use computers to make softcopies of Course Outcomes (COs) of the specific courses they are experts in. **Printer:** 1. To print out hardcopies of Course Outcomes (COs).	**MS Word:** 1. Course Coordinators use MS Word to make a detailed course outline and Course Assessment Reports with Course Outcomes (COs) mapping to Program Learning	**IRAS Database server:** 1. IRAS uses a database server to store and maintain student grades' information. **SPM database:** 1. Records of PLOs	1. Use the internet and emails to communicate with UGC/IEB or other stakeholders to discuss important topics related to mapping Course Outcomes to Program Learning Outcomes. **Others:** 1. Use phones or physical means with stakeholders to discuss important topics

	1. Receive Accreditation Manual from IEB. 2. Send the Accreditation manual to SPM 3. Direct Department Staff to tell Course Instructors and Coordinators to design Course Outline and Course Assessment Reports. **Course Instructor:** 1.Check if previous course content is present from SPM, otherwise make new course content.			Outcomes (PLOs). **Excel Sheet:** 1. Excel Sheet is used by Course Coordinators to map specific questions in the Midterm, Final exams and Project work to specific Course Outcomes (COs).		related to mapping Course Outcomes to Program Learning Outcomes.

	2. List COs. 3. Map Course Content to Course Outcomes (COs). 4. Map COs to PLOs. 5. Map COs to specific questions of Mid-term, Final Exams questions and Project Work. 6. Starting to design course assessment report using course outline, Course Content and COs.					
Check Number of student enrollment	**Student:** 1. Student enroll in a specific Degree program.	**Pen and Paper** 1. Sheet of number	**Computer/ Phone:** 1. Uses computers to	**Coded Excel sheet:** 1.Department head	**IRAS database** 1. Records of	**Internet/ Mail:** **1.** An Online platform (such as

in a department from SPM	2. Student information is sent to SPM from IRAS. **Department Head/Dean:** 1.Recieve the data from SPM 2. School-wise, department-wise and program-wise student enrollment comparison can be seen. **VC/Board of Trustees:** 1.Recieve the data from SPM 2. School-wise, department-wise and program-wise student enrollmen	of students in a department is made along with student's information.	make softcopies of report or sheet of student information in departments. **Printer:** 1. Print hardcopies of report and sheet	or dean uses automated excel sheets to calculate the number student's in the department. **MS Word:** 1. Used to make report softcopies.	students' enrollment in the department. **SPM database:** 1. Records of students' enrollment for all the departments.	Google Sheets) may be used for processing the student information data spreadsheet. **2.** Internet to access to SPM

	t comparison can be seen.					
Register for course	**Student:** 1. Login to IRAS 2. Student enroll in a specific course if all the pre requisite courses are completed otherwise can't process end. 3. Request for bill 4. Receive for bill 5. Pay the bill **Instructor** 1. Receive data of enrolled student. 2. Add student data in	**Pen and Paper** 1. Sheet of number of students enrolled for the course.	**Computer/ Phone:** 1. Uses computers to make softcopies of report or sheet of student information enrolled for the course. **Printer:** 1. Print hardcopies of report and sheet	**Coded Excel sheet:** 1.Instructor uses automated excel sheets for the semester OEB marks heet. **MS Word:** 1. Used to make report softcopies.	**Department Storage:** 1. Records of students' enrollment in the course. **IRAS database:** 1. Records of students' enrollment in the course.	**Internet/ Mail:** **1.** An Online platform (such as Google Sheets) may be used for processing the student information data spreadsheet.

	OEB markshee t.					
Recor d Stude nt Asses sment Data to SPM	**Faculty/ Course Coordina tor:** 1. Assign project work and assignme nts according to course outline. 2. Take quizzes and exams throughou t the semester according to course outline. 3. Record assessme nt data of students throughou t the semester of each student for every assessme nt (quizzes, assignme nts, project, exams) on	**Pen & Paper:** 1. Use pen & paper to record assess ment data and marks obtaine d on physica l paper in tabular format(hardco pies).	**Comput er:** 1. Creating softcopie s of records of all assessm ent data for specific courses are done on compute rs.	**Excel Sheet:** 1. Recor d neces sary asses sment data and final grades on Excel Sheet s. **IRAS:** 1. Uploa d studen ts' final grades to IRAS for viewin g by studen ts or the registr ar's office. **SPM** 1. Uploa	**SPM:** 1. Recor ds of stude nts' asses sment data and final grade s may be saved in the SPM for future refere nce. **IRAS Datab ase serve r:** 1. IRAS uses a datab ase server to store and maint ain	**Internet:** 1. The Internet is used to communic ate with IRAS to store final grades of students. 2.Internet to access SPM

	softcopies and hardcopies. 4. Record marks for each specific question in the midterms and final exams. 5. SPM calculate total marks of quizzes, assignments and midterm and final exams and assign final grades to each student of specific courses. 6. Convert finals and midterms marks. 7. Bring all the marks of every student for a			d student from IRAS to SPM	student grades' information.	

	course into a Markshee t. 8. Grade the student according to current mark distributio n if no change is needed else adjustme nt has been made. 9. Submit students' final grades and markshee t on SPM.					
Produ ce OBE Marks heet & Cours e Asses sment Repor t to SPM	**Faculty:** 1. Upload Marks in SPM to calculate total marks received for each CO by calculatin g the marks received for	**Pen and Paper** 1. OBE marksh eet stored in hardco py. Additio nal markin gs	**Comput er/ Phone:** 1. Uses compute rs to make softcopie s of the OBE Markshe et and Course Assess ment	**Code d Excel sheet:** 1.Facu lty/Co urse Coordi nator uses autom ated excel sheets to	**SPM Stora ge:** 1. Recor ds of stude nts' asses sment data and final grade s	**Internet/ Mail:** **1.** An Online platform (such as Google Sheets) may be used for processin g the OBE assessme nt data

	questions and/or other assessments mapped to COs. 2. SPM calculate total percentages received for each COs on the OBE Markshee t. 3. Declare if a student has achieved a specific CO (if CO percentag e is greater than or equal to 40). 4. Declare if a student has received a PLO for a related CO. 5. SPM make a	may be made to further separat e betwee n student s.	Reports from SPM. **Printer:** 1. Print hardcopi es of final versions of the OBE Markshe ets and Course Assess ment Reports from SPM	calcul ate the studen t's succe ss/ failure in achiev ing PLOs from SPM **MS Word:** 1. Used to make Cours e Asses sment Report softco pies. **SPM** 1. Store CLO and PLO inform ation to SPM	will be saved in the depar tment for future refere nce in the SPM	spreadshe et. 2. Internet to Access SPM

	table giving the verdict and analysis of how many students were able to receive a certain CO and PLO and other documents containing necessary information and data. 6. Design Course Assessment Report using Course Outline, Course Content and Course Outcomes. 7. Submit the final version of the OBE Markshee t to the SPM					

View grades and download Transcripts	**Students**: 1. Log into SPM. 2. Search semester wise result for intended semester. 3. See grades for specific semesters. 4. Download transcript through browser into hard disk. **Dean/DOH:** 1. Log into SPM. 2. Search semester wise result for intended semester for a specific student. 3. See grades for specific semesters.	**Pen and Paper** 1. Tabulated transcripts may be printed onto paper. Hardcopy is used as the primary source of truth during applications and other paperwork.	**Computer/ Phone:** 1. Used for accessing IRAS. **Printer:** 1. Used to print the tabulated transcript. Prints tabulated transcripts.	**IRAS:** 1. **Store's** letter grades of each completed course 2. Provides the online user **interface** for viewing grades and transcripts. **SPM** 1. Store transcript data	**IRAS Database Server:** 1. A Database Management Service is used to store, maintain, edit and receive student grades information in IRAS. **Web Server:** 1. User interface and website pages	**Internet/ Email** 1. The **Internet** is used to communicate with IRAS to store final grades of students. 2. Softcopies may be **mailed**.

	4. Download transcript through browser into hard disk. **Faculty/Higher Officials:** 1. Log into SPM. 2. Search semester wise result for intended semester for a specific student. 3. See grades for specific semesters. 4. Download transcript through browser into hard disk.				are served using a remote web server. **SPM Storage:** 1. Records of students' assessment data and final grades will be saved in the department for future reference in the SPM	
View Records	**IEB/ UGC:** 1. Login to SPM	**Pen and Paper:** 1. May be	**Computer:** 1. Used to display	**SPM** 1.Store information	**Department Records**	**The internet:** 1. OBE marksheet

OBE Marksheets, Course Assessment Reports over a time period for inspection and analysis of student performance trend from SPM	2. View records of OBE marksheet course Assessment report over time period for inspection and analysis of student performance trend 3. Download OBE marksheet course assessment report **Head of Dept/Dean of School:** 1. Login to SPM 2. View records of OBE marksheet course Assessment report over time period for inspection and analysis of student	used for noting/ marking down key points of the report. 2. Hardcopies of reports may be used.	OBE Marksheet and Course Assessment Reports softcopies from SPM. 2. Send OBE and Course Assessment Reports to SPM. 3. View OBE Marksheet from SPM	of OBE into SPM	1. Retrieval of OBE marksheets and Course Assessment reports when needed from SPM 2. Stores records on stakeholders' interpretation of student performance trends from SPM	s and course assessment reports may be **mailed** online. 2. Online platforms such as Google Docs/Sheets display reports of softcopies . 3. Internet to access SPM

	performance trend 3. Download OBE markshee t course assessme nt report performance trends . **VC/Board of Trustees:** 1. Login to SPM 2. View records of OBE markshee t course Assessm ent report over time period for inspection and analysis of student performance trend 3. Download OBE markshee t course assessme nt report **Faculty/Higher Officials:**					

	1. Login to SPM 2. View records of OBE marksheet course Assessment report over time period for inspection and analysis of student performance trend 3. Download OBE marksheet course assessment report					
Request for review and change of grades	**Students:** 1. Request for grade change and review to faculty. **Faculty/ Course Coordinator:** 1. Check exam	**Pen and Paper:** 1. May be used to note down key points or marks on the student's' answer sheets.	**Computer/ Phone:** 1. Used for communicating with the faculty.	**SPM**: 1. Used by the admin for changing the grade.	**SPM server:** 1. Update student grade data.	**Internet:** 1. Email is primarily used for communication. **Phone:** 1. May be used for communication.

	papers and other assessments upon request. 2. If change needs to be made, grade is changed in SPM and re-submitted . If not, end the process.					

PROCESS DIAGRAM (TO-BE)

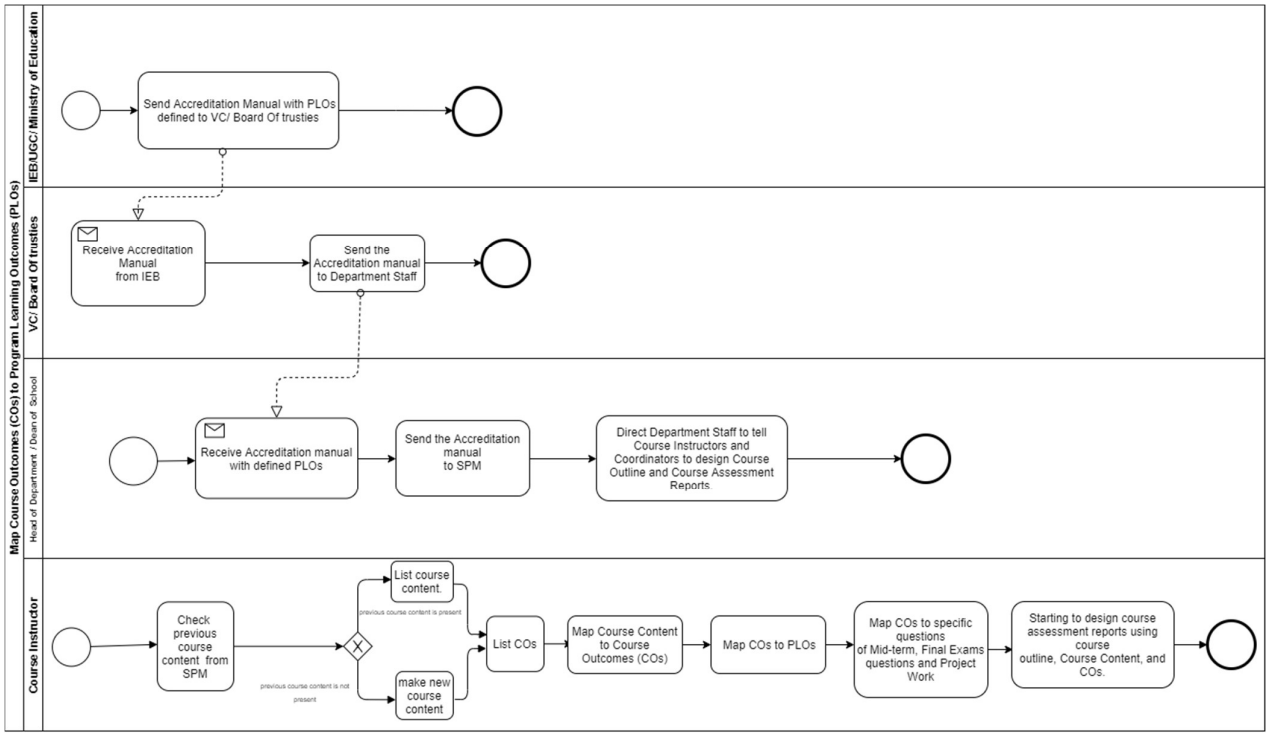


Figure 2.12: Process diagram for Map Course Outcomes (COs) to Program Learning Outcomes (PLOs)

