Student Performance Monitor

SADIA KHAN • ARMUN ALAM • JANNAT KHAIR CHOWDHURY • TANZILA TAHSIN MAYABEE
MUHAMMAD TALHA HASSAN • SAMSUL AMIN

CSE303
Database Management System

FINAL REPORT
Group 01

Sadia Khan	1831231
Jannat Khair Chowdhury	1830105
Tanzila Tahsin Mayabee	1820750
Armun Alam	1810281
Muhammad Talha Hassan	1830698
Samsul Amin	1710393

Contents

CHAPTER 1

INTRODUCTION

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

BACKGROUND OF THE PROJECT

The aim of our project is to design, build and deliver a software that we believe will help universities everywhere to promote a more productive and effective way of evaluating students. At the very core of our project, we have introduced the idea of Course Outcomes (COs) and Program Learning Outcomes (PLOs), where each CO is mapped to a PLO and each PLO represents a specific valuable skill that the students are expected to gain or enhance at the end of that course, such as problem analysis, design, implementation of a skill, etc. To evaluate the students efficiently the project intends to check whether the PLOs that are mapped to the COs requirement is fulfilled or not for each student. The system allows input from IEB to set PLO requirements. The faculties then input the COs for each of their students so that the system can map the COs to PLO accordingly. Through the implementation of this project it was found that the efficiency did not only save time but also improve quality. The PLOs are chosen carefully and specifically to ensure the student gains the most skills out of a course - students can keep track of their progress in each sector and pin-point the areas that need self-improvement and self-growth. In addition, our software hopes to benefit the institutional bodies as well – faculty members, administrative bodies and departmental bodies – to track progress of students, departmental performance and help them distribute and allocate resources better.

OBJECTIVE OF THE PROJECT

Our project intends to create an interactive, user-friendly software that will act as a platform for students, faculties and other members of the university to help improve the quality of education and revolutionize the way we integrate technology into our education. We believe the data we have collected, evaluated and arranged will unlock opportunities for massive advancements in our educational sector and will also contribute significantly to the field of Computer Science. Such being the case, SPM will enhance the project scope so that it will bring about benefits to all of the departments.

SCOPE OF THE PROJECT

The scope is to assist in the efficient and effective implementation of the project through the following tasks:

- ➔ Facilitate the implementation, including planning and management
- ➔ Conduct monitoring of the project
- ➔ Support for review and improvement of the project implementation
- ➔ Project initiation
- ➔ Data Collection
- ➔ Potential Modeling
- ➔ Program Analysis
- ➔ 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)

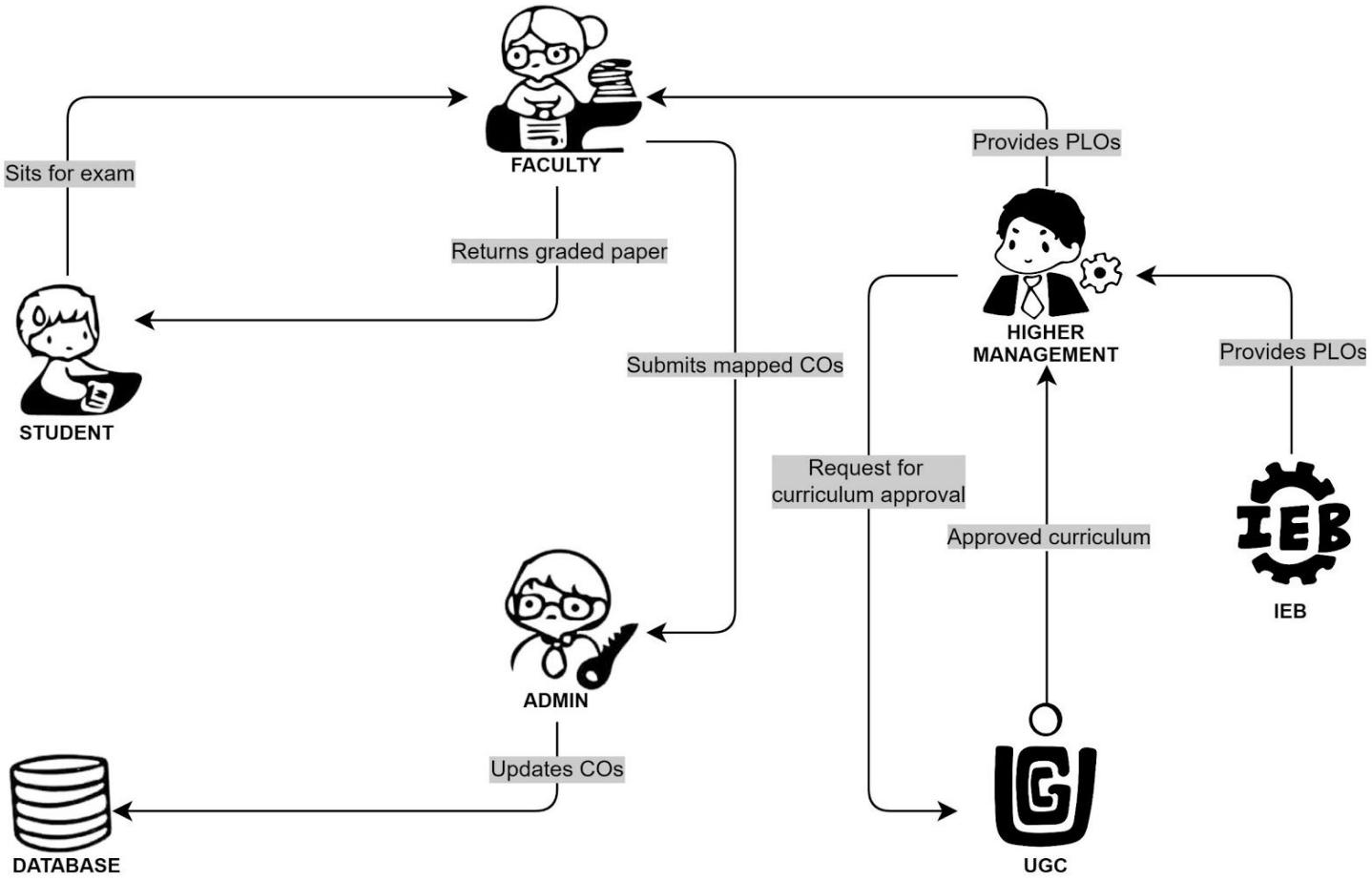


Figure 2.1: Rich Picture (As-Is)

SIX ELEMENTS (AS-IS)

Process	System Roles					
	Human	**Non-Comp Hardware**	**Computing Hardware**	**Software**	**Database**	**Network & Communication**
Set Question Papers according to COs and conduct Examinations	**Faculty Member** a) Set question papers for examinations according to mapped COs. b) Set classroom, time and date of examinations. c) Invigilate examinations and collect test papers. **Student** a) Sit for examinations and submit attempted test papers to faculty.	**Table and Chair** a) To use during examinations. **Pen and Paper** a) For attempting the examinations. b) Questions may be printed on paper. **Clock** a) Setting time for the examination. **Room** a) Designated room for examination.	**Computer** a) Used by faculties to access the COs from the Excel sheet. b) Faculties may also use it to take online examinations and interact with students. c) Students may use it to attend online examinations. **Mobile Phone** a) Some examinations may allow mobile phones for scanning and uploading pdfs to virtual examinations. **Printer** a) Used by faculties to print out question papers for students. **Networking devices (Router, Switch, Bridge, Hub)**	**Microsoft Office** a) The software from which the faculty will collect COs. **Google Classroom** a) Used by faculties and students during examinations. **Operating System** a) Any OS used by the users, e.g. Windows, Mac. **Printing Software** a) Printing software used for printing the question paper. **PDF viewer** a) To view questions in PDF or send the answer in PDF.	**Microsoft Excel Database** a) Faculty access COs from this.	**Internet** a) Used by faculties to access the Excel file via email. b) Used by faculties and students during examinations.

			a) Used by faculties and students to access the Internet.			
UGC approves curriculum based on PLO and CO	**Higher Management (HM)** a) Sends the Curriculum booklet to UGC. b) If it gets approved by the UGC then the HM publishes the Curriculum booklet. c) If it doesn't get approved the HM sets the Curriculum according to the demands of the UGC. d) HM Sends the Updated Curriculum to the Department. **UGC:** a) Receives the Curriculum booklet from the HM. b) Reviews the booklet if it requires changes it sends back feedback to the HM regarding the changes as needed else it is approved by	**Paper** a) It is used to print the booklet.	**Computer** a) Used by the HM to send the Curriculum by mail to the UGC. Also used for editing and updating the Curriculum booklet doc file. **Printer** a) Used by the HM to print the curriculum Booklet. **Networking devices(Router,Internet Cable by ISP Providers** a) Used by HM/UGC to access the internet.	**Microsoft Office** a) Used to edit or update the Curriculum file. **Gmail** a) Used to send mail to the UGC/HM. **Operating System** a) Any type of OS used by the users.e.g. Windows, Linux. **Adobe Acrobat** a) Used to view the PDF file. **Printing Software** a) Used for printing the curriculum doc.	**Microsoft Excel Files** a) HM access the data to edit or update the Curriculum	**Internet Connection** a) It is used by the HM/UGC to send or receive mail.

	the UGC.					
Generating Student Transcripts	**Student** a) Requests information regarding one or more courses from Admin. b) Provides admin with necessary info such as ID/CourseID etc. **Admin** a) Receives requests for Student transcript information. b) Asks for Student's identifier information and list of courses. c) Goes through Excel File to collect the necessary information d) Compiles necessary information into a file and passes it to student.	**Paper** a) The student may have to fill out a form with necessary information (Student ID,CourseIDs.etc.). b) Their course results/transcript information may be printed into a report. **Pen** a) The student may have to fill out a form with necessary information (Student ID/CourseIDs,etc).	**Computer** a) To access the internet and Excel files that hold information about student marks for a course. b) Students may request information via email. **Printer** a) To print out forms the student will have to fill out to give required information. b) To print out transcript/grade information. **Networking devices (Router, Switch, Bridge, Hub)** a) Used by Admin and Students to access the Internet.	**Microsoft Office** a) The data is stored in MS Excel files. **Operating System** a) Any OS used by the users, e.g. Windows, Mac, Linux etc. **Printing Software** a) Printing software used for printing the forms and transcripts. **PDF viewer** b) To view the transcript in PDF form.	**Microsoft Excel File System** a) The marks for each course separated. by COs and PLOs are kept in Excel Files.	**Internet Connection** a) Used to access MS Excel files. b) Used to request information from Admin. c) Used to send reports to students.

Mapping of COs from PLOs	**Faculty Member** a) Maps the COs from PLOs based on the syllabus covered in the course. b) Sends the mapped COs to the admin through email. **Admin** a) Receives the mapped COs from the faculty member. b) Updates it in the excel file containing the COs.	**Paper** a) Used if the faculty member or the admin wishes to print out the mapped COs.	**Computer** a) Used to edit the COs' Excel file. **Printer** a) Used to print out the COs for hardcopy storage backup in case something happens to the digital version.	**Microsoft Excel** a) Used to store the mapped COs. **Web Browser** a) To send and receive the COs through email.	**Microsoft Excel File System** a) Contains the mapped COs. **Hardcopy storage** a) Contains the hardcopy version of the COs' Excel file for backup.	**Internet** a) Used to send the emails containing COs.
Teachers evaluate students and submit COs to Admin to update in Excel Database	**Faculty Member** a) Mark exam papers and list COs for each student. b) Submit COs to Admin c) Send the Exam paper back to Student. **Admin** a) Update COs to Excel Database. **Student** a) Request the faculty to receive the exam paper.	**Pen and Paper** a) Used by Faculty to mark exam papers and list COs.	**Computer** a) Used by the Faculty to Store the exam result. b) Used to send COs to Admin via e-mail. c) Used by Admin to update COs in Excel Database. **Printer** a) Used by Faculty to print out physical copies of COs.	**Microsoft Office** a) Admin stores COs in MS Excel files. b) Faculty stores student's COs. . **Operating System** a) Any OS used by the users, Windows, Mac. **PDF viewer** a) To view questions and mark them digitally in PDF. **Google Mail** a) Used by Faculty to send	**Microsoft Excel Database** a) Used by Admin to update COs.	**Internet** a) Used by Faculties to send COs to admin via email. b) Used by Admin to update COs in Excel database.

			Networking devices (Router, Internet Cable by ISP Providers): a) Used by Faculty and Admin to access the Internet.	COs to admin for update. **Printing Software** a) Faculty prints out COs and delivers physical copy to Admin.		
Higher Management collects PLOs	**IEB** a) Provides PLO. **Faculty** a) Retrieve PLO from Higher Management. **Higher Management** a) Collects PLO from IEB.	**Book** a) Details containing the PLO. **Pen and Paper** a) For collecting and storing all the PLO.	**Computer** a) Higher Management will access the collected PLO which will be given by the IEB. b) IEB use the computer to create PLO which will be given to the universities. c) Faculties will use the computer to retrieve the PLO. **Printer** a) IEB can use printers to print the required documents for PLO. b) Faculties can use it to request for PLO from Higher Management. c) Higher Management can use it to print the	**Microsoft Excel / Google Sheets** a) Used to create the PLO. b) Used to save the retrieved PLO. **Email Software** a) Used for communication between Higher Management, Faculties and IEB.	**Marks Database** a) For storing the mapped CO. b) For storing the PLO. **IEB Database** a) Retrieving the PLO details from IEB.	**Internet** a) Used by faculties to access IEB website. b) Used by Higher Management to store and update the PLO. c) Used by IEB to update and store PLO in their database.

			retrieved PLO. **Mobile** a) Used for communicatio n between Higher Management, Faculties and IEB.			