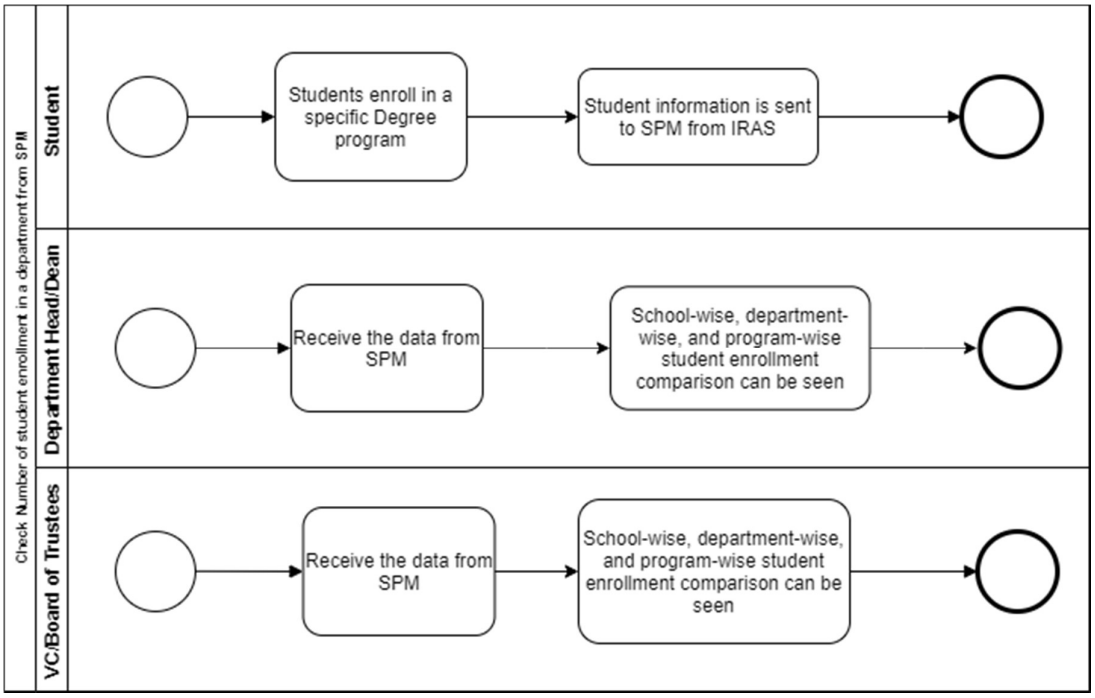


Figure 2.13: Process diagram for Check Number of student enrollment in a department from SPM

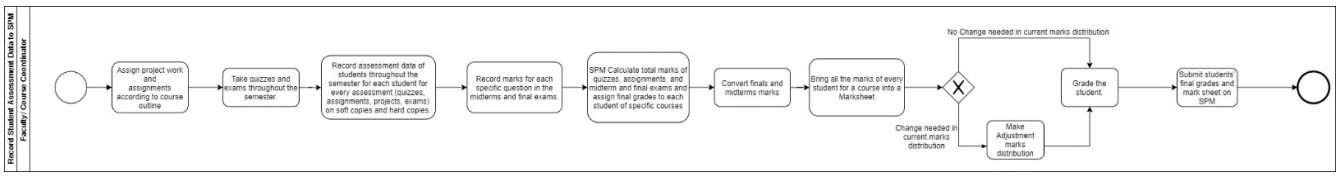


Figure 2.14: Process diagram for Record Student Assessment Data to SPM

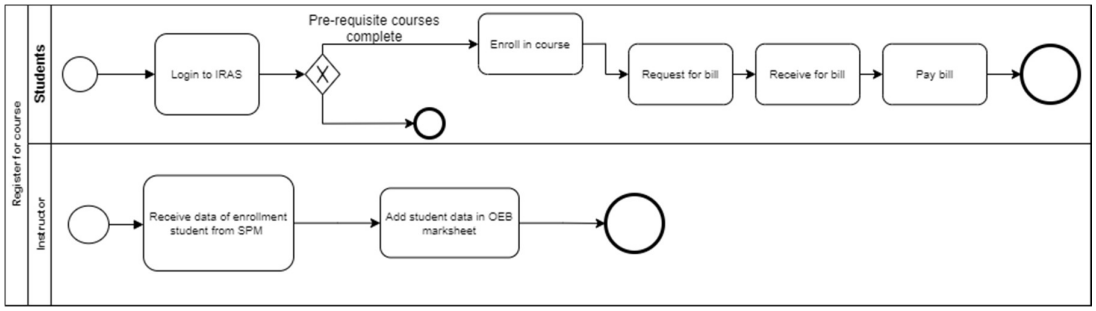


Figure 2.15: Process diagram for Register for course

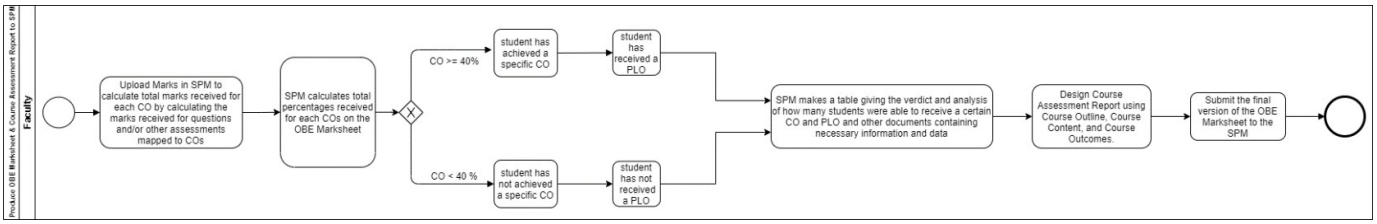


Figure 2.16: Process diagram for Produce OBE Marksheet & Course Assessment Report to SPM

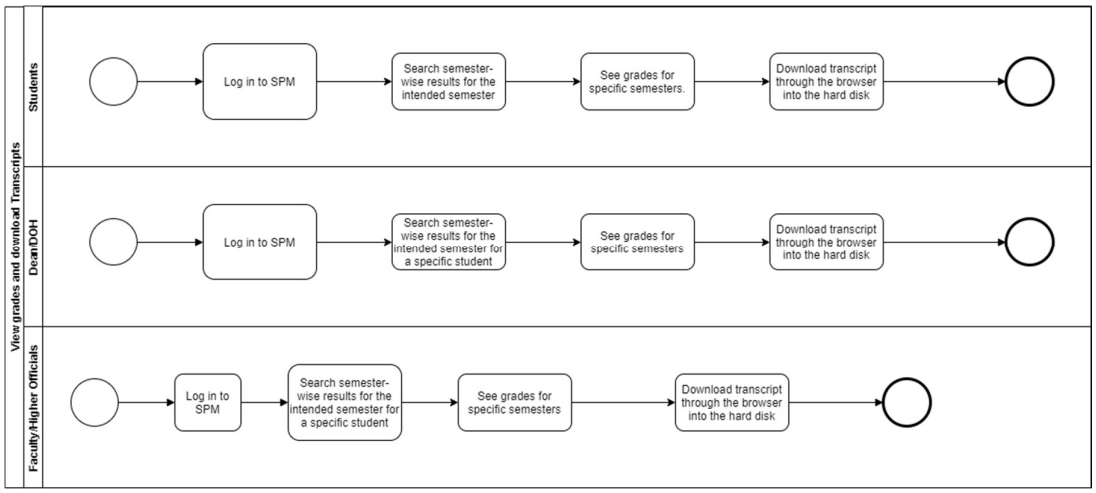


Figure 2.17: Process diagram for View grades and download Transcripts

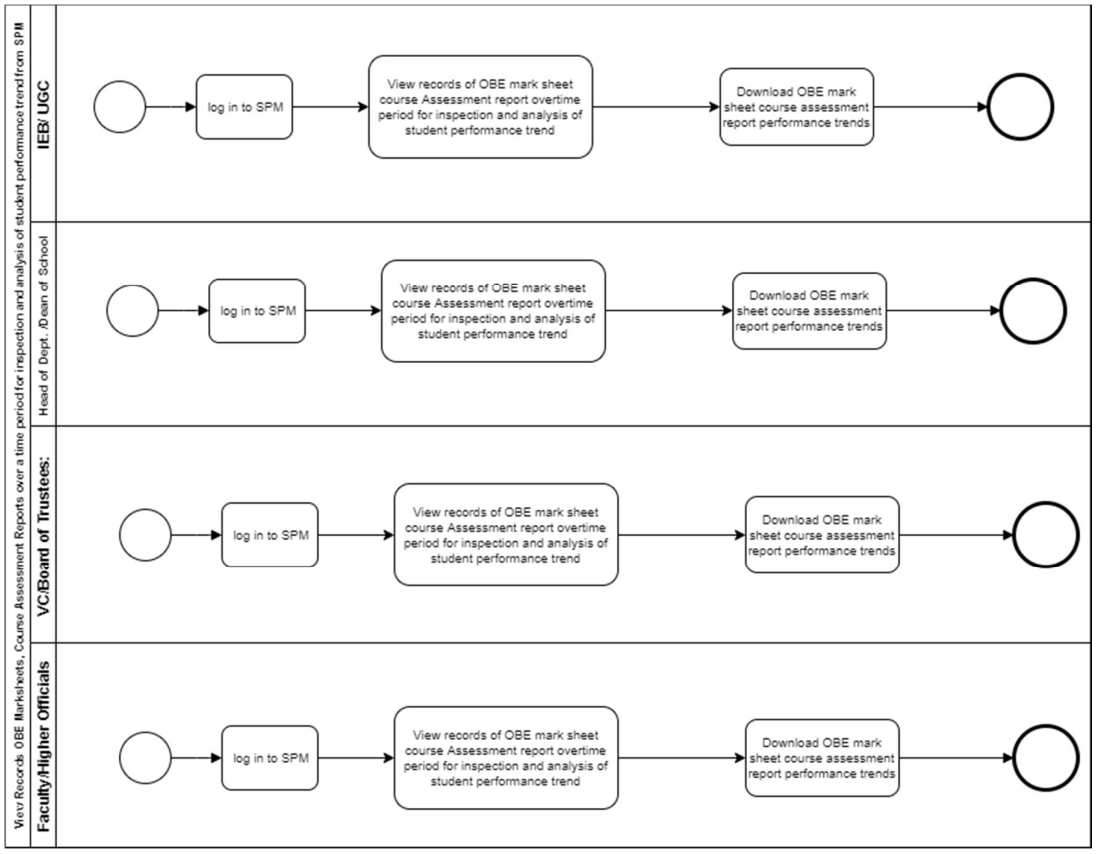


Figure 2.19: Process diagram for View Records OBE Marksheets, Course Assessment Reports over a time period for inspection and analysis of student performance trend from SPM

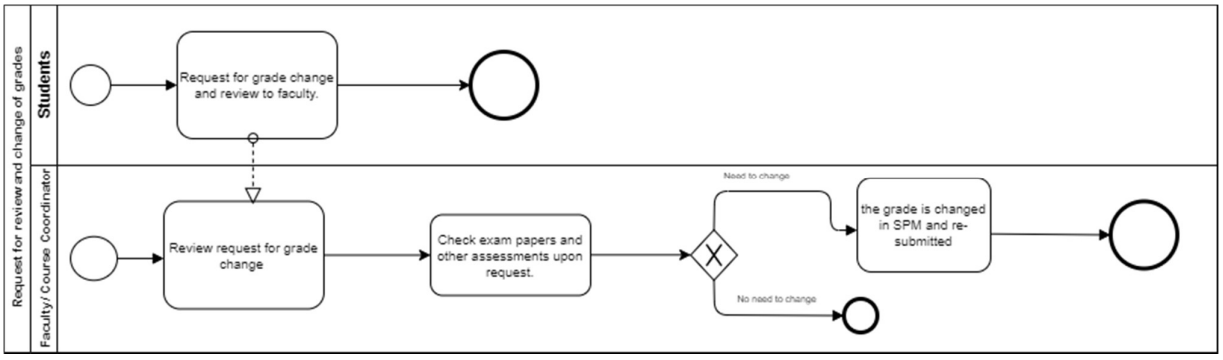


Figure 2.20: Process diagram for Request for review and change of grades

CHAPTER 3

LOGICAL SYSTEM DESIGN

- BUSINESS RULE
- ENTITY RELATIONSHIP DIAGRAM
- ENTITY RELATIONSHIP DIAGRAM TO RELATIONAL SCHEMA
- NORMALIZATION
- DATA DICTIONARY

BUSINESS RULE

A School must associates with many departments. And A Department must contain one or more Programs. However, each department must associates with one school. Department must be run by a department head and a school should be managed by a dean. VC should take care of all schools. A dean and department head can also play a role as a faculty.

A student must be enrolled in at least one program. However, each program may be enrolled by many students. A student may register in a section of a particular course of a semester. However, each course must have one or more sections and the section must be instructed by one faculty.

A section must be assigned with many assessments and each assessment contains multiple questions which are mapped with CO of the course. A student may seat for one or more assessments.

Each program must have multiple PLOs. COs must be mapped by one or more PLOs. The CO must be updated by the faculty for each course, and before the semester begins, the COs must be mapped to the PLOs so that the faculty can determine whether or not each student has achieved the required PLOs.

ENTITY RELATIONSHIP DIAGRAM

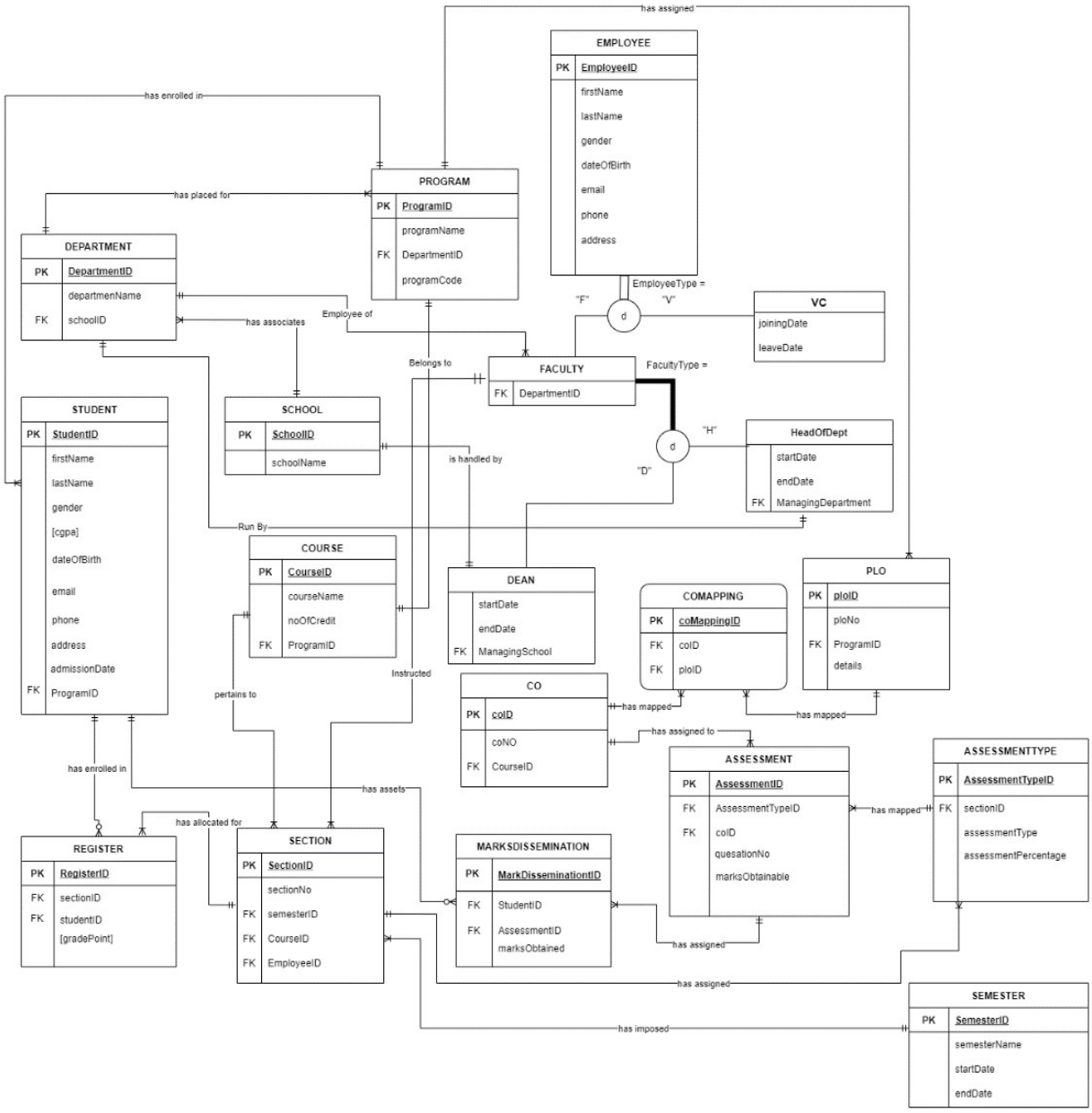


Figure 3.1: Entity Relationship Diagram of SPM

ENTITY RELATIONSHIP DIAGRAM TO RELATIONAL SCHEMA

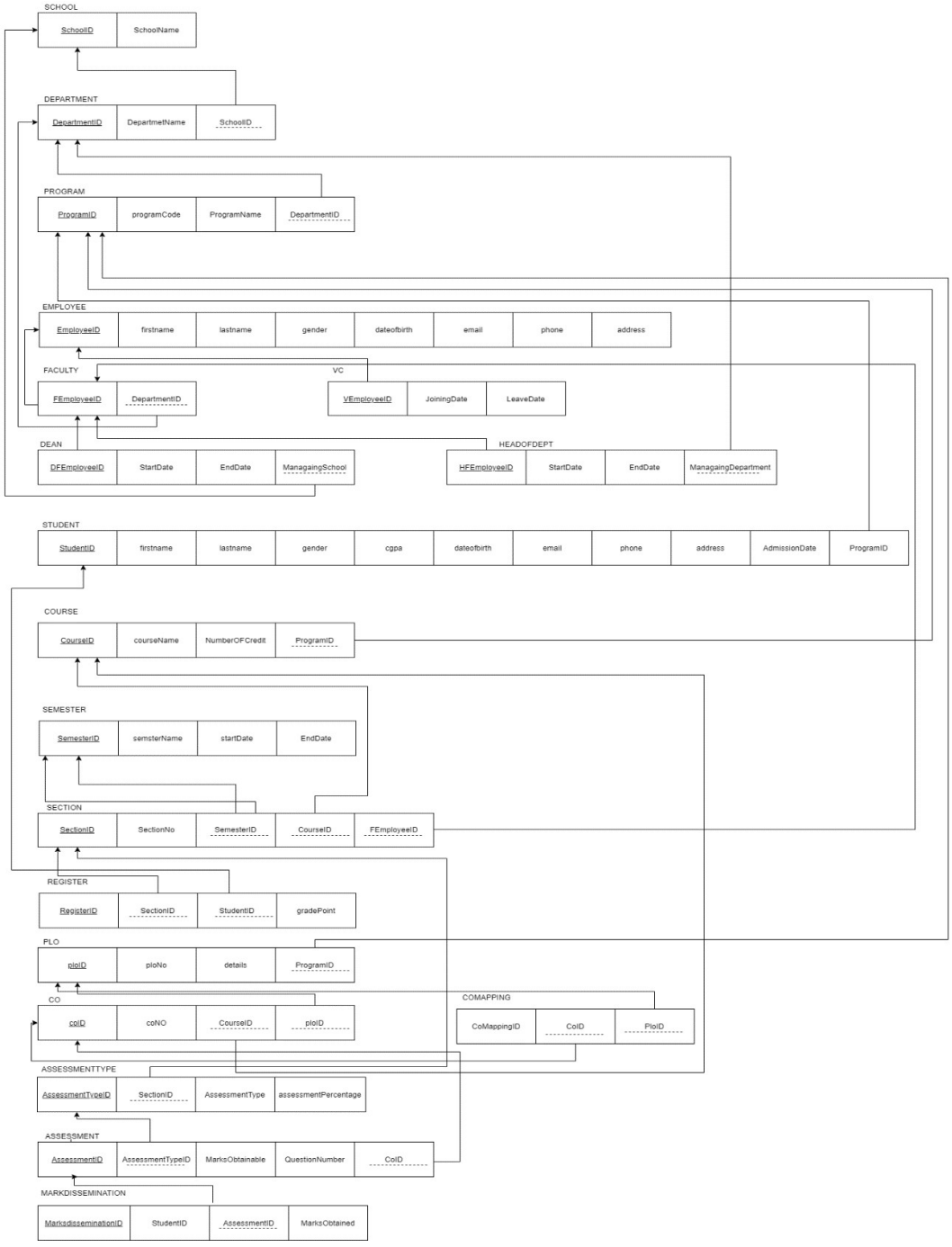


Figure 3.2**: Relational Schema Diagram of SPM**

NORMALIZATION

schoolID -> SchoolName

departmentID -> DepartName, SchoolID

programID -> ProgramName,ProgramCode , departmentID

employeeID-> eFirstName, eLastName, egender, eDateofBirth, eEmail, ePhoneNo, eAddress

femployeeID -> departmentID

veEmployeeID -> joiningDate, leaveDate

dfEmployeeID -> dfStartDate, dfendDate, managiningSchool

hfEmployeeID -> hfstartDate, hfendDate, manageDepartment

studentID -> sFirstName, sLastName, sGender, cgpa, sDateOfBirth, sEmail, sPhone, sAddress, admissionDate, programID

courseId -> courseName, numberOfCredit, programID,

semesterId -> semesterName, startDate, endDate

sectionId -> sectionNo, semesterId, courseID, employeeID

regesterID -> semesterID, sectionId, studentID, gradepoint

ploID -> ploNo, details, programID

coID -> coNo, courseID, ploID

AssessmentTypeID -> semesterID, sectionID, AssessmentType, assessment percentage

assessmentID -> marksObtainable, assessmentTypeID, questionNumber , coID

marksdisseminationID -> studentID, assessmentID, marksObtained

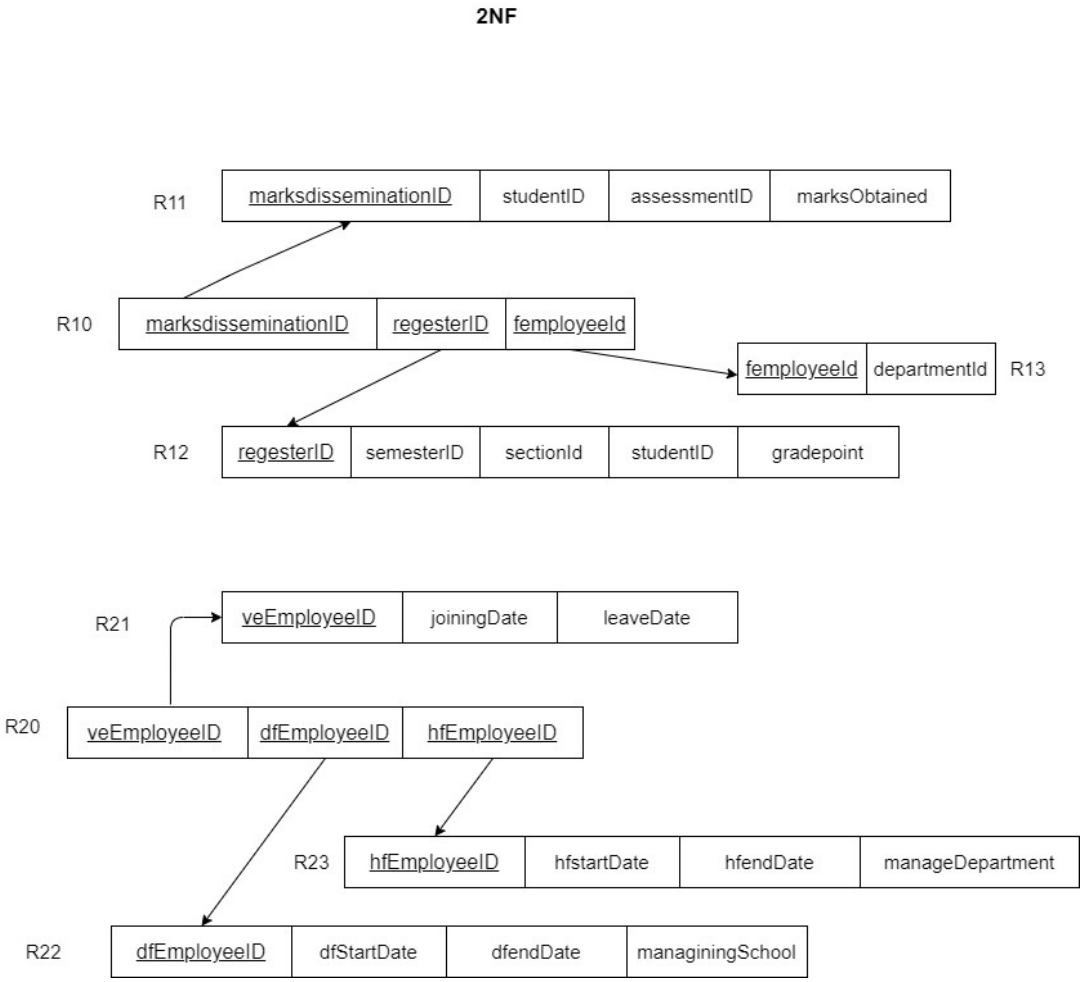
1NF

1NF

R1 (marksdisseminationID, regesterID,femployeeId,departmentId, studentID, assessmentID, marksObtained, semesterID, sectionId, gradepoint)