PROCESS DIAGRAM (AS-IS)

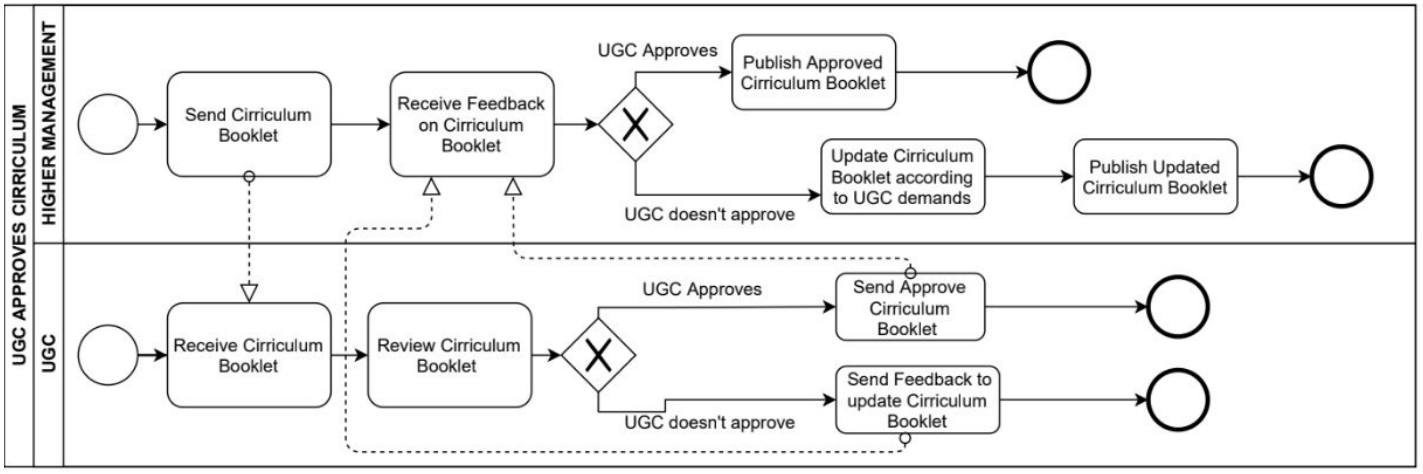


Figure 2.2: Process Diagram for UGC approves curriculum (AS-IS)

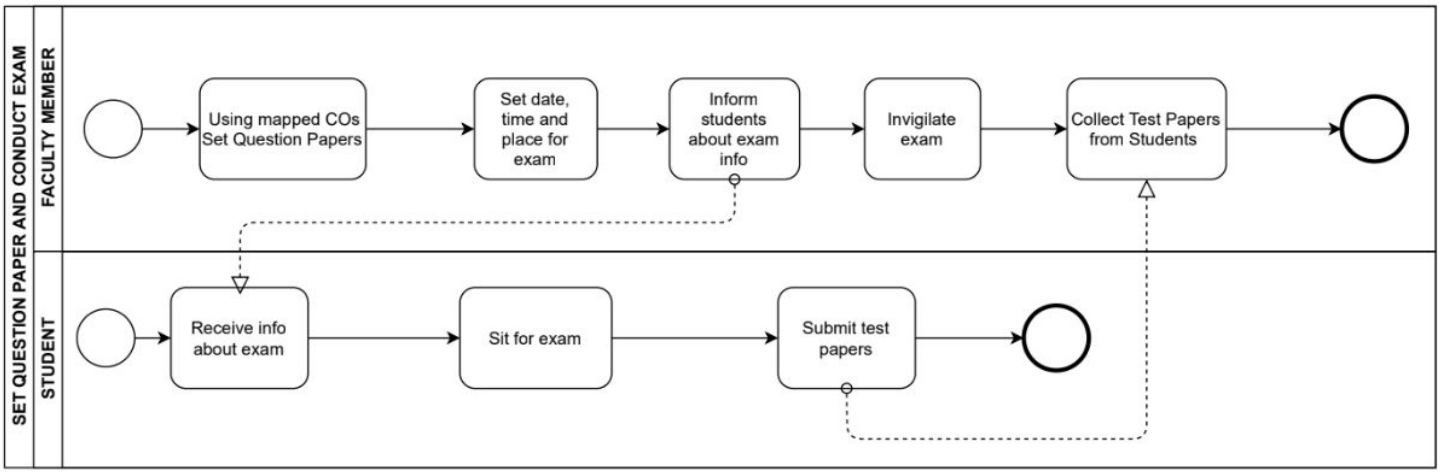


Figure 2.3: Process Diagram for Set question paper and conduct exam (AS IS)

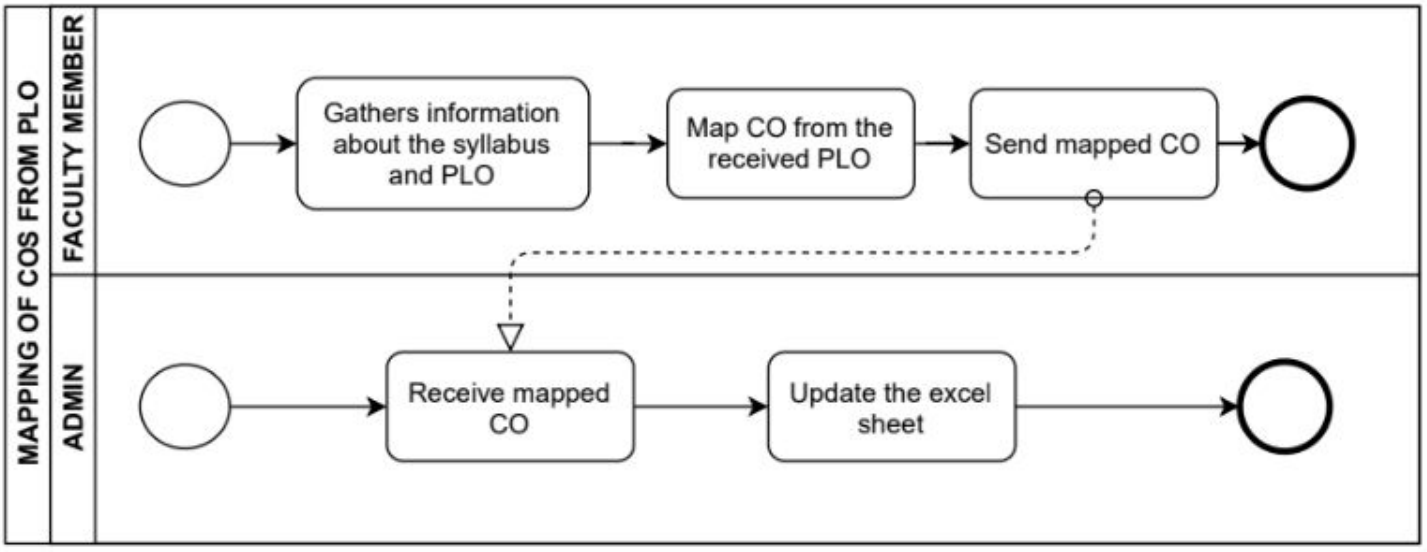


Figure 2.4: Process Diagram for Mapping of COs from PLO (AS-IS)

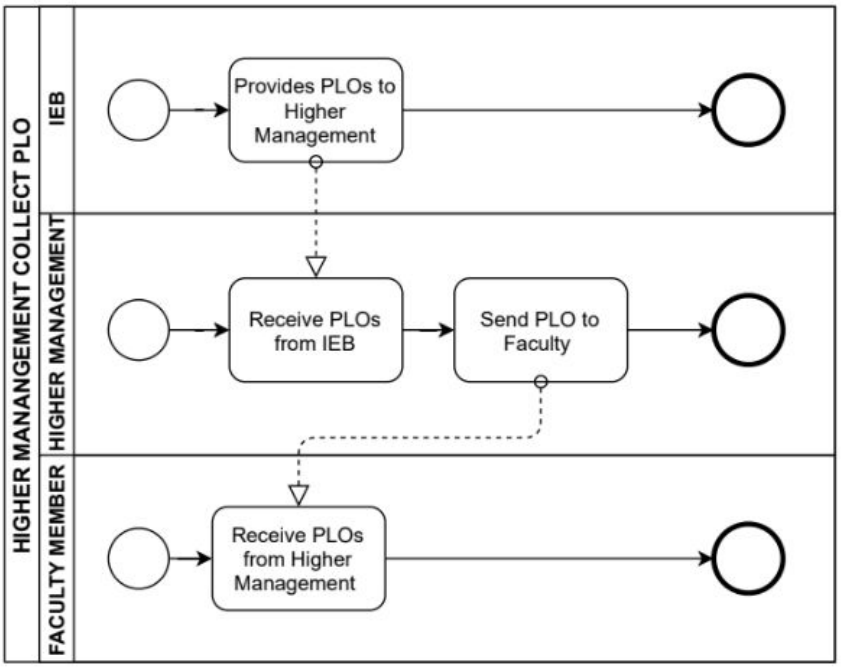


Figure 2.5: Process Diagram for Higher Management Collect PLO (AS-IS)

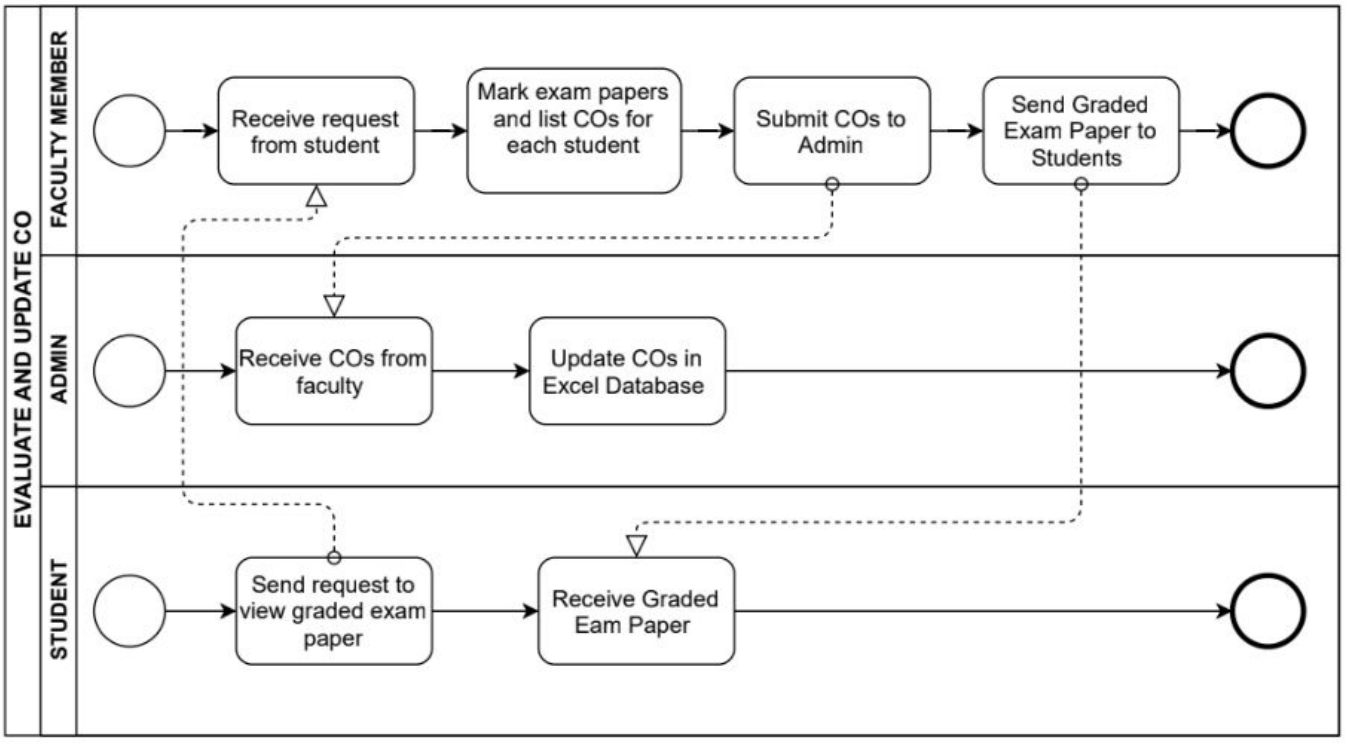


Figure 2.6: Process Diagram for Evaluate and update CO (AS-IS)

PROBLEM ANALYSIS

Process Name	Stakeholders	Concerns (Problems)	Analysis (Reason of the Problem)	Proposed Solution
Teachers evaluate and submit COs to Admin to update Excel Database	1. Faculty Members	The Faculty members have to provide data to Admin and then the Admin enters the data into the Excel Database. This process becomes too time consuming and uses up a lot of extra resources.	The process is time consuming since it takes time for the data to be passed from the faculty to the admin before being updated in the Database. It also wastes unnecessary resources such as paper and printers.	We can eliminate the involvement of the Admin in this process by allowing our Software to be directly accessible to faculty members. Hence, faculty members can directly update the COs on the database without the need of an Admin.
UGC approves curriculum based on PLO and CO	1. Higher Management (HM) 2. UGC	1. HM needs to send the curriculum booklet manually. 2. HM needs to send the updated Curriculum to the department everytime.	1. It will take time for the UGC to receive the Curriculum booklet and process the information. 2. It is a hassle to send manually every time the curriculum is updated	We can transfer the curriculum in our software by which it could be accessed easily by the members and it also could be edited real time by the HM and updated instantly whenever changes are required by the UGC.
CO Entry and Mapping	1. IUB Faculties 2. Admin	1. Faculties mapped each PLO to COs for each course and send it to the Admin 2. Admin receives updated COs and update it to the excel database	They might be subjected to change each semester depending on the course question pattern etc. The process is time consuming as well as the faculties have to send the mapped COs to the Admin and wait for the update	SPM already contains the PLOs so the faculties can directly map the COs from their own account

RICH PICTURE (TO-BE)