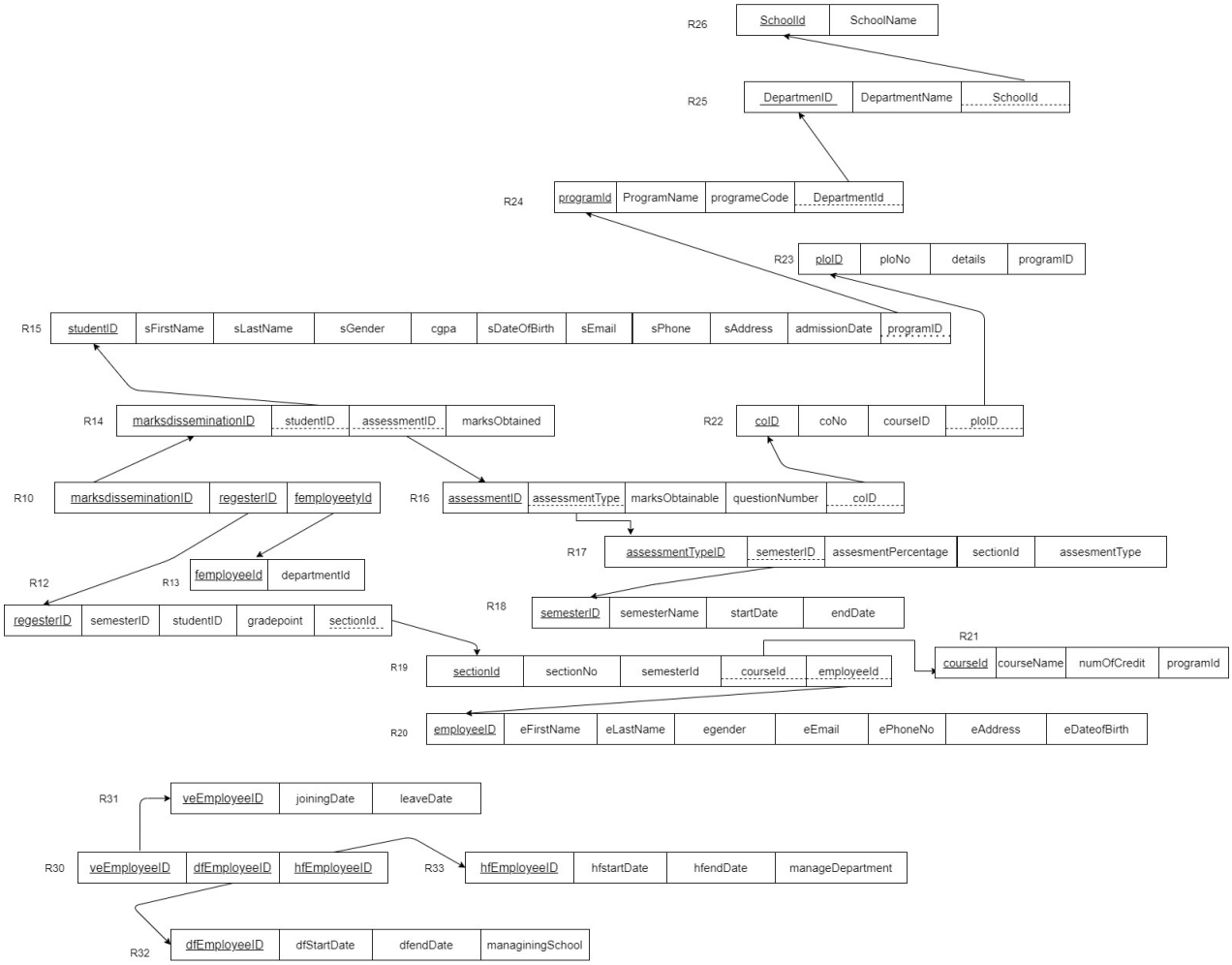
R2 (veEmployeeID , dfEmployeeID, hfEmployeeID, joiningDate, leaveDate, dfStartDate, dfendDate, managiningSchool, hfstartDate, hfendDate, manageDepartment)

2NF


2NF
R11
marksdisseminationID
studentID
assessmentID
marksObtained
R10
marksdisseminationID
regesterID
femployeeId
femployeeId
departmentId
R13
R12
regesterID
semesterID
sectionId
studentID
gradepoint
R21
veEmployeeID
joiningDate
leaveDate
R20
veEmployeeID
dfEmployeeID
hfEmployeeID
R23
hfEmployeeID
hfstartDate
hfendDate
manageDepartment
R22
dfEmployeeID
dfStartDate
dfendDate
managiningSchool

3NF

3 NF



BCNF:The primary key is not dependent on any other non primary key. so there is no BCNF.

Data Dictionary

Schools_T

Name	Data Type	Size	Remark
nschoolID	BIGINT	20	This is the Primary Key of School Example: “1”
cschoolName	VARCHAR	255	This is the name of the School. Example: “School of Engineering, Technology and Science”

Departments_T

Name	Data Type	Size	Remark
cdepartmentID	VARCHAR	255	This is the Primary Key of the Department. Example: “CSE”
cdepartmentName	VARCHAR	255	This is the name of the Department. Example: “Computer Science and Engineering”
nschool_id	BIGINT	20	This is the Foreign Key of the table School. Example: “1”

Programs_T

Name	Data Type	Size	Remark
cprogramID	VARCHAR	255	This is the Primary Key of the program. Example: “EEE”
cprogramCode	VARCHAR	255	This is the program name in short form for a Program Example:”B.Sc”.
cprogramName	VARCHAR	255	This is the name of the Degree Program. Example: “Bachelor of Science”
cdepartment_id	VARCHAR	255	This is the Foreign Key from the Department table. Example: “CSE”

Courses_T

Name	Data Type	Size	Remark
ccourseID	VARCHAR	255	This is the Primary Key for the Course. Example: “CSE203”

ccourseName	VARCHAR	255	This is the name of the Course. Example:" Data Structure"
nnoOfCredits	INTEGER	11	This is the credit for the Course. Example: "3"
cprogram_id	VARCHAR	255	This is the Foreign Key from Program table Example:" 1".

Employees_T

Name	Data Type	Size	Remark
nemplyeeID	BIGINT	20	This is the Primary Key for Faculty. Example: "1803"
cfname	VARCHAR	255	This is the first name of the Faculty. Example: " Sadita"
clname	VARCHAR	255	This is the last name of the Faculty Example: "Ahmed"
ddateOfBirth	DATE	DD-MM-Y	This the Date of Birth of the Faculty YYY Example: "01-01-1993"
cgender	VARCHAR	255	This is the gender of the Faculty. Example: "F"

cemail	VARCHAR	255	This is the email address of the Faculty. Example: “sadita@iub.edu.bd”
cphone	VARCHAR	255	This is the phone number of the Faculty. Example: “01292383111”
caddress	VARCHAR	255	This is the address of the Faculty. Example: “House 1, Road 1, Sector 1, Area, Dhaka, Bangladesh,

Faculties_T

Name	Data Type	Size	Remark
nFemplyeeID	BIGINT	20	This is the Primary Key for Faculty. Example: “1803”
cdepartment_id	BIGINT	20	This is the Foreign Key from the Department table. Example: “CSE”

vcs_T

Name	Data Type	Size	Remark
nvemployeeID	BIGINT	20	This is the Primary Key for employees. Example: “1803”
djoiningDate	DATE	YYYY-MM-DD	The date when the employee assigned in the

			job. Example “2012-12-05”
dleaveDate	DATE	YYYY-MM-DD	The date when the duration of assigned job end. Example “2016-12-05”

Deans_T

Name	Data Type	Size	Remark
nDFemployeeID	BIGINT	20	This is the Primary Key for Faculty. Example: “1803”
djoiningDate	DATE	YYYY-MM-DD	The date when the employee assigned in the job. Example “2012-12-05”
dleaveDate	DATE	YYYY-MM-DD	The date when the duration of assigned job end. Example “2016-12-05”
nmanagingSchool	BIGINT	20	This is the foreign Key from the Schools table Example: “1”

HeadOfDepts_T

Name	Data Type	Size	Remark
nHFemployeeID	BIGINT	20	This is the Primary Key for Faculty. Example: “1803”

djoiningDate	DATE	YYYY-MM-DD	The date when the employee assigned in the job. Example "2012-12-05"
dleaveDate	DATE	YYYY-MM-DD	The date when the duration of assigned job end. Example "2016-12-05"
nmanagingDepartment	BIGINT	20	This is the foreign Key from the departments table Example: "1"

Students_T

Name	Data Type	Size	Remark
cstudentID	BIGINT	20	This is the Primary Key for the Students. Example: "1800001"
cfname	VARCHAR	255	This is the first name of the Student. Example: "Shoban"
clname	VARCHAR	255	This is the last name of the Student. Example: "Bhowmik"
ddateOfBirth	DATE	DD-MM-YYYY	This the Date of Birth of the Student. Example: "1998-01-01"
cgender	VARCHAR	255	This is the gender of the Student. Example: "M"
cemail	VARCHAR	255	This is the email address of the Student. Example: "1850105@iub.edu.bd"
cphone	VARCHAR	255	This is the phone number of the Student.

			Example: “0191211141”
caddress	VARCHAR	255	This is the address of the Student. Example: “House 1, Road 1, Sector 1, Area, Dhaka, Bangladesh”
ncgpa	FLOAT	53	This is the cgpa of the Student. Example: 4.00
dadmissionDate	DATE	DD-MM-YYYY	This the Date of admission of the Student. Example: “2016-01-01”
cprogram_id	VARCHAR	255	This is the Foreign Key from Programs table Example:” B.Sc”.

Sections_T

Name	Data Type	Size	Remark
nsection_id	BIGINT	20	This is the Primary Key for Sections
nsectionNo	INTEGER	11	This is the section number. Example: “1”
ccourse_id	VARCHAR	255	This is the foreign key from the Courses table. Example: “CSE101”
nFemployee_id	BIGINT	20	This is the foreign key from Faculties table Example: “108051”
nsemester_id	BIGINT	20	This is the foreign key from semesters

			table Example: “1”

Registers_T

Name	Data Type	Size	Remark
nregisterID	BIGINT	20	This is the Primary Key for Registers
nSemisterID	BIGINT	20	This is the Foreign Key from semesters table
nsection_id	BIGINT	20	This is the Foreign Key from sections table
nstudent_id	BIGINT	20	This is the Foreign Key from students table
ngradePoint	FLOAT	53	This is the grade value for each course. Example “3.7”

Semesters_T

Name	Data Type	Size	Remark
nsemester_id	BIGINT	20	This is the Primary Key for semester
csemesterName	VARCHAR	255	This is the name of the semester. Example: “summer19”
dstartDate	DATE	YYYY-MM-DD	The date when the semester starts. Example “2012-12-05”

dendDate	DATE	YYYY-MM-DD	The date when the semester end. Example "2013-3-05"

Assessments_T

Name	Data Type	Size	Remark
nassessmentID	BIGINT	20	This is the Primary Key for Assessments
nco_id	BIGINT	20	This is the Foreign Key from cos table
cmarksObtainable	INTEGER	11	This is the question's mark. Example"30"
cquestionNumber	VARCHAR	255	This is the question number. Example "Q1"
nassessmentTypeID	BIGINT	20	This is the foreign key from assessmentTypes table

Plos_T

Name	Data Type	Size	Remark
nplo_id	BIGINT	20	This is the primary key for Program Learning Outcome. Example: "01"
nplo_no	INTEGER	11	This is the PLO NO for Program Learning Outcome. Example: "O1"

cdetails	VARCHAR22	255	This is the details of the Program Learning Outcome. Example: “An ability to select and apply the knowledge, techniques, skills, and modern
cprogram_id	VARCHAR	255	This is the foreign key from Programs table Example: ”B.Sc”.

Cos_T

Name	Data Type	Size	Remark
nco_id	BIGINT	20	This is the Primary Key cos Outcome. Example: ”1”
nco_no	INTEGER	11	This is the number of the Courses Outcome. Example: ”1”
ccourse_id	VARCHAR	255	This is the Foreign Key from the Courses table. Example: “CSE101”

MARKSDISSEMINATIONS_T

Name	Data Type	Size	Remark

nmarksdissemination_id	BIGINT	20	This is the Primary Key MarksDisseminations Outcome. Example: ”1”
nmarksObtained	INTEGER	11	This is the number of the marksObtained. Example: ”50”
nassessmentID	BIGINT	20	This is the Foreign Key from the assessment table. Example: “5”
nstudentID	BIGINT	20	This is the Foreign Key from the students2 table. Example: “1610044”

ASSESSMENTTYPES_T

Name	Data Type	Size	Remark
nassessmentTypeID	BIGINT	20	This is the Primary Key assessmentTypes Outcome. Example: ”1”
cassessmentType	VARCHAR	255	This is the assessment type of the assessment. Example: ”Quiz”, ”mid”
nassessmentPercentage	INTEGER	11	This is the number of percentage given an assessment Example: “50”
nsectionID	BIGINT	20	This is the Foreign Key from the sections table. Example: “8”

COMAPPING_T

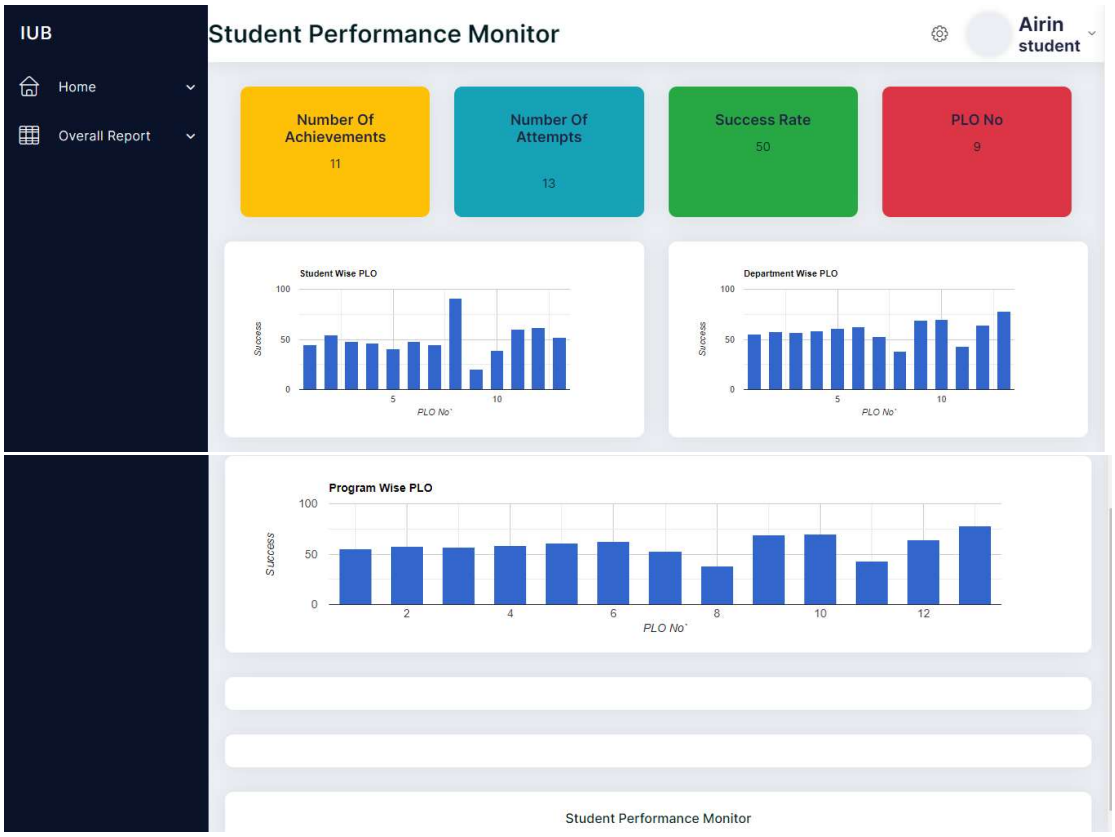
Name	Data Type	Size	Remark
ncoMappingID	INTEGER	20	This is the Primary Key of comapping Outcome. Example: ”1”
ncoID	BIGINT	20	This is the foreign key of cos_T Example: ”7”,
npoID	BIGINT	20	This is the foreign key of plos_T Example: ”7”,

CHAPTER 4

PHYSICAL SYSTEM DESIGN

- **INPUT FORMS**
- **OUTPUT FORMS**

Student Dashboard



Student Dashboard- we can see this is a student dashboard. When we first login to the student dashboard we see 4 cards which belong to the number of achievement. The number of attempts, success rate and lowest plo, navbar, and some charts like student or department or program wise plo. From the 'overall report' page here that student can view co and plo achievement for each of the courses taken so far. Here number of achievements and attempt means the success rate of how many plo is done by a student. We also notice that every row has its own plo percentage and there are 5 view tables in here. From the student view table we know the information which know helps s to make student dashboard. First we give a variable then we added the data table where number column (x and y axis) are connected. We create the graph by doing array type data which is coming from database. The sections files are connected by merging. Here are some joint command also which is helping us to create the variable. Then we use DB row in select command for laravel. Basically we use DB row for 'as' command. In laravel we should always use the 'get' command at the end because if we do not use it the command can't be watched in laravel. From views table plo we collected the database. In here we create student, department and

program plo. From student table we know there are so e program which is relatable to the department. By using dpth, we can find out from which department the program is belongs. First we doing normal query that we fetched the data which outcome came as a data dictionary then we converted it to the array then we send the array as json encode because browser can’t take array code that's why we converted the array code as a json code. We merge the data and option part and create this dashboard. In chart function we enter the data and data option.by Google chart we can easily do everything easily. We just used the id and ways to create this.

PHP variable represented by dollar sign. Like $ c its means courses. In course plo percentage we see how many courses are taken by one student and its plo and its percentages. Basically we login by using user id and fetch the similar id.by doing this the courses are dynamically shown their work. We also loop in here by using aegx adding the functions, in first array we input the data .in every course we need different unique id for this reason we used button id. In the graph chart sp1 means student co and sp2 means student plo. Here we input the array on front end database. We also create a PHP code because at first we emptied the query because when the array first run it will never emptied in second time. Than we inputted the data and draw the graph.

Code:

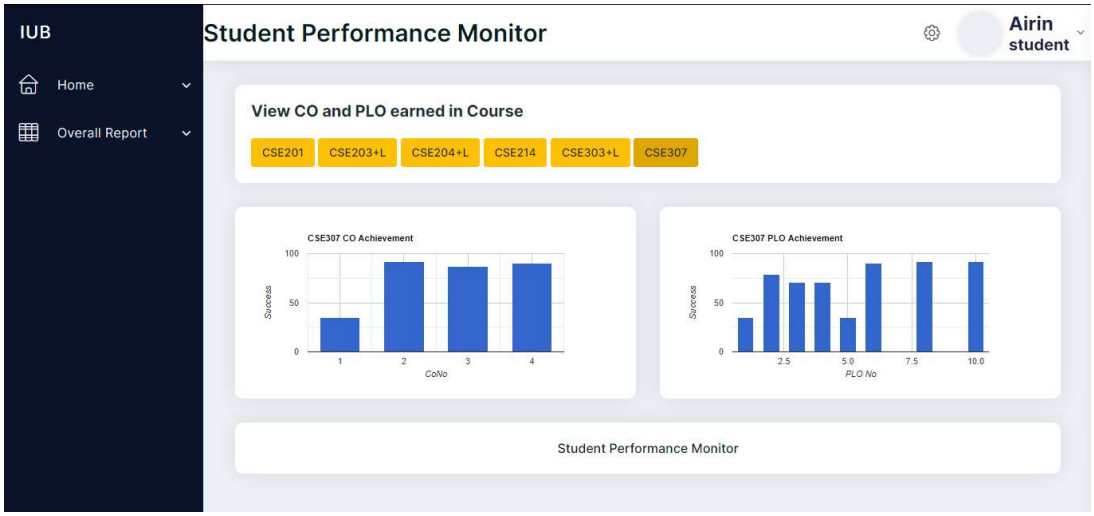
```
$student = Session::get('student');
    $cards = DB::table('view_student_plo')->where('studentID', $student->studentID)->get();
    $stdplo = DB::table('view_student_plo')
        ->join('plos', 'view_student_plo.ploid', '=', 'plos.ploid')
        ->select(
            DB::raw('plos.ploNo'),
            DB::raw('view_student_plo.plo_percentage as success')
        )
        ->where('studentID', $student->studentID)
```

```
            ->get();
        $dpth = DB::table('programs')
            ->join('students', 'students.programID', '=',
'programs.programID')
            ->select('departmentID')
            ->where('students.programID', $student->programID)
            ->limit(1)
            ->first();


        $pth = DB::table('programs')
            ->join('students', 'students.programID', '=',
'programs.programID')
            ->select('programs.programID')
            ->where('students.programID', $student->programID)
            ->limit(1)
            ->first();
        $dptplo = DB::table('view_student_plo')
            ->join('plos', 'view_student_plo.ploID', '=', 'plos.ploID')
            ->join('programs', 'plos.programID', '=', 'programs.programID')
            ->select(
                DB::raw('plos.ploNo'),
                DB::raw('AVG(view_student_plo.plo_percentage) as
success')
            )
            ->where('programs.departmentID', $dpth->departmentID)
            ->groupBy('plos.ploNo')
            ->get();
        $ptplo = DB::table('view_student_plo')
```

```
            ->join('plos', 'view_student_plo.ploID', '=', 'plos.ploID')
            ->join('programs', 'plos.programID', '=', 'programs.programID')
            ->select(
                DB::raw('plos.ploNo'),
                DB::raw('AVG(view_student_plo.plo_percentage) as
success')
            )
            ->where('programs.programID', $pth->programID)
            ->groupBy('plos.ploNo')
            ->get();
        $arrp = array();
        $arrs = array();
        $arrd = array();
        foreach ($stdplo as $s)
            array_push($arrs, [$s->ploNo, (float)$s->success]);
        foreach ($dptplo as $s)
            array_push($arrd, [$s->ploNo, (float)$s->success]);
        foreach ($ptplo as $s)
            array_push($arrp, [$s->ploNo, (float)$s->success]);
        $Lowest = DB::table('view_student_plo')->where('studentID',
$student->studentID)->where('plo_percentage', $cards-
>min('plo_percentage'))->first();
        return view('pages/student/studentdashboard')->with(
            [
                'student' => $student,
                'username' => $student->firstname,
                'userType' => 'student',
                'cards' => $cards,
```

```
            'lowest' => $Lowest,
            'stdplo' => json_encode($arrs),
            'dptplo' => json_encode($arrd),
            'ptplo' => json_encode($arrp)
        ]
    );
```

Overall Report Page:



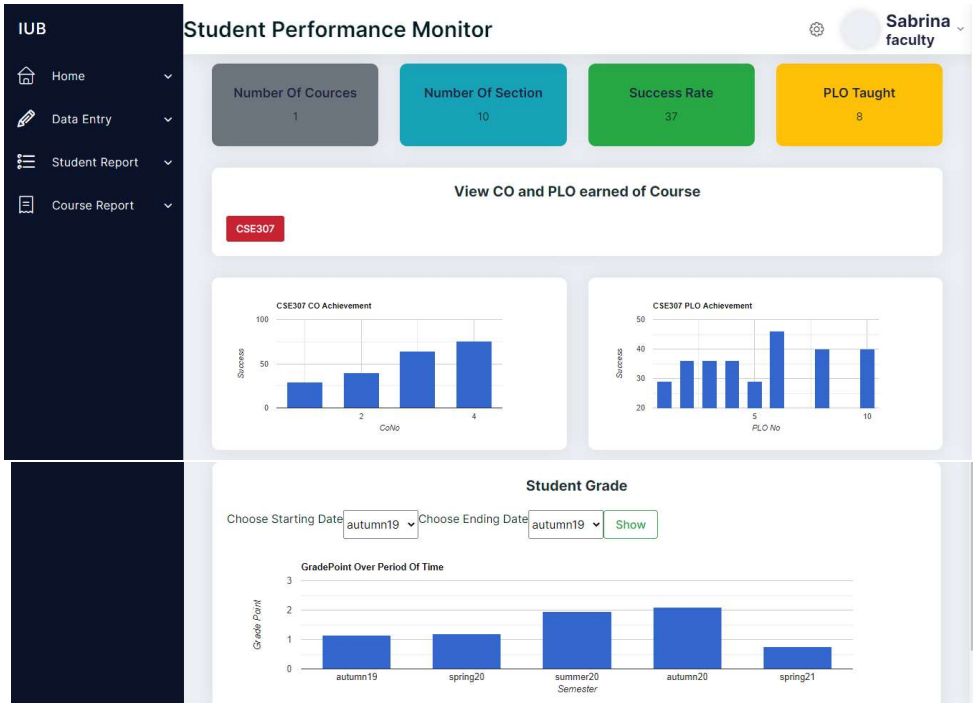
This page shows the number of courses taken by a student in a particular semester. This page will be view when an user click on the overall report tab shown the left navigation bar. Here a student name Airin took CE201,CSE203+L,CSE204+L,CSE214,CSE303+L and CSE307 on a particular semester. If a user wants to view courses achievements. Click on the tab of the specific course and two column chart will appear. For example: if the user click CSE307 the left column chart which label as CO Achievement shows the number of co achieved or successfully gain by the student on that particular course and the right side column chart which label as PLO Achievement shows the number of co achieved or successfully gain by the student on that particular course.

Code: $student = Session::get('student');

```
$plos = DB::table('plos')
    ->join('programs', 'programs.programID', '=', 'plos.programID')
    ->join('students', 'students.programID', '=', 'plos.programID')
    ->select('ploID')
    ->where('studentID', $student->studentID)
    ->where('programs.programID', $student->programID)
    ->get();
$success = DB::table('course_plo_percentage')
    ->select('courseID', 'ploID', 'success')
    ->where('studentID', $student->studentID)
    ->get();
$courses = DB::table('course_plo_percentage')
    ->select(
        'courseID',
        DB::raw('CASE
        WHEN courseID="CSE203+L" THEN 1
        WHEN courseID="CSE204+L" THEN 2
        WHEN courseID="CSE309" THEN 3
        WHEN courseID="CSE307" THEN 4
        WHEN courseID="CSE210+L" THEN 5
        WHEN courseID="CSE214" THEN 6
        WHEN courseID="CSE201" THEN 7
        WHEN courseID="CSE216+L" THEN 8
        WHEN courseID="CSE303+L" THEN 9
        WHEN courseID="ACN201" THEN 10
        WHEN courseID="ACN202" THEN 11
        WHEN courseID="ACN301" THEN 12
```

```
            WHEN courseID="ACN305" THEN 13
            WHEN courseID="MIS340" THEN 14
            WHEN courseID="MIS341" THEN 15
            WHEN courseID="ACN302" THEN 16
            ELSE 0
            END AS id')
        )
        ->where('studentID', $student->studentID)
        ->groupBy('courseID')
        ->get();
    $p1 = DB::table('view_student_co') //student co
        ->join('cos', 'view_student_co.coID', '=', 'cos.coID')
        ->select('studentID', 'courseID', 'cos.coNo', 'co_percentage')
        ->where('studentID', $student->studentID)
        ->get();
    $p2 = DB::table('course_plo_percentage') //student plo
        ->where('studentID', $student->studentID)
        ->join('plos', 'plos.ploID', '=', 'course_plo_percentage.ploID')
        ->get();
```

Faculty Dashboard:



Faculty dashboard- If a user is faculty. The user need to login from the login page as a faculty to view this page. This page is the following home page from the left navigation bar. The page contain 4 cards at the top of the page. The cards are following:

1) Number Of Courses – the card number of courses shows the number of courses that are taken by the faculty user.

2) Number of Section – the card number of section shows the number of section that are taken by the faculty user.