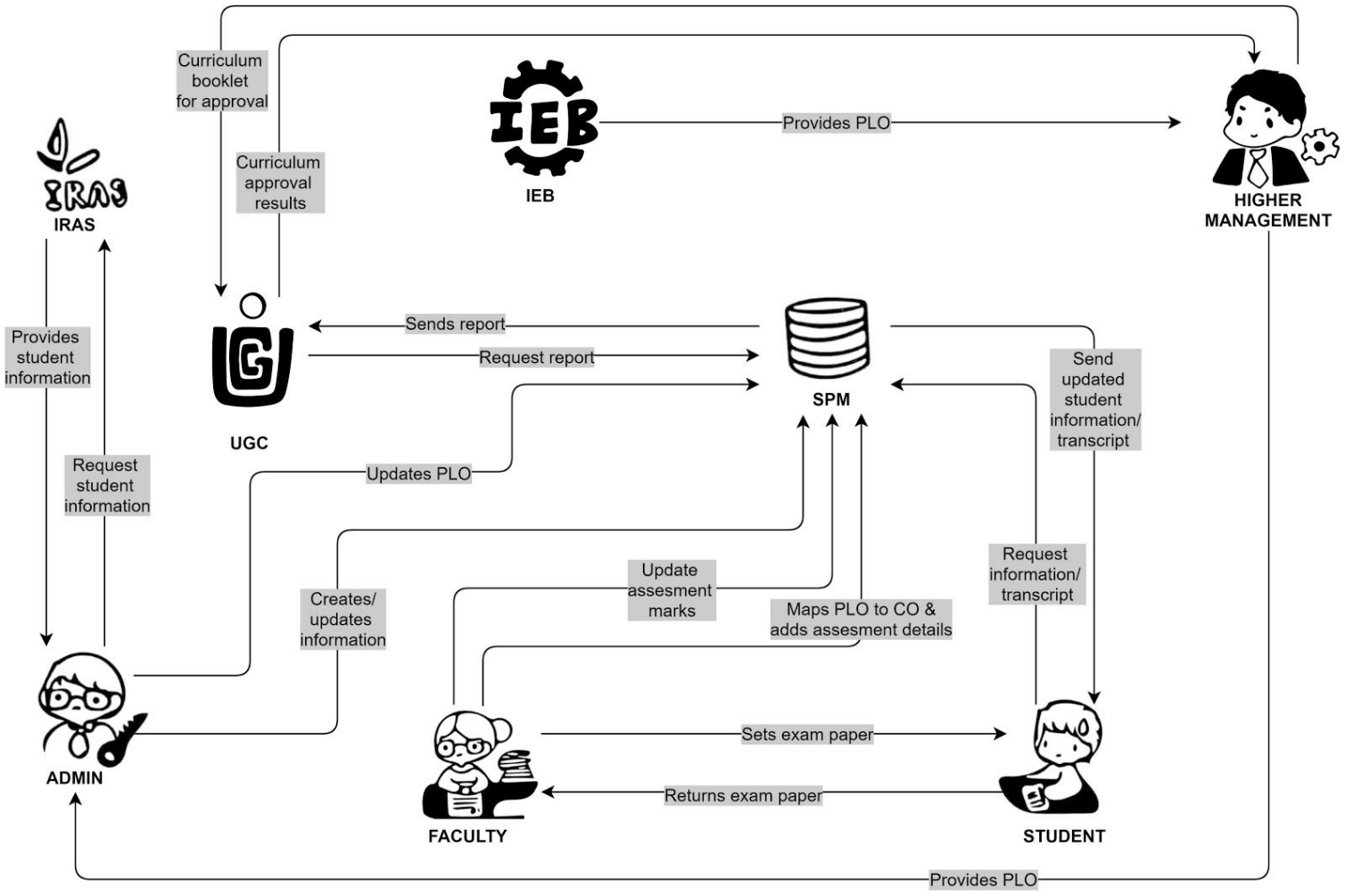


Figure 2.7: Rich Picture (To-Be)

SIX ELEMENTS (TO-BE)

Process	System Roles					
	Human	Non-Comp Hardware	Computing Hardware	Software	Database	Network & Communication
Admin creates user accounts	**Admin** a) Extracts Students' and Faculties' information from iRAS. b) Updates information into the SPM Database. c) Creates new Student/Faculty accounts when required. **Developing team and IT Experts** a) Builds and Maintains the SPM system. **Internet Service Providers (ISP)** a) Provides the Internet service to the data sources, SPM users and SPM system.	**Paper and Stationary** a) Used to collect information in forms from users which do not have accounts in iRAS i.e UGC. **Memo** a) Used by others for requesting access to the system.	**Computers/Laptop** a) SPM admin will use computers to access iRAS so to collect data. b) Users will use the computer to view the data. c) Faculties will use the computer to update and view. **Database Server** a) Used by iRAS to store data and by SPM developers to collect data. **Networking Devices (Router, Switch, Bridge, Hub):** a) Used to access iRAS, SPM.	**Operating Software** a) Used by iRAS and SPM **iRAS** a) Used as a source of data from which the accounts will be created **SPM** a) The software for which the admin will create accounts	**iRAS Database** a) Used by the Admin as a source of information for user accounts. **Excel Files** a) User account data may be stored in excel files which will then be used by SPM. **SPM Database** a) The user account information will be stored here - usernames/account names etc	**Internet** a) It is used to access and store data from the iRAS to SPM.

Retrieves PLO and updates PLO and mapped CO to SPM	**IEB** a) Provides PLO to Higher Management. **Faculty** a) Updates mapped CO to SPM. **Admin** a) Adds PLO to SPM b) Maps PLO to CO **Higher Management** a) Provides the PLO to Admin	**Book** a) Contains details of the PLO. **Pen and Paper** a) For mapping all the CO with the received PLO from IEB.	**Computer** a) IEB will use the computer to access SPM and update the SPM when needed. b) Faculties will use the computer to view the course details and mapping the CO. c) IEB uses the computer to create PLO which will be given to the universities. **Mobile** a) Used for communicatio n between Faculties and IEB. **Networking devices (Router, Switch, Bridge, Hub)** a) Used to access the Internet.	**SPM** a) Used to create the PLO. b) Used to create CO. c) Updates CO and PLO. **Email Software** a) Used for communication between Faculties and IEB. **Operating System** a) Any OS used by the users, Windows, Mac.	**SPM Database** a) For storing the mapped CO. b) For storing the PLO. **IEB Database** a) Retrieving the PLO details from IEB.	**Internet** a) Used by faculties to access IEB website and update SPM b) Used by IEB to update and store PLO in their database.

Set Question Papers according to COs and conduct Examinations	**Faculty** a) Retrieve COs from Software. b) Format and set question papers for examinations according to COs collected. c) Set classroom, time and date of examinations. d) Conduct examinations and collect test papers. **Student** a) Sit for examinations and submit attempted test papers to faculty.	**Table and Chair** a) To use during examinations. **Pen and Paper** a) For attempting the examinations. b) Questions may be printed on paper **Clock** a) Setting time for the examination **Room** a) Designated room for examination	**Computer** a) Used by faculties to access the COs from the software. b) Faculties may also use it to take online examinations and interact with students. c) Students may use it to attend online examinations. **Mobile phones** a) Some examinations may allow mobile phones for scanning and uploading pdfs to virtual examinations. **Printer** a) Used by faculties to print out question papers for students. **Database Server** a) Used by faculties to access and collect the COs to set questions. **Networking devices (Router,**	**SPM** a) The software from which the faculty will collect COs. **Google Classroom** a) Used by faculties and students during examinations." **Operating System** a) Any OS used by the users, Windows, Mac **Printing Software** a) Printing software used for printing the question paper **PDF viewer** a) To view question in PDF or send the answer in PDF	**SPM Database** a) Faculty access COs from this.	**Internet** a) Used by faculties to access our software and its database b) Used by faculties and students during examinations

			Switch, Bridge, Hub) a) Used by faculties and students to access the Internet.			
Teachers will evaluate students and update marks in SPM	**Faculty** a) Update the database with the achieved COs marks of the Student. b) Send the Exam paper back to the Student. **Student** a) Request the faculty to receive the Exam paper.	**Paper and Pen** a) The exam paper is marked with the pen.	**Computer/Laptop** a) Used by the faculty member to store the exam result and update the database with achieved COs. **Database Server** a) Used by faculty members to access and store or update the database. **Mobile** a) Used for communication between faculty members and students. **Networking devices (Router, Internet Cable by ISP Providers)** a) Used by faculty members to access the Internet.	**SPM** a) It is used to store and update the Student profile. **Operating System** a) Any OS used by the users, e.g. Windows, Mac.	**SPM Database** a) It is used to store the updated data of the Student.	**Internet Connection** a) It is used by the faculty to access the SPM software and the database

Request Transcript from SPM	**Student** a) Request the system to generate a transcript from the database. b) Receive the transcript and view/print it.	**Paper** a) Used in case the student wants to print the transcript	**Computer/Phone** a) Used to request for the transcript. b) View the transcript. **Printer** a) Print the transcript on a piece of paper. **Networking devices (Router, Switch, Bridge, Hub):** a) Used to access the Internet	**SPM** a) Queries the database to collect the required/relevant information of the student to generate a transcript, e.g. grades, courses, credits etc. b) Calculate the relevant information from the student's data, e.g. CGPA. c) Create a web page for the student to see their transcript. **Operating System** a) Any OS used by the users, Windows, Mac **Printing Software** a) Printing software used for printing the transcript **PDF viewer** a) To view transcript in PDF	**SPM Database** a) Provide the SPM software with the information it queries from the database.	**Internet** a) To access the SPM software.

Update Curriculum in SPM	**Admin** a) Updates changes in the curriculum	**Paper** a) Used in case the updates are noted	**Computer/Phone** a) Used to update changes. **Printer** a) Print the curriculum data report **Networking devices (Router, Switch, Bridge, Hub)** a) Used to access the Internet.	**SPM** a) SPM will contain the updated data. **Operating System** a) Any OS used by the users, e.g. Windows, Mac. **Printing Software** a) Printing software used for printing the curriculum report. **PDF viewer** a) To view the curriculum report in PDF.	**SPM Database** a) Provide the SPM software with the information it queries from the database.	**Internet** a) To access the SPM software.
UGC Requests Reports from SPM to evaluate Department/ University and Generate Report	**UGC** a) UGC may request student performance reports based on a particular department or the whole university. b) They may provide feedback regarding their evaluations. **Admin** a) Admin will prepare a report on students' performance based on given parameters.	**Paper and Stationary** a) UGC may send out an application requesting to evaluate IUB using SPM. **Memo** a) Used to send messages to the SPM development team regarding UGC requests.	**Printer/Fax** a) Used to print out reports to send to UGC. b) Reports may also be faxed. **Laptop/PC & other devices** a) UGC will need a computer or phone to access SPM. b) UGC may use one for evaluating and sending results. **Networking devices**	**SPM** a) SPM will contain all individual student evaluations. b) Organises and calculates individual data to show various trends of the department/school/university. **Email Software** a) Requests for reports may be sent via email. **Operating System**	**SPM Database** a) Will contain all data regarding COs and PLOs for individual students.	**Internet** a) To access SPM. b) Reports may be requested or sent via email which is accessed by using the Internet.

			(Router, Switch, Bridge, Hub) a) Used to access the Internet.	a) Any OS used by the users, e.g. Windows, Mac. **Printing Software** a) Printing software used for printing the report. **PDF viewer** a) To view report in PDF		

PROCESS DIAGRAM (TO-BE)