3) Success rate – the card success rate shows the success rate of students plo achievement average under his or her sections.

4)PLO Taught – the card PLO taught shows the number of different plo taught by the faculty.

Under the cards a section name View CO and PLO earned of Courses. This shows the number of courses tab. Here the user only teaches CSE307 so the CSE307 clickable tab will appear. If a user wants to view the students performance of a particular course. User can click tab of the course and three new column charts will appear. Chart label by Co achievement shows the no of the cos achieved or success rate of all the student on his or her course. Furthermore, Chart lable by PLO achievement shows the no of the plo achieved or

success rate of all the student on his or her course. And at bottom section of the page name Students grade shows grade point

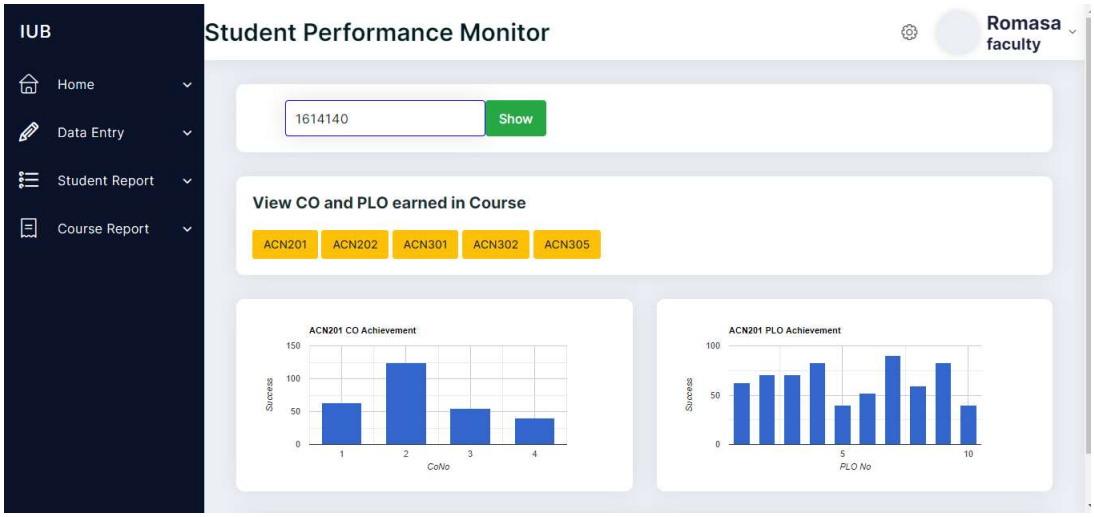
Code:

```
$faculty = Session::get('faculty');
        $startDate = NULL;
        $endtDate = NULL;
        $garray = array();
        if (!empty(Session::get('startDate'))) {
            $startDate = Session::get('startDate');
            $endDate = Session::get('endDate');
            $data =
                DB::table('v_transcript')
                ->join('sections', 'sections.sectionID', '=',
'v_transcript.sectionID')
                ->join('semesters', 'semesters.semesterID', '=',
'sections.semesterID')
                ->select('semesterName', DB::raw('AVG(gradePoint) as
gp'))
                ->where('FemployeeID', $faculty->employeeID)
                ->whereRaw('semesters.semesterID >= ? AND
semesters.semesterID <= ?', [$startDate, $endDate])
                ->groupBy('courseID', 'startDate',
'semesters.semesterName')
                ->get();
            foreach ($data as $d)
                array_push($garray, [$d->semesterName, (float)$d->gp]);
        }
        $card1 = DB::table('faculty_plo')
            ->select(
```

```
            DB::raw('DISTINCT courseID'),
            DB::raw('CASE
            WHEN courseID="CSE203+L" THEN 1
            WHEN courseID="CSE204+L" THEN 2
            WHEN courseID="CSE309" THEN 3
            WHEN courseID="CSE307" THEN 4
            WHEN courseID="CSE210+L" THEN 5
            WHEN courseID="CSE214" THEN 6
            WHEN courseID="CSE201" THEN 7
            WHEN courseID="CSE216+L" THEN 8
            WHEN courseID="CSE303+L" THEN 9
            WHEN courseID="ACN201" THEN 10
            WHEN courseID="ACN202" THEN 11
            WHEN courseID="ACN301" THEN 12
            WHEN courseID="ACN305" THEN 13
            WHEN courseID="MIS340" THEN 14
            WHEN courseID="MIS341" THEN 15
            WHEN courseID="ACN302" THEN 16
            ELSE 0
            END AS id')
        )
        ->where('FemployeeID', $faculty->employeeID)
        ->get();
    $card2 = DB::table('faculty_plo')
        ->select(DB::raw('DISTINCT sectionID'))
        ->where('FemployeeID', $faculty->employeeID)
        ->get();
```

```
        $card3 = DB::table('faculty_plo')
            ->where('FemployeeID', $faculty->employeeID)
            ->get();
        $card4 = DB::table('faculty_plo')
            ->select(DB::raw('DISTINCT ploID'))
            ->where('FemployeeID', $faculty->employeeID)
            ->get();
        $p1 = DB::table('faculty_co')
            ->select('faculty_co.courseID', 'coNo',
DB::raw('AVG(co_percentage) as success'))
            ->join('cos', 'cos.coID', '=', 'faculty_co.coID')
            ->where('FemployeeID', $faculty->employeeID)
            ->groupBy('faculty_co.courseID', 'faculty_co.coID', 'coNo')
            ->get();
        $p2 = DB::table('faculty_plo')
            ->select('faculty_plo.courseID', 'ploNo',
DB::raw('AVG(co_percentage) as success'))
            ->join('plos', 'plos.ploID', '=', 'faculty_plo.ploID')
            ->where('FemployeeID', $faculty->employeeID)
            ->groupBy('faculty_plo.courseID', 'faculty_plo.ploID', 'ploNo')
            ->get();
        $arr = array()
```

Student Report Page:



Student Report page- If a user click on student report from the left navigation bar. A view page will appear name student report page. At the top section of the page a input form will appear with a show button on the right. If a faculty user wants to view a specific student achievement on other courses that are taken by the student. On the input form id of the student needs to be typed and input into the form and click show. Two new section will appear if the id is valid or a invalid message will be shown if not. Two new section with first name View CO and PLO earned of Courses. This shows the number of courses tab. If a user wants to view the students performance of a particular course. Click on the clickable tab of the course. And one new section will appear with two new column chart label by Co achievement shows the no of the cos achieved or success rate of all the student on his or her course.

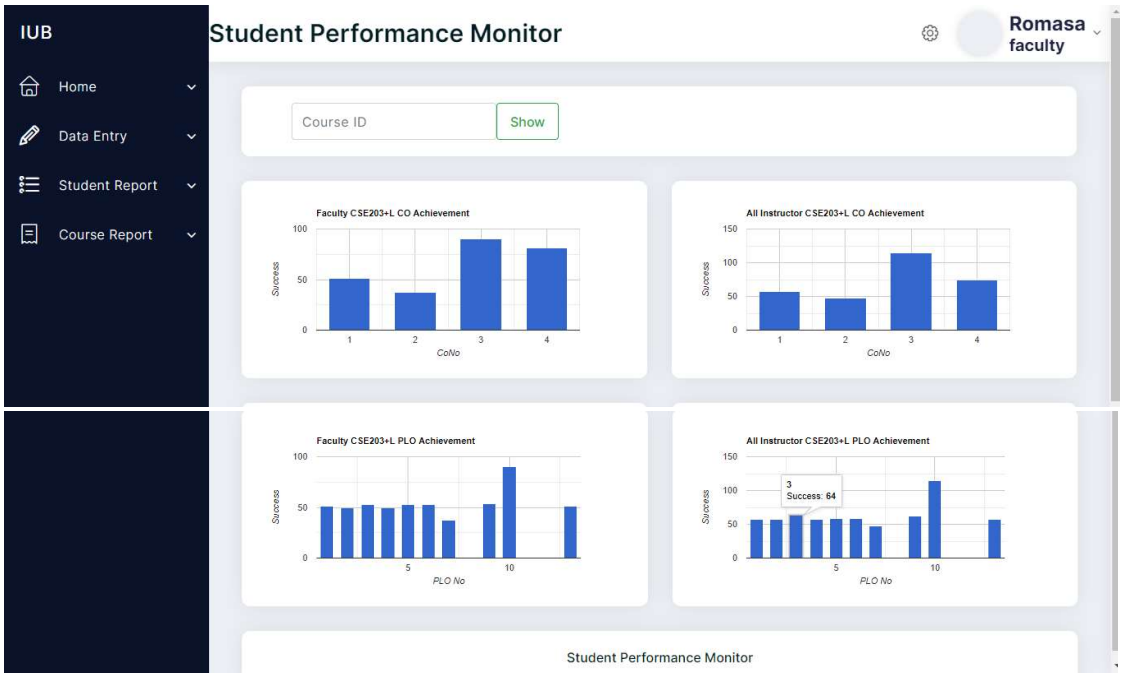
Code:

```
$cID = NULL;
        $arr = array();
        $p1 = NULL;
        $p2 = NULL;
        $courses = NULL;
        if (!empty(Session::get('cID'))) {
            $cID = Session::get('cID');
            $courses = DB::table('course_plo_percentage')
```

```
    ->select(
        'courseID',
        DB::raw('CASE
    WHEN courseID="CSE203+L" THEN 1
    WHEN courseID="CSE204+L" THEN 2
    WHEN courseID="CSE309" THEN 3
    WHEN courseID="CSE307" THEN 4
    WHEN courseID="CSE210+L" THEN 5
    WHEN courseID="CSE214" THEN 6
    WHEN courseID="CSE201" THEN 7
    WHEN courseID="CSE216+L" THEN 8
    WHEN courseID="CSE303+L" THEN 9
    WHEN courseID="ACN201" THEN 10
    WHEN courseID="ACN202" THEN 11
    WHEN courseID="ACN301" THEN 12
    WHEN courseID="ACN305" THEN 13
    WHEN courseID="MIS340" THEN 14
    WHEN courseID="MIS341" THEN 15
    WHEN courseID="ACN302" THEN 16
    ELSE 0
    END AS id')
    )
    ->where('studentID', $cID)
    ->groupBy('courseID')
    ->get();
$p1 = DB::table('view_student_co')
    ->join('cos', 'view_student_co.coID', '=', 'cos.coID')
```

```
            ->select('studentID', 'courseID', 'cos.coNo', 'co_percentage')
            ->where('studentID', $cID)
            ->get();
        $p2 = DB::table('course_plo_percentage')
            ->where('studentID', $cID)
            ->join('plos', 'plos.ploID', '=', 'course_plo_percentage.ploID')
            ->get();
    }
```

Course Report Page:



On this page, we see a search option whereby searching a course ID the faculty user will see the success of his own "PLO" & "CO" as well as the success of that course Moreover, the user can see the success of "PLO" & "CO" of other courses.

Code:

```
$faculty = Session::get('faculty');
    $cID = NULL;
    $c1 = NULL;
```

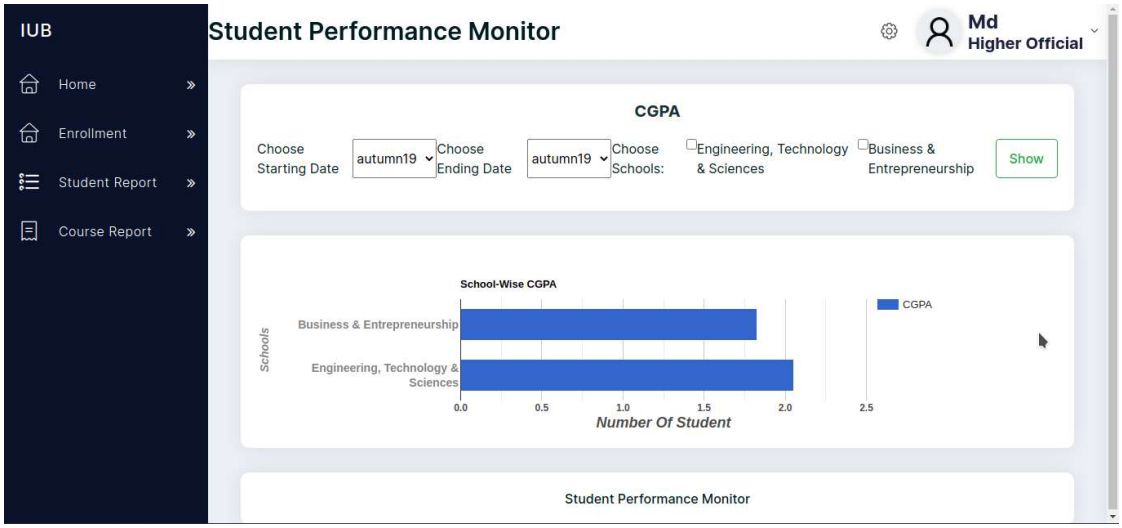
```
        $c2 = NULL;
        $c3 = NULL;
        $c4 = NULL;
        $arr = array();
        if (!empty(Session::get('cID'))) {

            $cID = Session::get('cID');
            if (DB::table('faculty_co')
                ->select('faculty_co.courseID', 'coNo',
DB::raw('AVG(co_percentage) as success'))
                ->join('cos', 'cos.coID', '=', 'faculty_co.coID')
                ->where('faculty_co.courseID', $cID)
                ->where('faculty_co.FemployeeID', $faculty->employeeID)
                ->groupBy('faculty_co.courseID', 'faculty_co.coID', 'coNo')
                ->exists()
            ) {
                $c1 = DB::table('faculty_co')
                    ->select('faculty_co.courseID', 'coNo',
DB::raw('AVG(co_percentage) as success'))
                    ->join('cos', 'cos.coID', '=', 'faculty_co.coID')
                    ->where('faculty_co.courseID', $cID)
                    ->where('faculty_co.FemployeeID', $faculty-
>employeeID)
                    ->groupBy('faculty_co.courseID', 'faculty_co.coID',
'coNo')
                    ->get();
            }
            $c2 = DB::table('faculty_co')
```

```
            ->select('faculty_co.courseID', 'coNo',
DB::raw('AVG(co_percentage) as success'))
            ->join('cos', 'cos.coID', '=', 'faculty_co.coID')
            ->where('faculty_co.courseID', $cID)
            ->groupBy('faculty_co.courseID', 'faculty_co.coID', 'coNo')
            ->get();
        if (DB::table('faculty_plo')
            ->select('faculty_plo.courseID', 'ploNo',
DB::raw('AVG(co_percentage) as success'))
            ->join('plos', 'plos.ploID', '=', 'faculty_plo.ploID')
            ->where('faculty_plo.courseID', $cID)
            ->where('FemployeeID', $faculty->employeeID)
            ->groupBy('faculty_plo.courseID', 'faculty_plo.ploID',
'ploNo')
            ->exists()
        ) {
            $c3 = DB::table('faculty_plo')
                ->select('faculty_plo.courseID', 'ploNo',
DB::raw('AVG(co_percentage) as success'))
                ->join('plos', 'plos.ploID', '=', 'faculty_plo.ploID')
                ->where('faculty_plo.courseID', $cID)
                ->where('FemployeeID', $faculty->employeeID)
                ->groupBy('faculty_plo.courseID', 'faculty_plo.ploID',
'ploNo')
                ->get();
        }
        $c4 = DB::table('faculty_plo')
            ->select('faculty_plo.courseID', 'ploNo',
DB::raw('AVG(co_percentage) as success'))
```

```
            ->join('plos', 'plos.ploID', '=', 'faculty_plo.ploID')
            ->where('faculty_plo.courseID', $cID)
            ->groupBy('faculty_plo.courseID', 'faculty_plo.ploID',
'ploNo')
            ->get();
    }
```