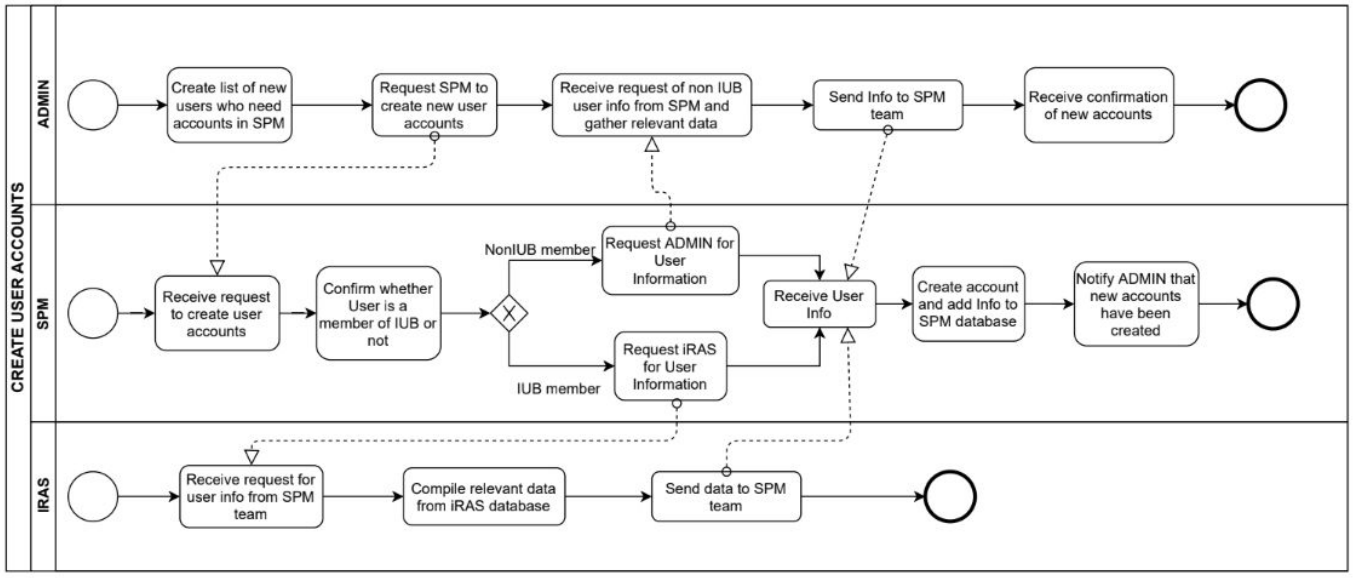


Figure 2.8: Process diagram for creating user account

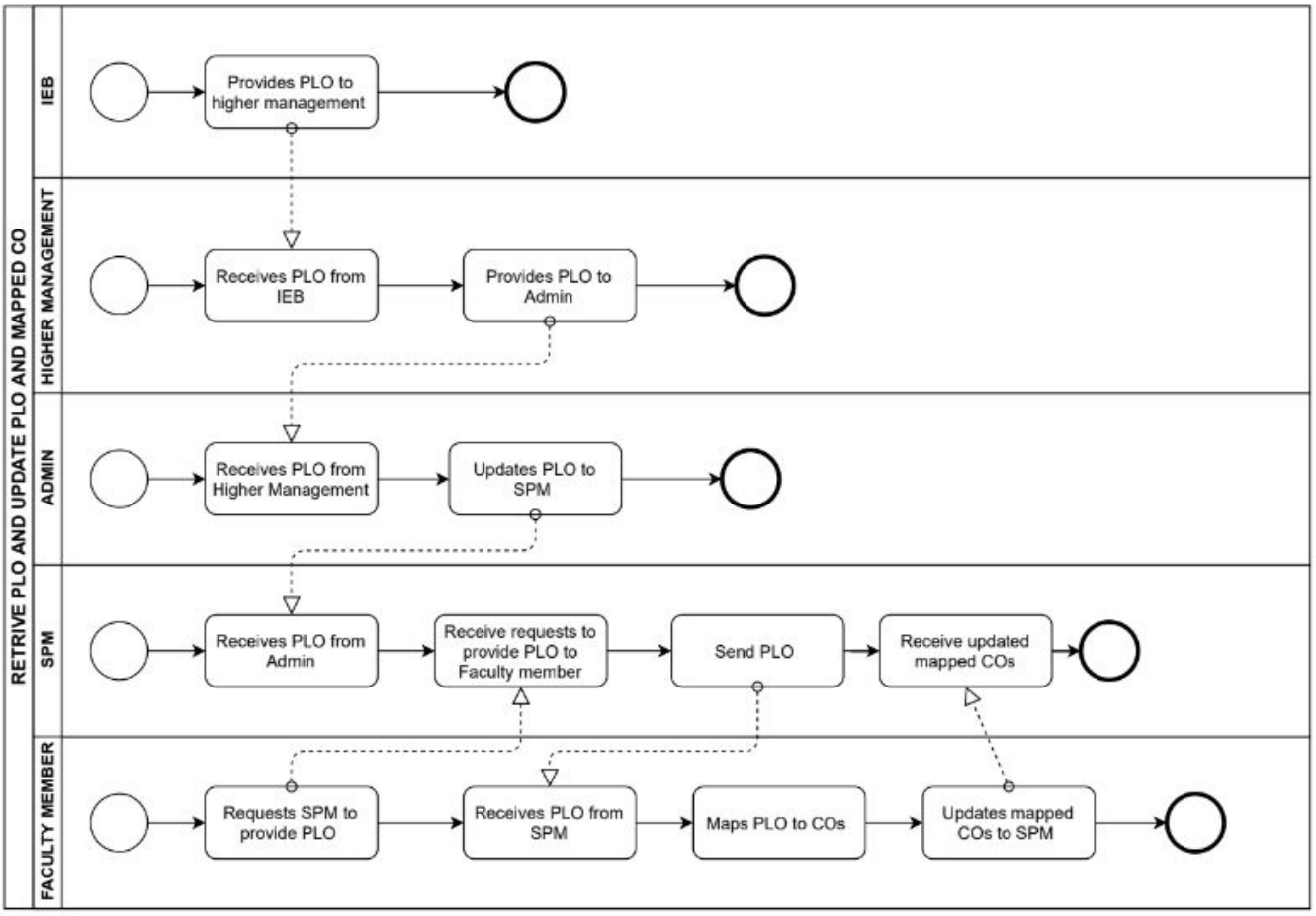


Figure 2.9: Process diagram for retrieving and updating PLO and mapped CO

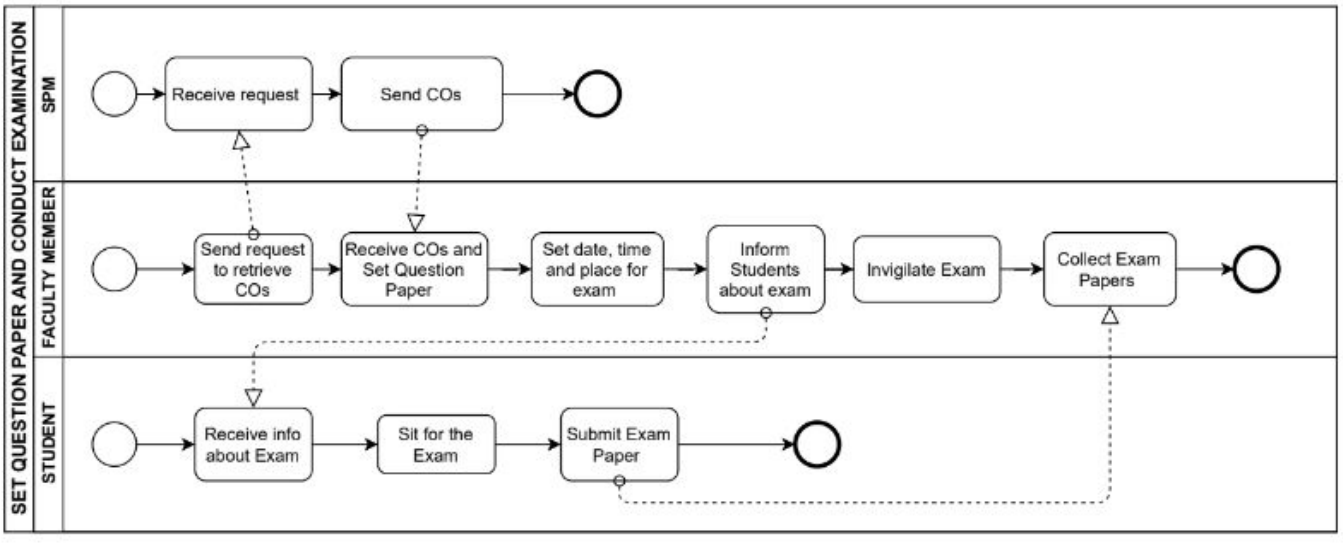


Figure 2.10: Process diagram for setting exam paper and conducting examination

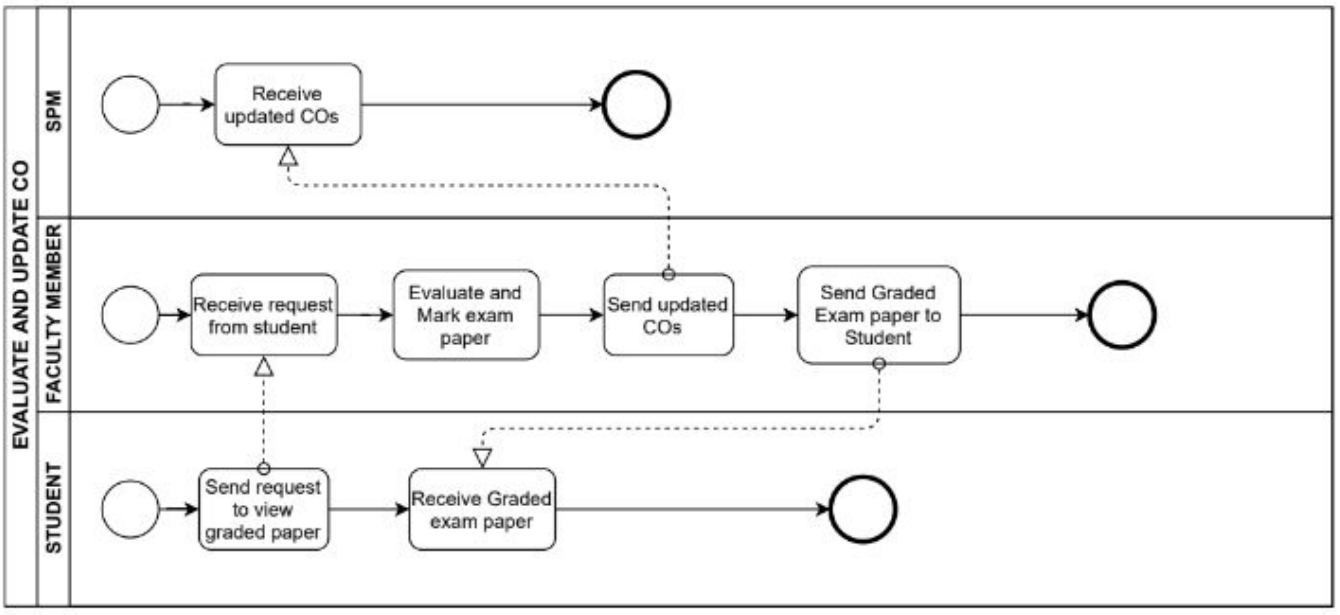


Figure 2.11: Process diagram for evaluating and updating CO

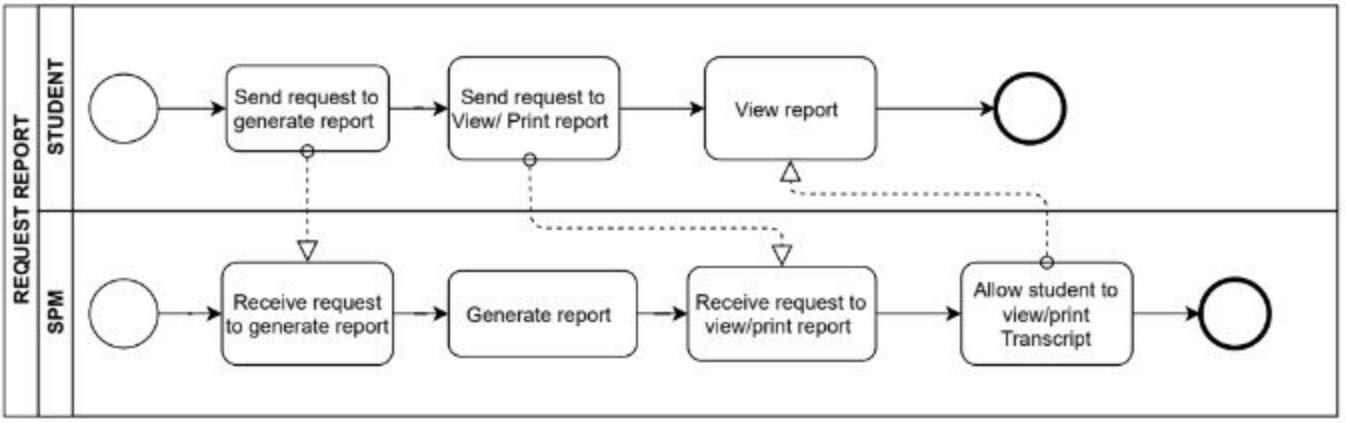


Figure 2.12: Process diagram for requesting report

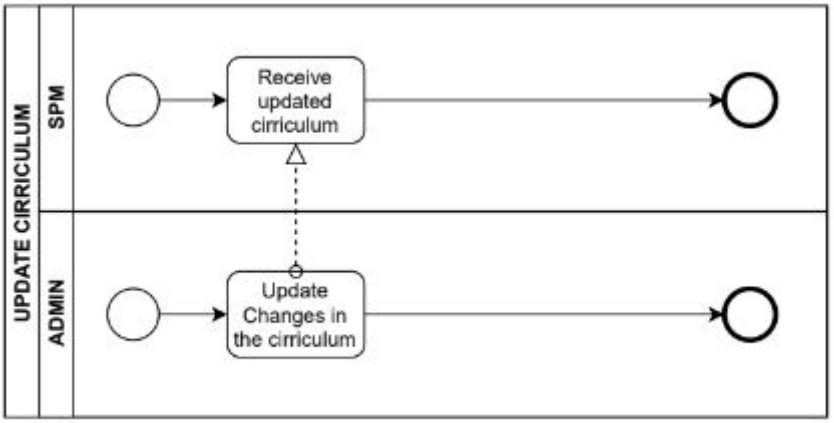


Figure 2.13: Process diagram for updating curriculum

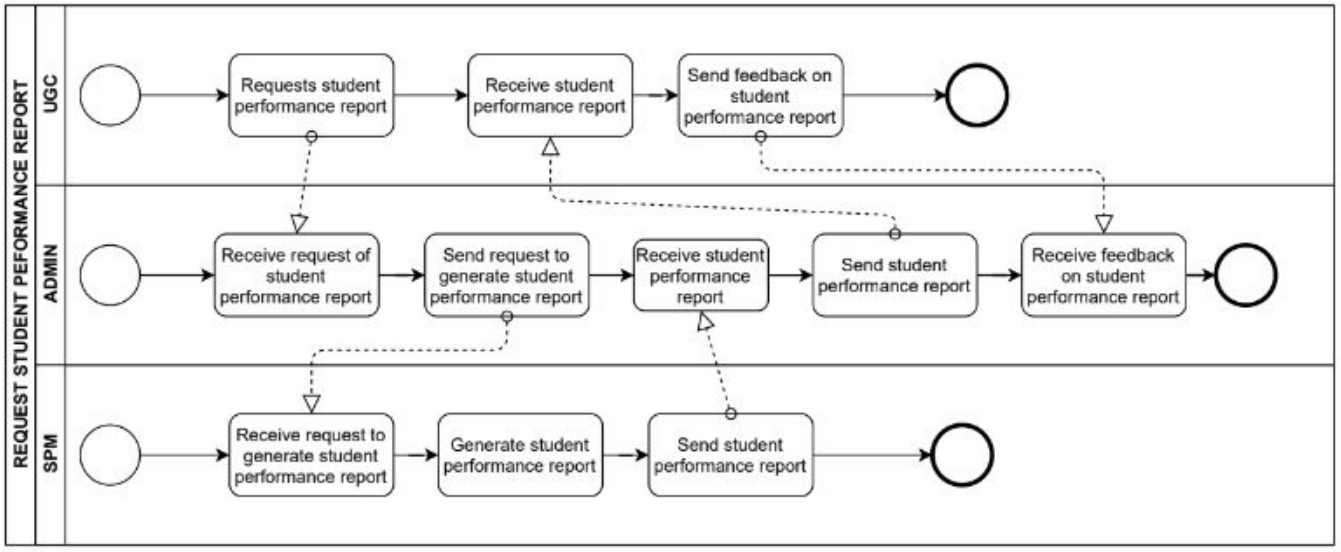


Figure 2.14: Process diagram for requesting student performance report

CHAPTER 3

LOGICAL SYSTEM DESIGN

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

BUSINESS RULE

The goal of the software is to increase efficiency in monitoring the students performance. The SPM system is where all the PLO(Program Learning Outcome) and CO(Course Outcome) is stored. The CO is needed to be updated by the faculty for each course and before the semester starts to map the COs to the PLOs so that they can check if each student has achieved the required PLOs.

In the system, IEB has no authorisation to update the PLOs, so it has to send it to the Admin and then the Admin updates the PLO for the faculties to map. The faculties can update the COs based on the given PLOs. The students can view their achieved PLOs for a particular course they've taken and see the required PLOs for the program in the system UGC has no authorisation in monitoring the students performance so they have to request it through admin in order to view it.

ENTITY RELATIONSHIP DIAGRAM

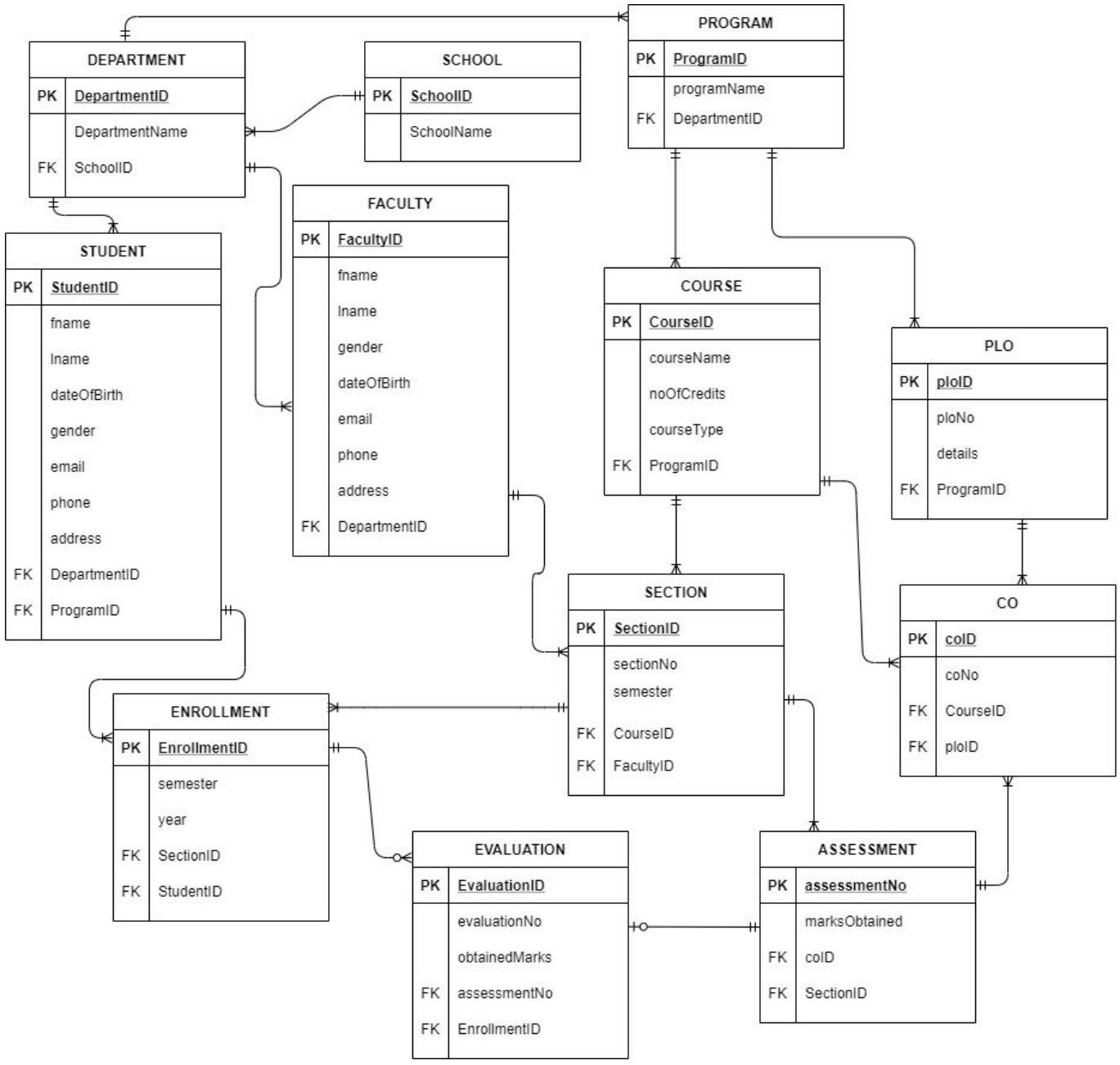


Figure 3.1: Entity Relationship Diagram of SPM

ENTITY RELATIONSHIP DIAGRAM TO RELATIONAL SCHEMA

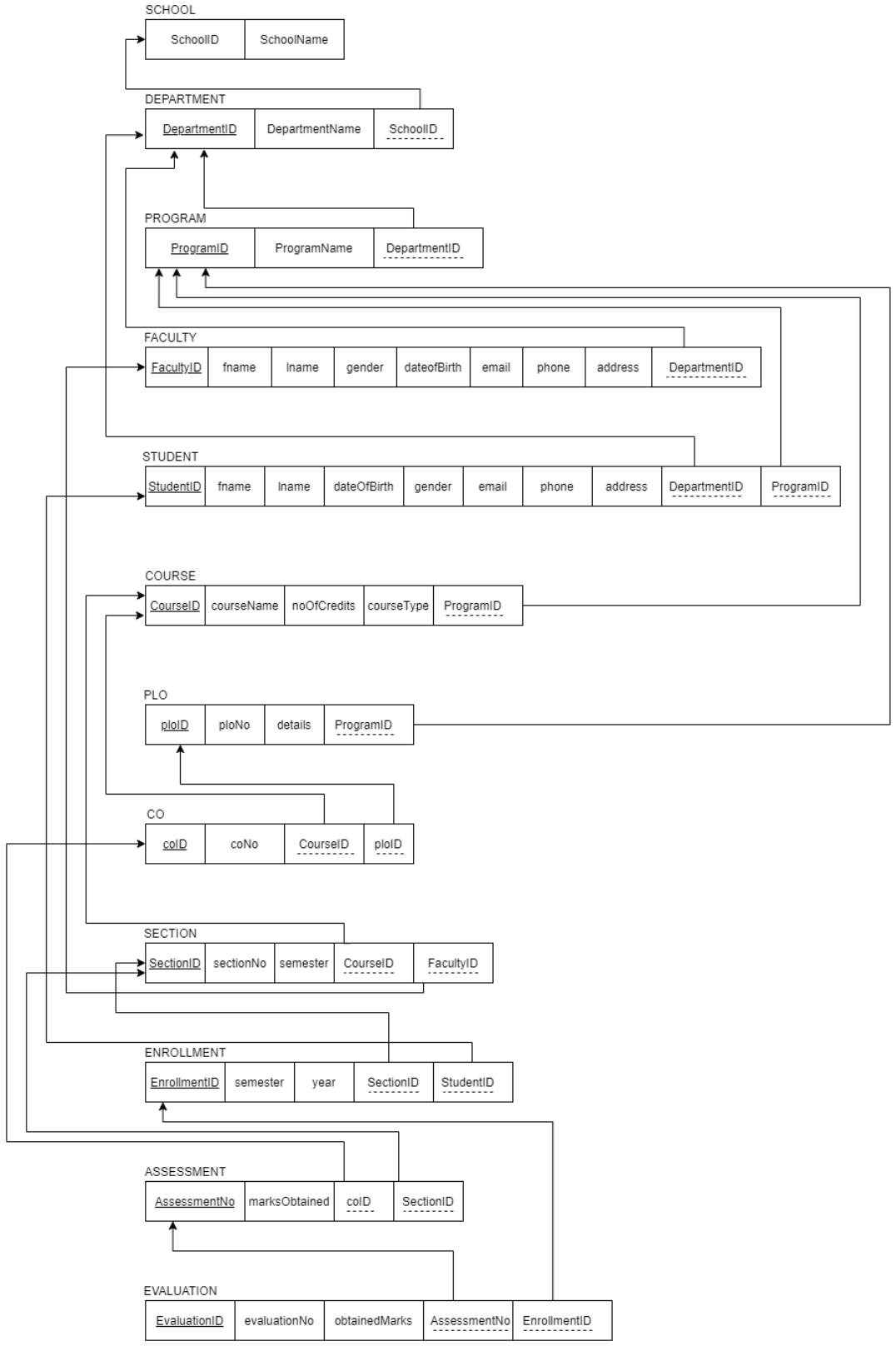


Figure 3.2: Relational Schema Diagram of SPM

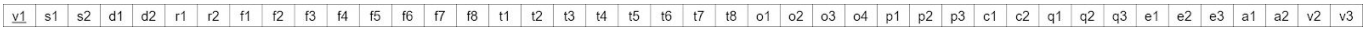
NORMALIZATION

Enrollment	EnrollmentID	e1	Evaluation	EvaluationID	v1
	semester	e2		evaluationNo	v2
	year	e3		obtainedMarks	v3
	StudentID	t1		assessmentNo	a1
	SectionID	q1		EnrollmentID	e1
Section	SectionID	q1	Student	StudentID	t1
	sectionNo	q2		fname	t2
	semester	q3		lname	t3
	CourseID	o1		dateOfBirth	t4
	FacultyID	f1		gender	t5
Course	CourseID	o1		email	t6
	courseName	o2		phone	t7
	noOfCredits	o3		address	t8
	courseType	o4		DepartmentID	d1
	ProgramID	r1		ProgramID	r1
Program	ProgramID	r1	Faculty	FacultyID	f1
	programName	r2		fname	f2
	DepartmentID	d1		lname	f3
Course	CourseID	o1		gender	f4
	courseName	o2		dateOfBirth	f5
	noOfCredits	o3		email	f6
	courseType	o4		phone	f7
	ProgramID	r1		address	f8
School	SchoolID	s1		DepartmentID	d1
	SchoolName	s2	Assessment	assessmentNo	a1
CO	coID	c1		marksObtained	a2
	courseName	c2		coID	c1
	ploID	p1		SectionID	q1
	CourseID	o1	PLO	ploID	p1
Department	DepartmentID	d1		ploNo	p2
	DepartmentName	d2		details	p3
	SchoolID	s1		ProgramID	r1

s1➔	s2
d1➔	d2, s1
r1➔	r2, d1
f1➔	f2, f3, f4, f5, f6, f7, f8, d1
t1➔	t2, t3, t4, t5, t6, t7, t8, r1, d1
o1➔	o2, o3, o4, r1
p1➔	p2, p3, r1
c1➔	c2, p1, o1
q1➔	q2, q3, o1, f1
e1➔	e2, e3, q1, t1
a1➔	a2, c1, q1
v1➔	v2, v3, a1, e1

SchoolID➔	SchoolName
DepartmentID➔	DepartmentName, SchoolID
ProgramID➔	programName, DepartmentID
FacultyID➔	fname, lname, gender, dateOfBirth, email, phone, address, DepartmentID
StudentID➔	fname, lname, dateOfBirth, gender, email, phone, address, DepartmentID, ProgramID
CourseID➔	courseName, noOfCredits, courseType, ProgramID
ploID➔	ploNo, details, ProgramID
coID➔	courseName, ploID, CourseID
SectionID➔	sectionNo, semester, CourseID, FacultyID
EnrollmentID➔	semester, year, SectionID, StudentID
assessmentNo➔	marksObtained, ocID, SectionID
EvaluationID➔	evaluationNo, obtainedMarks, assesmentNo, EnrollmentID

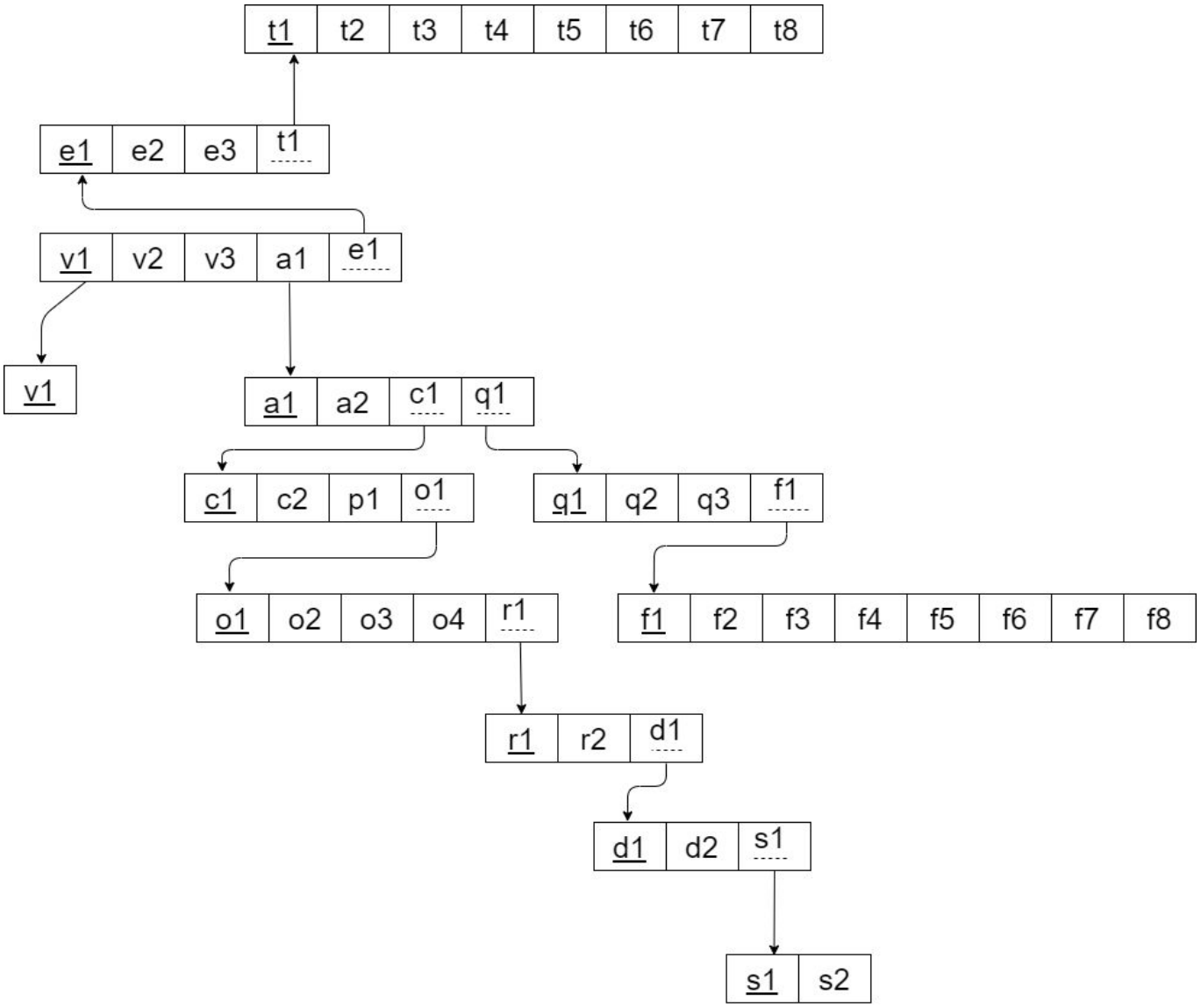
1NF


v1 s1 s2 d1 d2 r1 r2 f1 f2 f3 f4 f5 f6 f7 f8 t1 t2 t3 t4 t5 t6 t7 t8 o1 o2 o3 o4 p1 p2 p3 c1 c2 q1 q2 q3 e1 e2 e3 a1 a2 v2 v3

2NF


s1 s2 d1 d2 r1 r2 f1 f2 f3 f4 f5 f6 f7 f8 t1 t2 t3 t4 t5 t6 t7 t8 o1 o2 o3 o4 p1 p2 p3 c1 c2 q1 q2 q3 e1 e2 e3 a1 a2 v2 v3
v1

3NF



BCNF

No non-key can identify any primary key or part of the primary key. Therefore, all the relations are in BCNF.