Higher Official Dashboard

School Wise Students CGPA



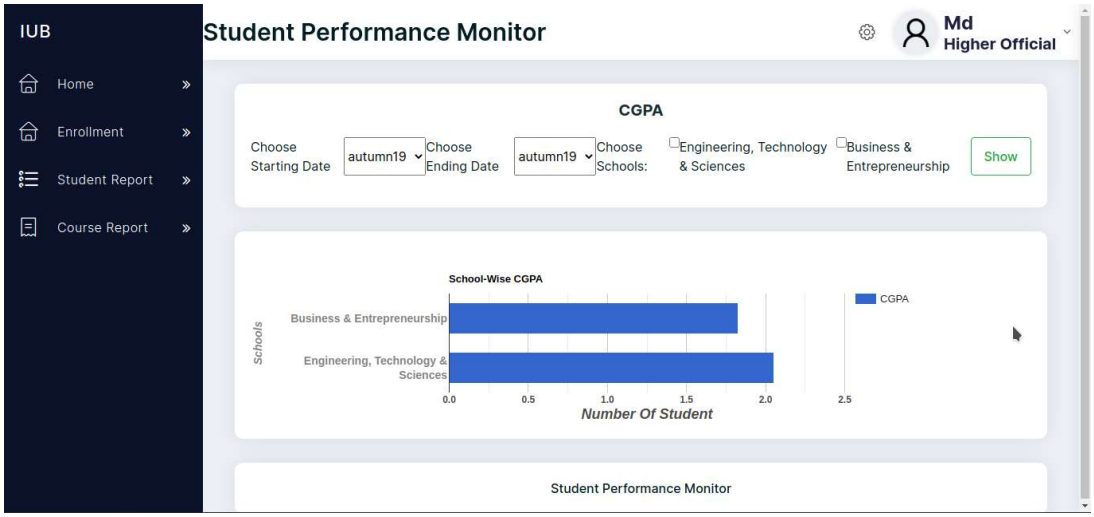
On this page we see that there are three buttons, one says "Choose start date", the next one has "Choose end date" and the other has the "Choose School" option. If the user wants, he can choose the options of his choice and the user will see the CGPA of the students in the school of a certain period or semester and the user will see the CGPA of the students in any semester and any school.

Code:

```
DB::statement("DROP VIEW IF EXISTS temps;");
        DB::statement("CREATE VIEW temps AS
        SELECT p.departmentID,s.programID,AVG(VVVT.cgpa)
Acgpa
        FROM(SELECT studentID,SUM(gpa)/COUNT(gpa) cgpa
```

```
        FROM(SELECT
studentID,semesterID,(TgradeWeight/Tcredit) gpa
        FROM(SELECT
vt.studentID,s.semesterID,SUM(vt.gradePoint*noOfCredit)
TgradeWeight, SUM(noOfCredit) Tcredit
        FROM v_transcript vt,sections s, registers r, courses c
        WHERE vt.sectionID=s.sectionID AND
r.studentID=vt.studentID AND s.courseID=c.courseID AND
        s.semesterID>=$startDate AND s.semesterID<=$endDate
        GROUP BY vt.studentID,s.semesterID ) VT) VVT
        GROUP BY studentID)VVVT, students s,programs p
        WHERE s.studentID=VVVT.studentID AND
p.programID=s.programID
        GROUP BY p.departmentID,s.programID;");


        $data = DB::table('temps')
            ->join('departments', 'departments.departmentID', '=',
'temps.departmentID')
            ->join('schools', 'departments.schoolID', '=',
'schools.schoolID')
            ->select('schoolName', DB::raw('AVG(Acgpa) AS Acgpa'))
            ->whereIn('schools.schoolID', $school)
            ->groupBy('schools.schoolName')
            ->get();
```

School Wise Students CGPA



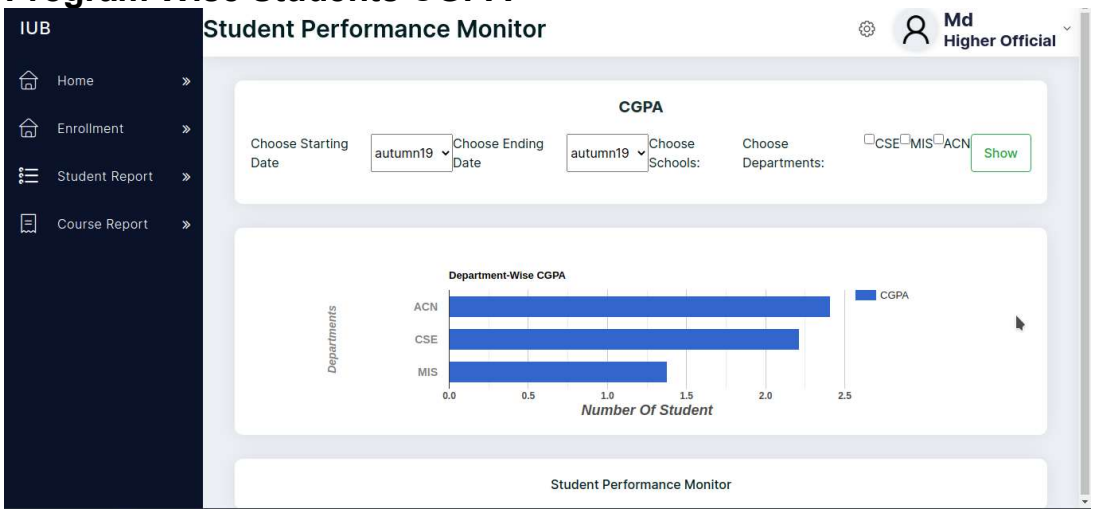
On this page we see that there are three buttons, one says "Choose start date", the next one has "Choose end date" and the other has the "Choose department" option. If the user wants, he can choose the options of his choice and the user will see the CGPA of the students in the department of a certain period or semester and the user will see the CGPA of the students in any semester and any department.

Code:

```
DB::statement("DROP VIEW IF EXISTS temps;");
        DB::statement("CREATE VIEW temps AS
        SELECT p.departmentID,s.programID,AVG(VVVT.cgpa) Acgpa
        FROM(SELECT studentID,SUM(gpa)/COUNT(gpa) cgpa
        FROM(SELECT studentID,semesterID,(TgradeWeight/Tcredit) gpa
        FROM(SELECT vt.studentID,s.semesterID,SUM(vt.gradePoint*noOfCredit) TgradeWeight, SUM(noOfCredit) Tcredit
        FROM v_transcript vt,sections s, registers r, courses c
        WHERE vt.sectionID=s.sectionID AND r.studentID=vt.studentID AND s.courseID=c.courseID AND
        s.semesterID>=$startDate AND s.semesterID<=$endDate
        GROUP BY vt.studentID,s.semesterID ) VT) VVT
```

```
        GROUP BY studentID)VVVT, students s,programs p
        WHERE s.studentID=VVVT.studentID AND
p.programID=s.programID
        GROUP BY p.departmentID,s.programID;");


        $data = DB::table('temps')
           ->select('departmentID','Acgpa')
           ->whereIn('departmentID', $department)
           ->get();
```

Program Wise Students CGPA



On this page we see that there are three buttons, one says "Choose start date", the next one has "Choose end date" and the other has the "Choose program" option. If the user wants, he can choose the options of his choice and the user will see the CGPA of the students in the program of a certain period or semester, and the user will see the CGPA of the students in any semester and any program.

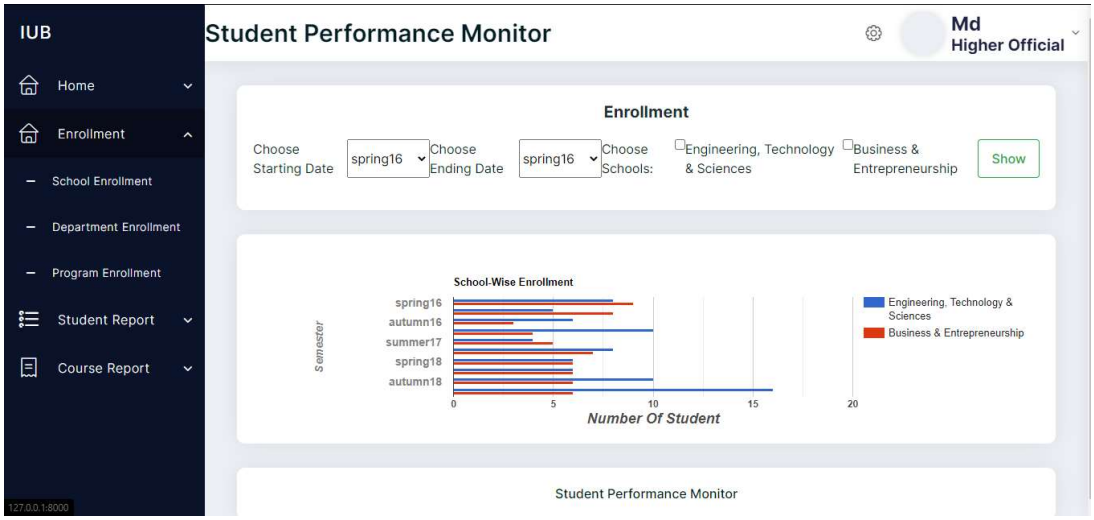
Code:

```
DB::statement("DROP VIEW IF EXISTS temps;");
        DB::statement("CREATE VIEW temps AS
        SELECT p.departmentID,s.programID,AVG(VVVT.cgpa)
Acgpa
```

```
        FROM(SELECT studentID,SUM(gpa)/COUNT(gpa) cgpa
        FROM(SELECT
studentID,semesterID,(TgradeWeight/Tcredit) gpa
        FROM(SELECT
vt.studentID,s.semesterID,SUM(vt.gradePoint*noOfCredit)
TgradeWeight, SUM(noOfCredit) Tcredit
        FROM v_transcript vt,sections s, registers r, courses c
        WHERE vt.sectionID=s.sectionID AND
r.studentID=vt.studentID AND s.courseID=c.courseID AND
        s.semesterID>=$startDate AND s.semesterID<=$endDate
        GROUP BY vt.studentID,s.semesterID ) VT) VVT
        GROUP BY studentID)VVVT, students s,programs p
        WHERE s.studentID=VVVT.studentID AND
p.programID=s.programID
        GROUP BY p.departmentID,s.programID;");


        $data = DB::table('temps')
            ->select('programID','Acgpa')
            ->whereIn('programID', $program)
            ->get();
```

Message #screen-shots

School Wise Student Enrollment Page



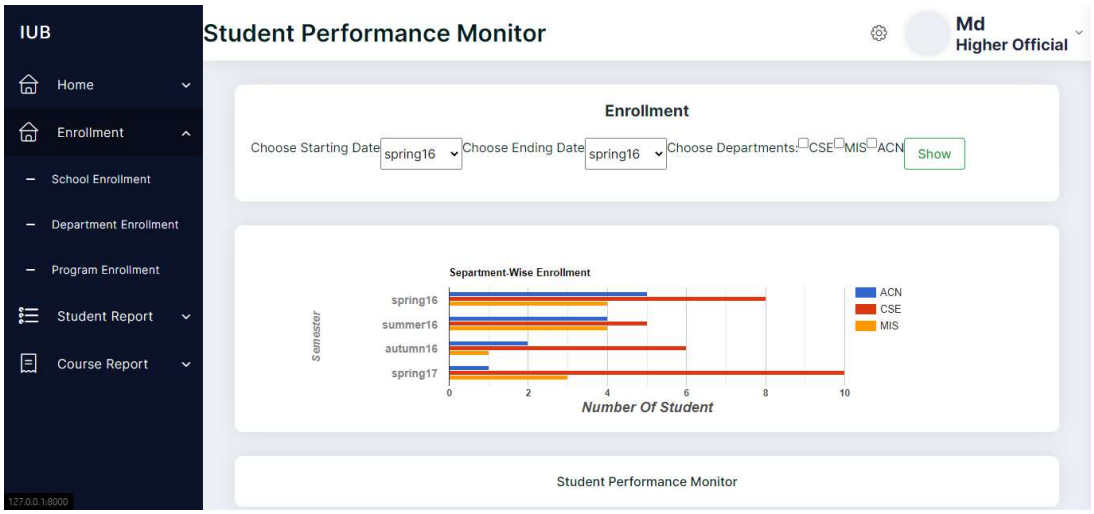
On this page we see that there are three buttons, one says "Choose start date", the next one has "Choose end date" and the other has the "Choose School" option. If the user wants, he can choose the options of his choice and see how many students have been enrolled in a school of a certain period or semester. You will also see how many students have been enrolled in any semester and in any School.