DATA DICTIONARY

Department_T

Name	Data Type	Size	Remark
cdepartmentID	VARCHAR	5	This is the Primary Key of the Department. Example: "CSE"
cdepartmentName	VARCHAR	30	This is the name of the Department. Example: "Computer Science and Engineering"
cschool_id	VARCHAR	5	This is the Foreign Key of the table School. Example: "SETS"

School_T

Name	Data Type	Size	Remark
cschoolID	VARCHAR	5	This is the Primary Key of School Example: "SETS"
cschoolName	VARCHAR	30	This is the name of the School. Example: "School of Engineering, Technology and Science"

Program_T

Name	Data Type	Size	Remark
cprogramID	VARCHAR	5	This is the Primary Key for a Program Example: "B.Sc".
cprogramName	VARCHAR	30	This is the name of the Degree Program. Example: "Bachelor of Science"
cdepartment_id	VARCHAR	5	This is the Foreign Key from the Department table. Example: "CSE"

Student_T

Name	Data Type	Size	Remark
cstudentID	VARCHAR	7	This is the Primary Key for the Student. Example: "1800001"
cfname	VARCHAR	30	This is the first name of the Student. Example: "Muhammad Talha"
clname	VARCHAR	30	This is the last name of the Student. Example: "Hassan"

ddateOfBirth	DATE	DD-MM-YYYY	This the Date of Birth of the Student. Example: "01-01-1998"
cgender	VARCHAR	1	This is the gender of the Student. Example: "M"
cemail	VARCHAR	30	This is the email address of the Student. Example: "1830105@iub.edu.bd"
cphone	VARCHAR	15	This is the phone number of the Student. Example: "0191211141"
caddress	VARCHAR	30	This is the address of the Student. Example: "House 1, Road 1, Sector 1, Uttara, Dhaka, Bangladesh"
cdepartment_id	VARCHAR	5	This is the Foreign Key from the Department table. Example: "CSE"
cprogram_id	VARCHAR	5	This is the Foreign Key from Program table Example: "B.Sc".

CO_T

Name	Data Type	Size	Remark
nid	INTEGER		This is the Primary Key for Course Outcome. Example:
ncoNo	INTEGER		This is the number of the Course Outcome. Example: "1"
ccourse_id	VARCHAR	7	This is the Foreign Key from the Course table. Example: "CSE101"
cplo_id	VARCHAR	5	This is the foreign key from the Program Learning Outcome table. Example: "PLO1"

PLO_T

Name	Data Type	Size	Remark
cploNo	VARCHAR	5	This is the primary key for Program Learning Outcome. Example: "PLO1"
cdetails	VARCHAR	50	This is the details of the Program Learning Outcome. Example: "An ability to select and apply the knowledge, techniques, skills, and modern

			tools of the computer science and engineering discipline"
cprogram_id	VARCHAR	5	This is the foreign key from Program table Example: "B.Sc".

Faculty_T

Name	Data Type	Size	Remark
cfacultyID	VARCHAR	4	This is the Primary Key for Faculty. Example: "1801"
cfname	VARCHAR	30	This is the first name of the Faculty. Example: "Mahady"
clname	VARCHAR	20	This is the last name of the Faculty Example: "Hasan"
ddateOfBirth	DATE	DD-MM-YYYY	This the Date of Birth of the Faculty. Example: "01-01-1993"
cgender	VARCHAR	1	This is the gender of the Faculty . Example: "M"
cemail	VARCHAR	30	This is the email address of the Faculty. Example: "mahady@iub.edu.bd"
cphone	VARCHAR	15	This is the phone number of the Faculty. Example: "01292383111"
caddress	VARCHAR	30	This is the address of the Faculty. Example: "House 1, Road 1, Sector 1, Uttara, Dhaka, Bangladesh"
cdepartment_id	VARCHAR	5	This is the Foreign Key from the Department table. Example: "CSE"

Course_T

Name	Data Type	Size	Remark
ccourseID	VARCHAR	7	This is the Primary Key for the Course. Example: "CSE203"
ccourseName	VARCHAR	40	This is the name of the Course. Example: "Data Structure"
nnoOfCredits	INTEGER		This is the credit for the Course. Example: "3"

ccourseType	VARCHAR	10	This is the type of the Course. Example: "Core"
cprogram_id	VARCHAR	5	This is the Foreign Key from Program table Example: "B.Sc".

Section_T

Name	Data Type	Size	Remark
nid	INTEGER		This is the Primary Key for Section
nsectionNo	INTEGER		This is the section number. Example: "1"
ccourse_id	VARCHAR	7	This is the foreign key from the Course table. Example: "CSE101"
cfaculty_id	VARCHAR	4	This is the foreign key from Faculty table Example: "CO1"

Enrollment_T

Name	Data Type	Size	Remark
nenrollmentID	INTEGER		This is the Primary Key for Enrollment
csemester	VARCHAR	6	This is the semester of Enrollment Example: "Summer"
dyear	YEAR	yyyy	This is the year of Enrollment Example: "2018"
nsection_id	INTEGER		This is the Foreign Key from Section table
cstudent_id	VARCHAR	7	This is the Foreign key from the Student Table. Example: "1800001"

Assessment_T

Name	Data Type	Size	Remark
nassessmentNo	INTEGER		This is the Primary Key for Enrollment
cmarks	VARCHAR	6	This is the semester of Enrollment Example: "Summer"
nsection_id	INTEGER		This is the Foreign Key from Section table
nco_id	INTEGER		This is the Foreign Key from the Course

			Outcome table Example:

Evaluation_T

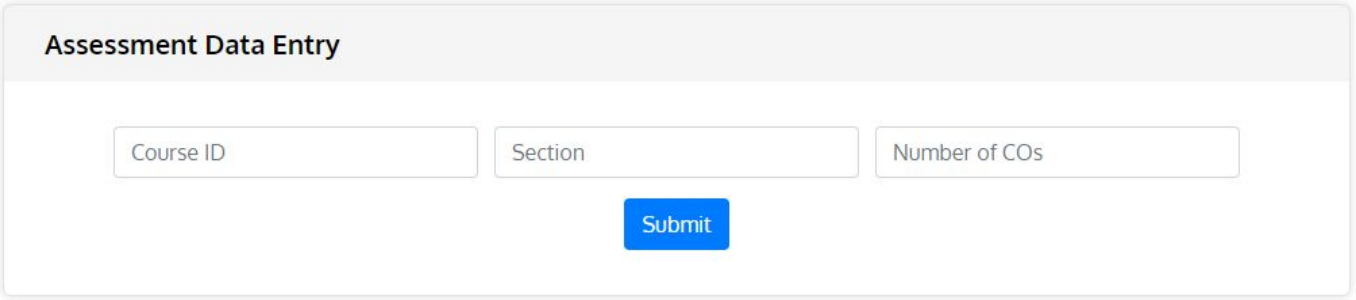
Name	Data Type	Size	Remark
nevaluationNo	INTEGER		This is the Primary Key for Evaluation
nobtainedMarks	FLOAT		This is the marks obtained by the Student Example: "29.5"
nassessment_id	INTEGER		This is the Foreign Key from Assessment table

CHAPTER 4

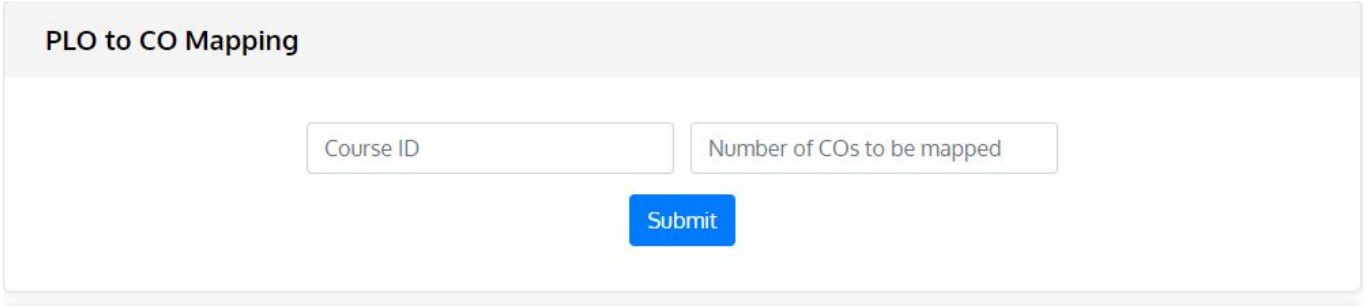
PHYSICAL SYSTEM DESIGN

- INPUT FORMS
- OUTPUT FORMS

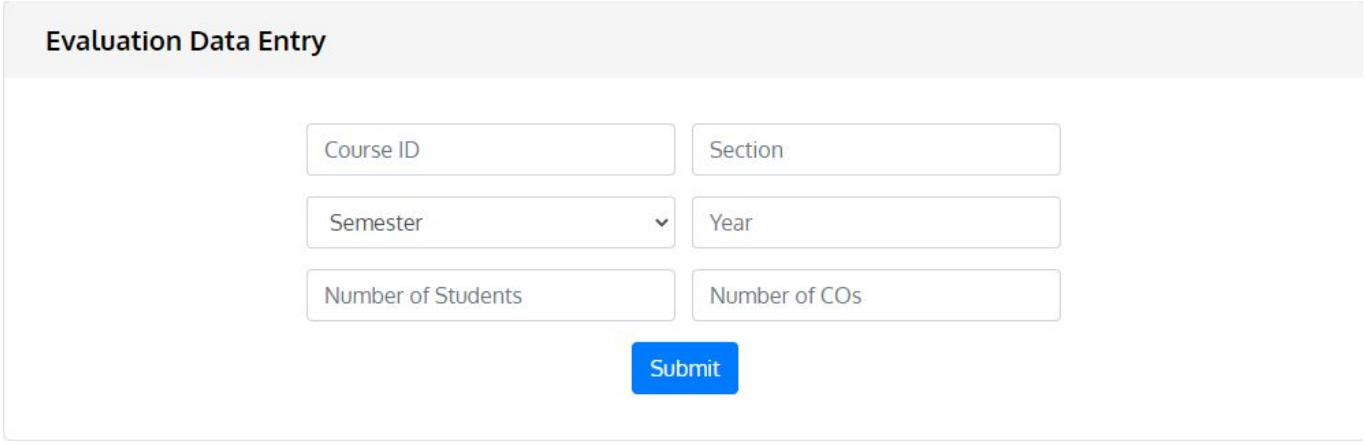
INPUT FORMS




```
@allowedUsers(allowedRoles=['Faculty'])
def assessment(request):
    if request.method == 'POST':
        # course_id = request.POST.get('course-id')
        faculty_id = '1234'
        course_id = request.POST.get('course-id')
        sectionNo = request.POST.get('section')
        coMarks = request.POST.getlist('coMarks')

        section_id = None
        try:
            section_id = Section_T.objects.raw('''
                SELECT *
                FROM mainapp_section_t
                WHERE course_id = '{}' AND sectionNo = {};
            '''.format(course_id, sectionNo))
            section_id = section_id[0].id
        except:
            section_id = None

        if section_id is None:
            section = Section_T(sectionNo=sectionNo, course_id=course_id,
faculty_id=faculty_id)
            section.save()
            section_id = section.id

        for j in range(1, len(coMarks) + 1):
            co_id = CO_T.objects.raw('''
                SELECT *
                FROM mainapp_co_t
                WHERE course_id = '{}' AND coNo = {}
            '''.format(course_id, j))
            assessment = Assessment_T(section_id=section_id, co_id=co_id[0].id,
marks=coMarks[j - 1])
            assessment.save()

        return redirect('dataentry')
```




```
@allowedUsers(allowedRoles=['Faculty'])
def mapping(request):
    if request.method == 'POST':
        # print(request.POST)
        # print(request.POST.get('course-id'))
        # print(request.POST.getlist('coMaps'))

        course_id = request.POST.get('course-id')
        coMaps = request.POST.getlist('coMaps')

        course = Course_T(course_id, program_id='BSc', noOfCredits=3)
        course.save()

        for i in range(len(coMaps)):
            co = CO_T(coNo=i + 1, course_id=course_id, plo_id=coMaps[i])
            co.save()


    return redirect('dataentry')
```

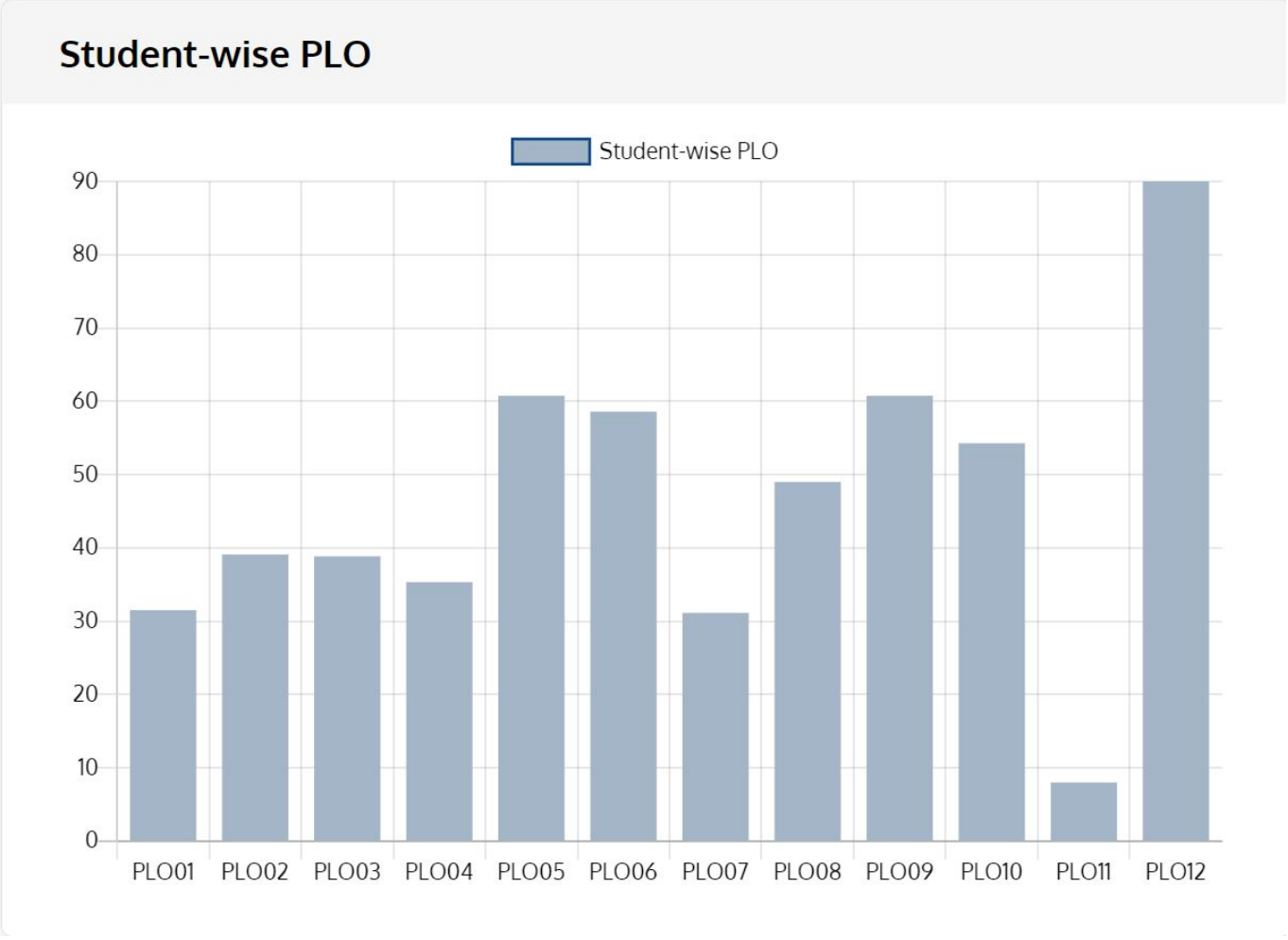



```
@allowedUsers(allowedRoles=['Faculty'])
def evaluation(request):
    if request.method == 'POST':
        course_id = request.POST.get('course-id')
        section = request.POST.get('section')
        semester = request.POST.get('semester')
        year = request.POST.get('year')

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

```
section_id = None
    try:
        section_id = Section_T.objects.raw('''
            SELECT *
            FROM mainapp_section_t
            WHERE course_id = '{}' AND sectionNo = {};
        '''.format(course_id, section))
        section_id = section_id[0].id
    except:
        section_id = None
    assessment_list = []
    coLength = 0
    try:
        coLength = len(coMarks[0]) + 1
    except:
        coLength = 0
    for j in range(1, coLength):
        assessment_id = None
        try:
            assessment_id = Assessment_T.objects.raw('''
                SELECT *
                FROM mainapp_assessment_t
                WHERE section_id = {} AND co_id IN (
                    SELECT id
                    FROM mainapp_co_t
                    WHERE course_id = '{}' AND coNo = {}
                )
            '''.format(section_id, course_id, j))
            assessment_list.append(assessment_id[0].assessmentNo)
        except:
            assessment_id = None
            assessment_list.append(assessment_id)

# print(course_id)
    # print(student_id)
    # print(semester)
    # print(year)
    # print(section)
    # print(coMarks)
    # print(section_id)
    # print(assessment_list)

for i in range(len(student_id)):
        enrollment_id = None
        try:
            enrollment_id = Enrollment_T.objects.raw('''
                SELECT *
                FROM mainapp_enrollment_t
                WHERE student_id = '{}' AND section_id = {}
            '''.format(student_id[i], section_id))
            enrollment_id = enrollment_id[0].enrollmentID
        except:
            enrollment_id = None
```

```
         if enrollment_id is None:
                enrollment = Enrollment_T(student_id=student_id[i],
section_id=section_id, semester=semester, year=year)
                enrollment.save()
                enrollment_id = enrollment.enrollmentID

            for j in range(len(assessment_list)):
                evaluation = Evaluation_T(enrollment_id=enrollment_id,
assessment_id=assessment_list[j], obtainedMarks=coMarks[i][j])
                evaluation.save()

        return redirect('dataentry')
```

OUTPUT FORMS




```
SELECT AVG(TotalPlo.PLOpercentage) AS ActualPlo
                FROM (
                SELECT (PLO / TotalComark * 100) AS PLOpercentage
                FROM (
                        SELECT SUM(DISTINCT e.obtainedMarks) AS PLO, SUM(DISTINCT
a.marks) AS TotalCoMark
```

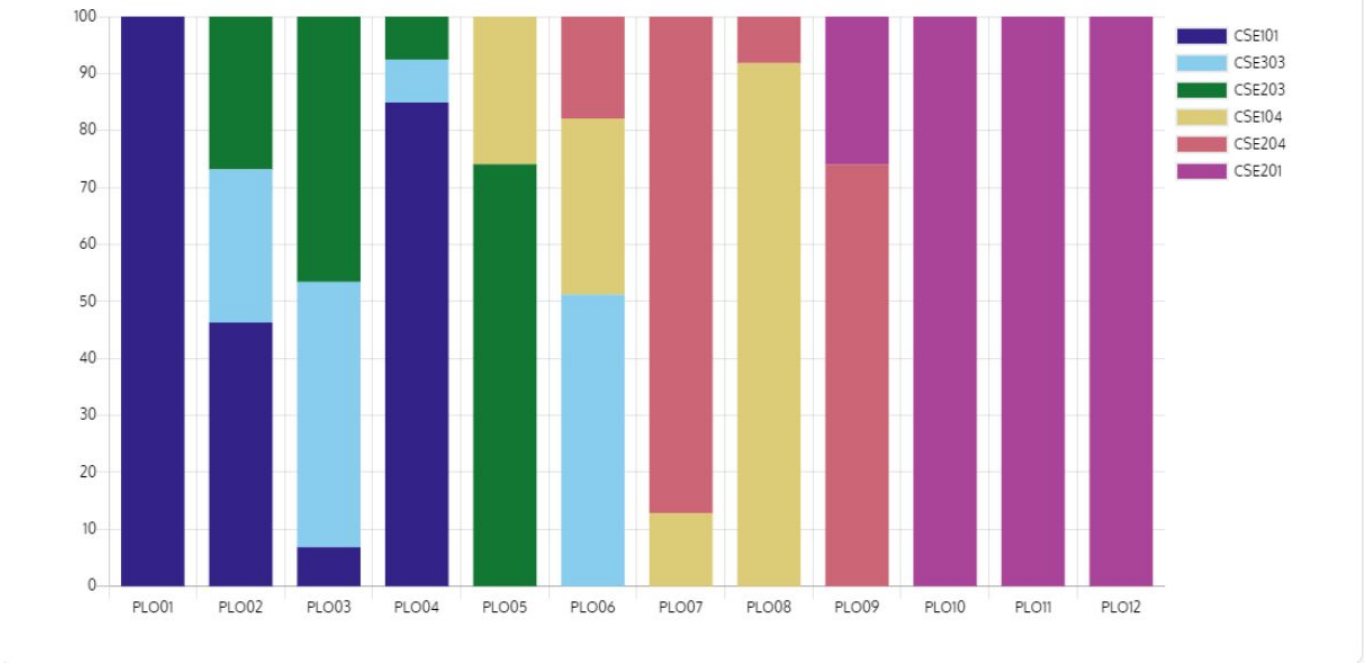
```
                    FROM mainapp_enrollment_t en,
                        mainapp_evaluation_t e,
                        mainapp_assessment_t a,
                        mainapp_co_t c,
                        mainapp_plo_t p
                    WHERE en.student_id = '{}'
                        AND en.enrollmentID = e.enrollment_id
                        AND e.assessment_id = a.assessmentNo
                        AND a.co_id = c.id
                        AND c.plo_id = '{}'
                    GROUP BY en.section_id
                ) ploPer
            ) TotalPlo;
```

Department-wise PLO


```
SELECT AVG(TotalPlo.PLOpercentage) AS ActualPlo
            FROM (
                SELECT (PLO / TotalComark * 100) AS PLOpercentage
                    FROM (
                        SELECT SUM(e.obtainedMarks) AS PLO, SUM(a.marks) AS
TotalCoMark
                        FROM mainapp_enrollment_t en,
                            mainapp_evaluation_t e,
                            mainapp_assessment_t a,
```

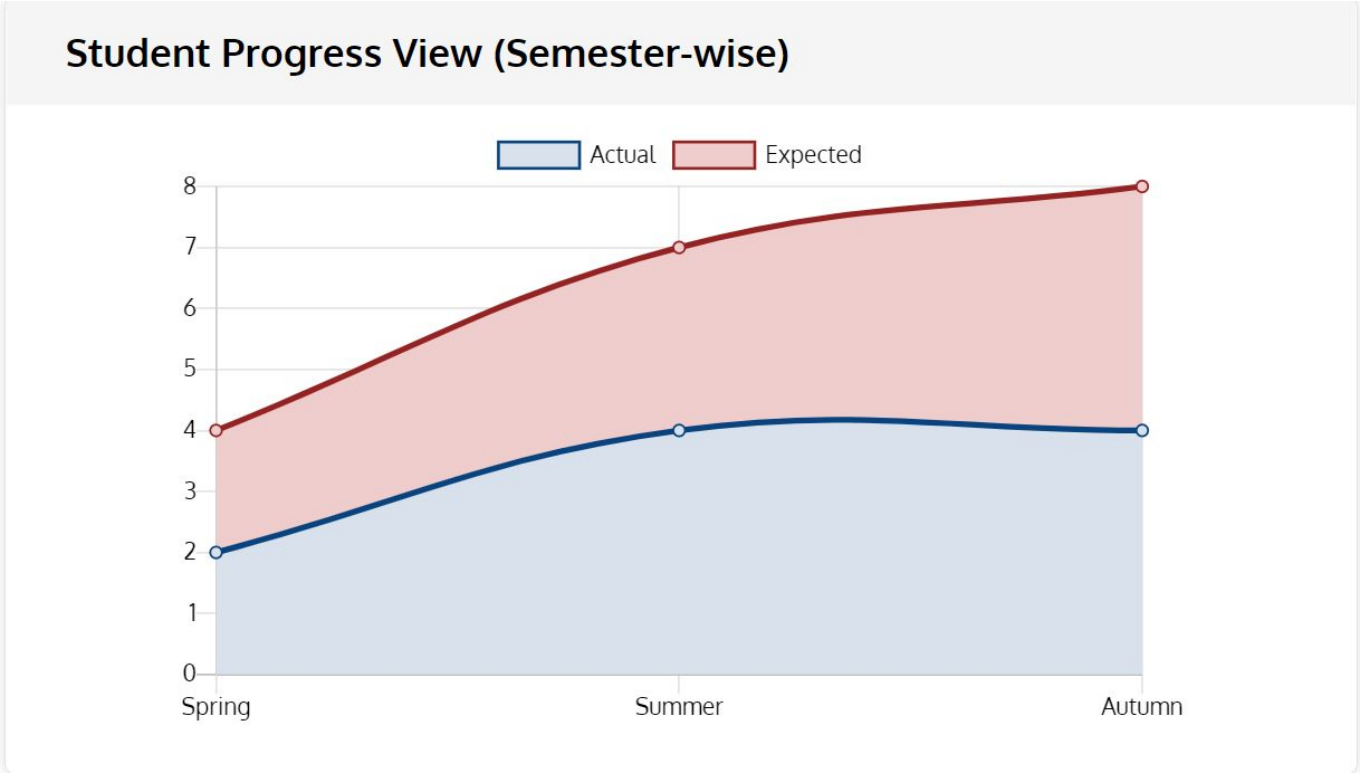
```
                    mainapp_co_t c,
                    mainapp_plo_t p,
                    mainapp_student_t st
                WHERE st.department_id = '{}'
                AND st.studentID = en.student_id
                AND en.enrollmentID = e.enrollment_id
                AND e.assessment_id = a.assessmentNo
                AND a.co_id = c.id
                AND c.plo_id = '{}'
                GROUP BY en.section_id
            ) ploPer
        ) TotalPlo;
```

Course-wise PLO analysis




```
SELECT DISTINCT co.course_id, co.coNo, p.ploNo, (PLO / TotalComark * 100) AS
PLOpercentage
        FROM mainapp_plo_t p, mainapp_co_t co, (
            SELECT DISTINCT c.course_id,c.coNo, c.plo_id, SUM(DISTINCT
e.obtainedMarks) AS PLO, SUM(DISTINCT a.marks) AS TotalCoMark
            FROM mainapp_enrollment_t en,
                mainapp_evaluation_t e,
                mainapp_assessment_t a,
                mainapp_co_t c,
                mainapp_plo_t p
            WHERE en.student_id = '{}'
                AND en.enrollmentID = e.enrollment_id
                AND e.assessment_id = a.assessmentNo
                AND a.co_id = c.id
                AND c.plo_id = p.ploNo
```

```
                GROUP BY en.section_id,c.plo_id
                ORDER BY c.plo_id
            ) ploPer
        WHERE co.coNo = ploPer.coNo
            AND p.ploNo = ploPer.plo_id
            AND co.course_id = ploPer.course_id;
```