Code:

```
$startDate = Session::get('startDate');
        $endDate = Session::get('endDate');
        $school = Session::get('school');
        $data = DB::table('enrollment')
            ->join('semesters', 'semesters.startDate', '=', 'enrollment.admissionDate')
            ->select('schoolName', 'semesterName', DB::raw('SUM(amount) amount'))
            ->whereIn('schoolID', $school)
            ->where('semesterID', '>=', $startDate)
            ->where('semesterID', '<=', $endDate)
            ->groupBy('schoolName', 'semesterName')
            ->get();
```

```
$semesters = DB::table('semesters')
    ->select('semesterName')
    ->where('semesterID', '>=', $startDate)
    ->where('semesterID', '<=', $endDate)
    ->get();
```

Department Wise Student Enrollment Page



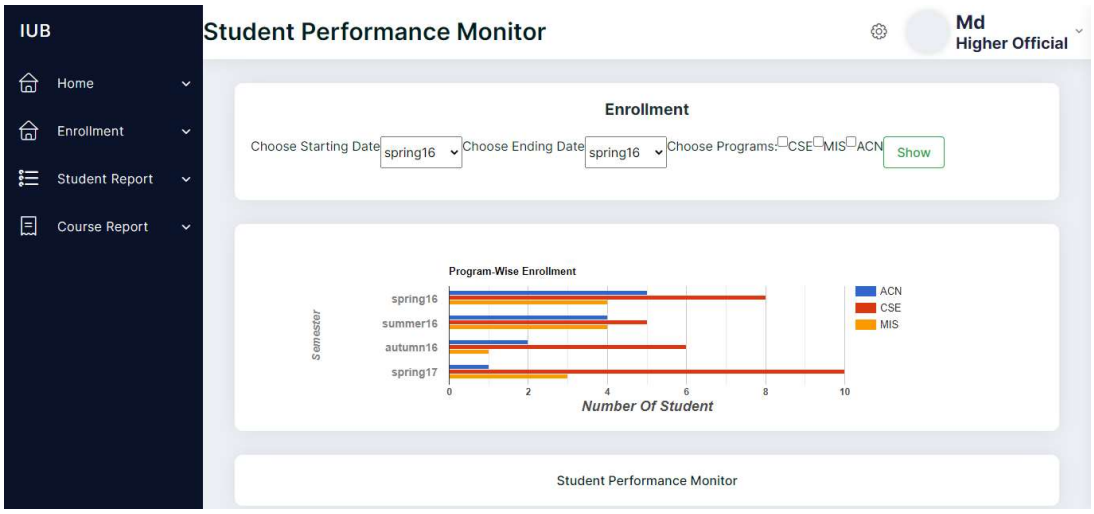
On this page we see that there are three buttons, one says "Choose start date", the next one has "Choose end date" and the other has the "Choose department" option. If the user wants, he can choose the options of his choice and see how many students have been enrolled in a department of a certain period or semester. You will also see how many students have been enrolled in any semester and in any department.

Code:

```
$startDate = Session::get('startDate');
    $endDate = Session::get('endDate');
    $department = Session::get('department');
    $data = DB::table('enrollment')
        ->join('semesters', 'semesters.startDate', '=', 'enrollment.admissionDate')
        ->select('departmentID', 'semesterName', DB::raw('SUM(amount) amount'))
```

```
->whereIn('departmentID', $department)
->where('semesterID', '>=', $startDate)
->where('semesterID', '<=', $endDate)
->groupBy('departmentID', 'semesterName')
->get();
```

Program Wise Student Enrollment Page



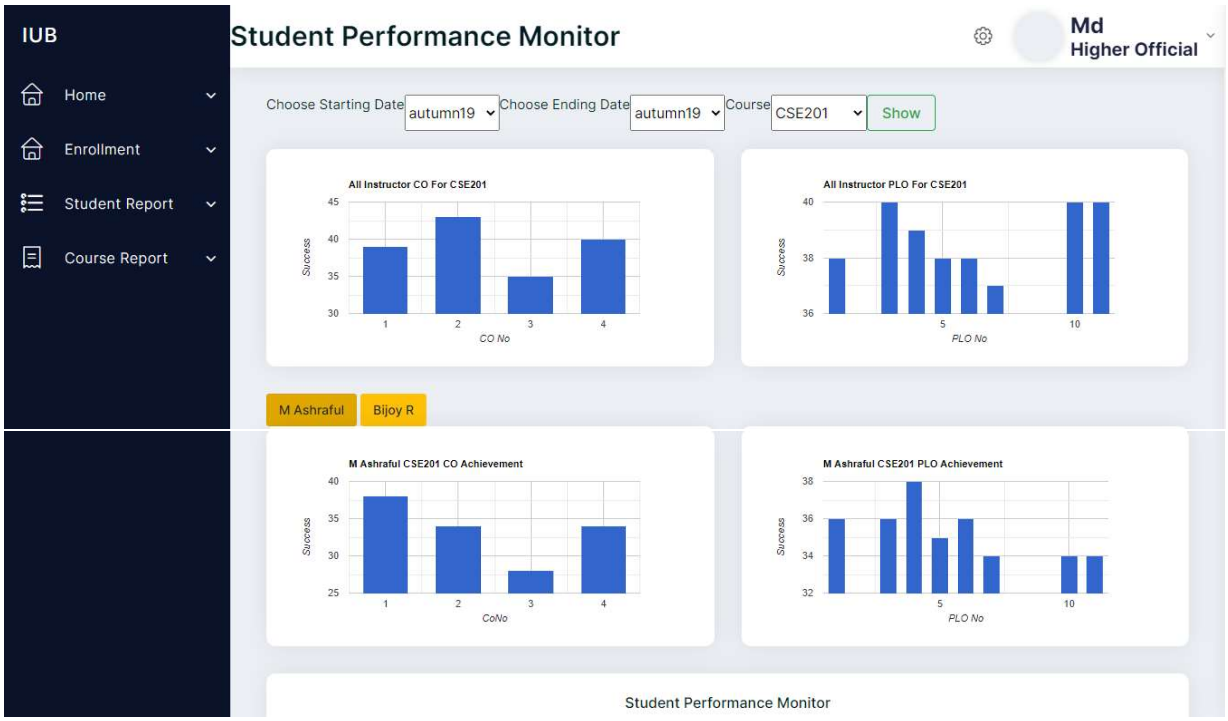
On this page we see that there are three buttons, one says "Choose start date", the next one has "Choose end date" and the other has the "Choose program" option. If the user wants, he can choose the options of his choice and see how many students have been enrolled in a program of a certain period or semester. You will also see how many students have been enrolled in any semester and in any program.

Code:

```
$higherO = Session::get('higherO');
    $startDate = NULL;
    $endtDate = NULL;
    $program = NULL;
    $arr = array();
    if (!empty(Session::get('startDate'))) {
        $startDate = Session::get('startDate');
```

```
        $endDate = Session::get('endDate');
        $program = Session::get('program');
        $data = DB::table('enrollment')
            ->join('semesters', 'semesters.startDate', '=',
'enrollment.admissionDate')
            ->select('programID', 'semesterName',
DB::raw('SUM(amount) amount'))
            ->whereIn('programID', $program)
            ->where('semesterID', '>=', $startDate)
            ->where('semesterID', '<=', $endDate)
            ->groupBy('programID', 'semesterName')
            ->get();

        $semesters = DB::table('semesters')
            ->select('semesterName')
            ->where('semesterID', '>=', $startDate)
            ->where('semesterID', '<=', $endDate)
            ->get();
```

Course Report Page



On this page we can see that there are three buttons, one says "Choose start date", the next one has "Choose end date" and the other has the "Choose course" option. If the user wants, he can choose the options of his choice and see the instructor wised "CO" and "PLO" achievement of any one time or semester course.

Code:

```
$startDate = Session::get('startDate');
        $endDate = Session::get('endDate');
        $courseID = (string)Session::get('courseID');

        DB::statement("DROP VIEW IF EXISTS temp1");
        DB::statement("DROP VIEW IF EXISTS temp2");

        DB::statement("CREATE VIEW temp1 AS
        SELECT FemployeeID,courseID,coNo,AVG(mo/mg*100) co_percentage
```

```
        FROM (SELECT
s.studentID,FemployeeID,c.courseID,c.coNo,SUM(marksObtained)
mo,SUM(marksObtainable) mg

        from students s, marksDisseminations m, assessments a,cos
c,comappings cm,plos p,assessmentTypes ta,sections st

        WHERE s.studentID=m.studentID AND
a.assessmentID=m.assessmentID AND a.coID=c.coID AND
ta.assessmentTypeID=a.assessmentTypeID

        AND st.sectionID=ta.sectionID AND p.ploID=cm.ploID AND
c.coID = cm.coID

        AND (c.courseID='$courseID' AND
st.semesterID>=$startDate AND st.semesterID<=$endDate)

        GROUP by
FemployeeID,s.studentID,c.courseID,st.sectionID,c.coID,c.coNo) VT

        GROUP By FemployeeID,courseID,coNo;");

        DB::statement("CREATE VIEW temp2 AS

        SELECT FemployeeID,courseID,ploNo,AVG(mo/mg*100)
plo_percentage

        FROM(SELECT
s.studentID,FemployeeID,c.courseID,st.sectionID,p.ploNo,SUM(mark
sObtained) mo,SUM(marksObtainable) mg

        from students s, marksDisseminations m, assessments a,cos
c,comappings cm,plos p,assessmentTypes ta,sections st

        WHERE s.studentID=m.studentID AND
a.assessmentID=m.assessmentID AND a.coID=c.coID AND
ta.assessmentTypeID=a.assessmentTypeID

        AND st.sectionID=ta.sectionID AND p.ploID=cm.ploID AND
c.coID = cm.coID AND

        (c.courseID='$courseID' AND st.semesterID>=$startDate
AND st.semesterID<=$endDate)

        GROUP by
FemployeeID,s.studentID,c.courseID,st.sectionID,p.ploID,p.ploNo) VT

        GROUP BY FemployeeID,courseID,ploNo;");
```

```
        $data1 = DB::table('temp1')
            ->select('coNo', DB::raw('AVG(co_percentage) success'))
            ->groupBy('coNo')
            ->get();
        $data2 = DB::table('temp2')
            ->select('ploNo', DB::raw('AVG(plo_percentage) success'))
            ->groupBy('ploNo')
            ->get();
        $faculty = DB::table('temp1')
            ->join('employees', 'employees.employeeID', '=',
'temp1.FemployeeID')
            ->select(DB::raw('DISTINCT FemployeeID'), 'firstname')
            ->get();
```

CHAPTER 5

CONCLUSION

- PROBLEM AND SOLUTION
- ADDITIONAL FEATURES AND FUTURE DEVELOPMENT
- CONCLUSION AND RECOMMENDATIONS

PROBLEM AND SOLUTION

The problems that we found while working:

- Lack of communication among the team mates.
- Not being able to get the most update and accurate data.
- Was hard to manage a same database for the ADMINS and STAKEHOLDERS to maintain the work in the same website.
- Formatting the data in a proper way, as all the universities has a unique way to upload their file.
- The limited time limit of the semester has prevented us from achieving the full potential of this software. If we had to get chances to gain more resources and data to work with, we could believe we could have achieved much more reliable and accurate results, representations and predictions.

ADDITIONAL FEATURES AND FUTURE DEVELOPMENT

1. The addition of an assessment page where faculties will be able to add marks for a specific assessment of a student throughout the term. Our SPM will automatically generate the achieved CO and PLO.

2. Users will be expanded to also include advisors, where advisors will get relevant information about the students they're advising for improved and more beneficial interactions between students and advisors.

3. The addition of Curriculum Page in the SPM where members of the Higher Management team can add and edit any changes to curriculum. Moreover, faculty members and students can check these updates to stay informed about the latest changes.

CONCLUSION AND RECOMMENDATIONS

In conclusion, our website was designed keeping **easy** for user. The website has no learning purpose as everything is very simple and basic. It remains user friendly for the users to be able to adapt to the environment in a grasp. Because of our website now this work requires less human involvement, increases efficiency and consumes less time. So we believe that our proposed system is now more beneficial and efficient for this work