```
SELECT COUNT(Acheived.ActualPlo) AS PLoacheivedornot
                    FROM(
                        SELECT AVG(TotalPlo.PLOpercentage) AS ActualPlo
                        FROM (
                            SELECT (PLO / TotalComark * 100) AS PLOpercentage
                            FROM (
                                SELECT SUM(DISTINCT e.obtainedMarks) AS PLO,
SUM(DISTINCT a.marks) AS TotalCoMark
                                FROM mainapp_enrollment_t en,
                                        mainapp_evaluation_t e,
                                        mainapp_assessment_t a,
                                        mainapp_co_t c,
                                        mainapp_plo_t p
                                WHERE en.student_id = '{}'
                                    AND en.semester = '{}'
                                    AND en.year = '{}'
                                    AND en.enrollmentID = e.enrollment_id
                                    AND e.assessment_id = a.assessmentNo
                                    AND a.co_id = c.id
                                    AND c.plo_id = '{}'
                                GROUP BY en.semester
                            ) ploPer
```

```
                    ) TotalPlo
                ) Acheived
                WHERE Acheived.ActualPlo >=40;


                SELECT COUNT(Acheived.ActualPlo) AS PLoacheivedornot
                FROM(
                    SELECT AVG(TotalPlo.PLOpercentage) AS ActualPlo
                    FROM (
                        SELECT (PLO / TotalComark * 100) AS PLOpercentage
                        FROM (
                            SELECT SUM(DISTINCT e.obtainedMarks) AS PLO,
SUM(DISTINCT a.marks) AS TotalCoMark
                            FROM mainapp_enrollment_t en,
                                    mainapp_evaluation_t e,
                                    mainapp_assessment_t a,
                                    mainapp_co_t c,
                                    mainapp_plo_t p
                            WHERE en.student_id = '{}'
                                AND en.semester = '{}'
                                AND en.year = '{}'
                                AND en.enrollmentID = e.enrollment_id
                                AND e.assessment_id = a.assessmentNo
                                AND a.co_id = c.id
                                AND c.plo_id = '{}'
                            GROUP BY en.semester
                        ) ploPer
                    ) TotalPlo
                ) Acheived
```

PLO Achievement Table

COURSE	PLO01	PLO02	PLO03	PLO04	PLO05	PLO06	PLO07	PLO08	PLO09	PLO10	PLO11	PLO12
CSE101	31.5%	54.3%	8.0%	90.0%	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
CSE303	N/A	31.5%	54.3%	8.0%	N/A	90.0%	N/A	N/A	N/A	N/A	N/A	N/A
CSE203	N/A	31.5%	54.3%	8.0%	90.0%	N/A	N/A	N/A	N/A	N/A	N/A	N/A
CSE104	N/A	N/A	N/A	N/A	31.5%	54.3%	8.0%	90.0%	N/A	N/A	N/A	N/A
CSE204	N/A	N/A	N/A	N/A	N/A	31.5%	54.3%	8.0%	90.0%	N/A	N/A	N/A
CSE201	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	31.5%	54.3%	8.0%	90.0%

```
SELECT DISTINCT co.course_id, co.coNo, p.ploNo, (PLO / TotalComark * 100) AS
PLOpercentage
                FROM mainapp_plo_t p, mainapp_co_t co, (
                    SELECT DISTINCT c.course_id,c.coNo, c.plo_id, SUM(DISTINCT
e.obtainedMarks) AS PLO, SUM(DISTINCT a.marks) AS TotalCoMark
                    FROM mainapp_enrollment_t en,
                        mainapp_evaluation_t e,
                        mainapp_assessment_t a,
                        mainapp_co_t c,
```

```
                    mainapp_plo_t p
                WHERE en.student_id = '{}'
                    AND en.enrollmentID = e.enrollment_id
                    AND e.assessment_id = a.assessmentNo
                    AND a.co_id = c.id
                    AND c.plo_id = p.ploNo
                GROUP BY en.section_id,c.plo_id
                ORDER BY c.plo_id
            ) ploPer
        WHERE co.coNo = ploPer.coNo
            AND p.ploNo = ploPer.plo_id
            AND co.course_id = ploPer.course_id;
```

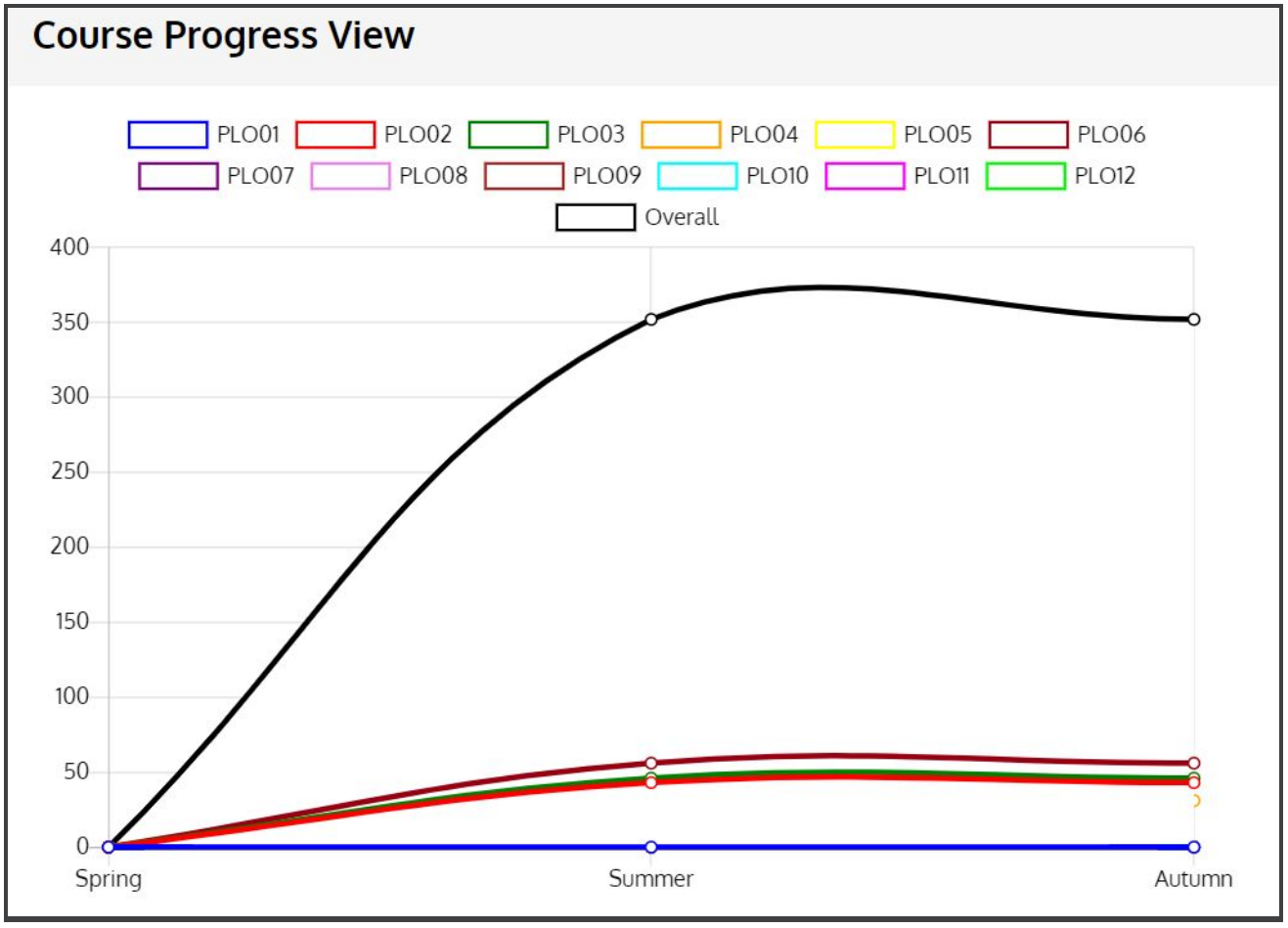
Course Verdict Table

Number of Students: 88

CO	PLO	Successfully Achieved	Successful Percentage (%)	Failed to Achieve	Failed Percentage (%)
1	PLO02	43.0	48.864	45.0	51.136
2	PLO03	46.0	52.273	42.0	47.727
3	PLO04	31.0	35.227	57.0	64.773
4	PLO06	56.0	63.636	32.0	36.364

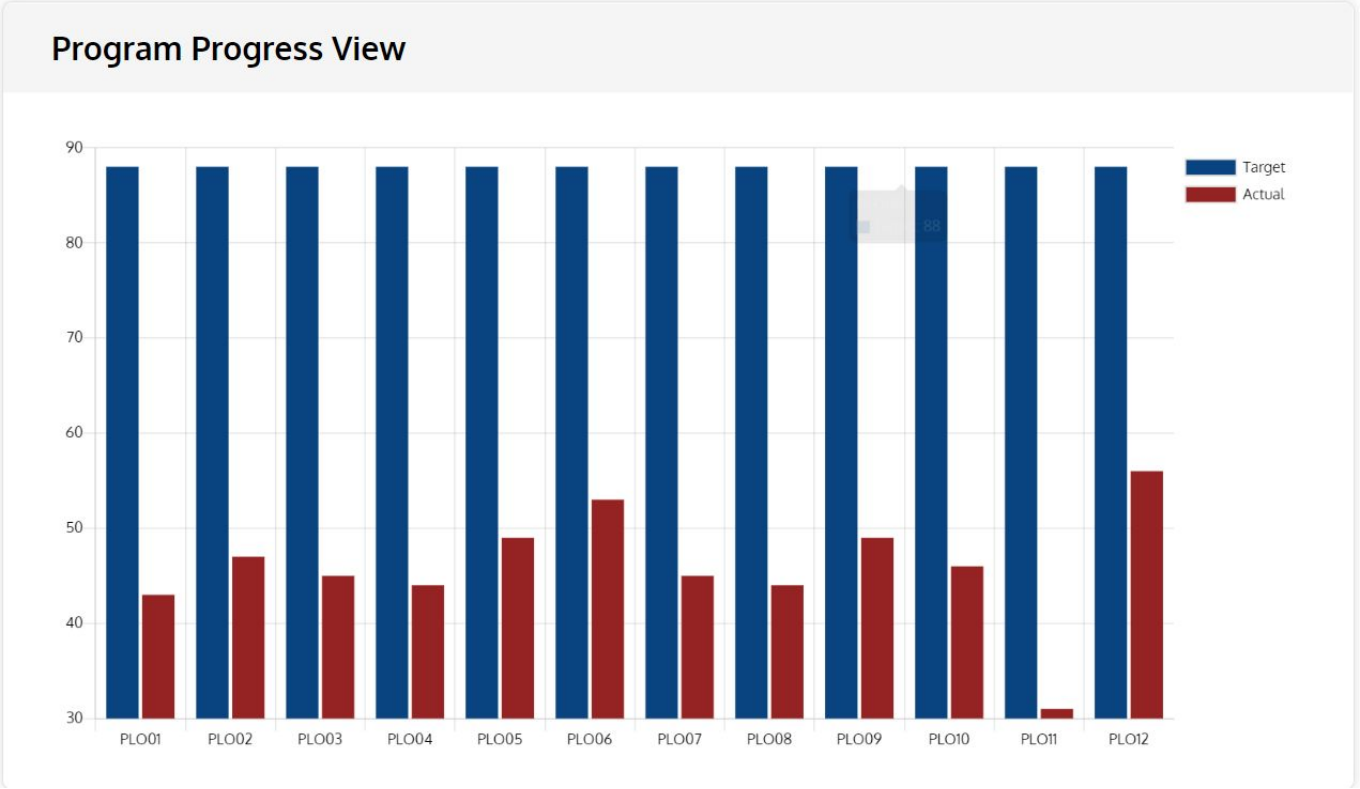
```
SELECT coNo,ploNo,COUNT(TotalPlo.PLOpercentage) AS Acheive
            FROM (
                    SELECT  co.course_id, co.coNo, p.ploNo,(PLO / TotalComark * 100)
AS PLOpercentage
                    FROM mainapp_plo_t p,
                        mainapp_co_t co,
                        (
                            SELECT
en.student_id,c.course_id,c.coNo,c.plo_id,SUM(DISTINCT e.obtainedMarks) AS
PLO,SUM(DISTINCT a.marks)AS TotalCoMark
                            FROM mainapp_enrollment_t en,
                                mainapp_evaluation_t e,
                                mainapp_assessment_t a,
                                mainapp_co_t c,
                                mainapp_plo_t p
                            WHERE en.enrollmentID = e.enrollment_id
                                AND e.assessment_id = a.assessmentNo
                                AND a.co_id = c.id
                                AND c.course_id = '{}'
                                AND c.plo_id = p.ploNo
                            GROUP BY  student_id,c.course_id,c.coNo,p.ploNo
                        ) ploPer
                    WHERE co.coNo = ploPer.coNo
                    AND p.ploNo = ploPer.plo_id
                    AND co.course_id = ploPer.course_id
                GROUP BY student_id,co.course_id,co.coNo,ploNo
                HAVING PLOpercentage >=40
                )TotalPlo
```

```
            GROUP BY course_id,coNo,ploNo;

SELECT coNo,ploNo,COUNT(TotalPlo.PLOpercentage) AS Acheive
            FROM (
                    SELECT  co.course_id, co.coNo, p.ploNo,(PLO / TotalComark * 100)
AS PLOpercentage
                    FROM mainapp_plo_t p,
                        mainapp_co_t co,
                        (
                            SELECT
en.student_id,c.course_id,c.coNo,c.plo_id,SUM(DISTINCT e.obtainedMarks) AS
PLO,SUM(DISTINCT a.marks)AS TotalCoMark
                            FROM mainapp_enrollment_t en,
                                mainapp_evaluation_t e,
                                mainapp_assessment_t a,
                                mainapp_co_t c,
                                mainapp_plo_t p
                            WHERE en.enrollmentID = e.enrollment_id
                                AND e.assessment_id = a.assessmentNo
                                AND a.co_id = c.id
                                AND c.course_id = '{}'
                                AND c.plo_id = p.ploNo
                            GROUP BY  student_id,c.course_id,c.coNo,p.ploNo
                        ) ploPer
                    WHERE co.coNo = ploPer.coNo
                    AND p.ploNo = ploPer.plo_id
                    AND co.course_id = ploPer.course_id
                GROUP BY student_id,co.course_id,co.coNo,ploNo
                )TotalPlo

            GROUP BY course_id,coNo,ploNo;
```



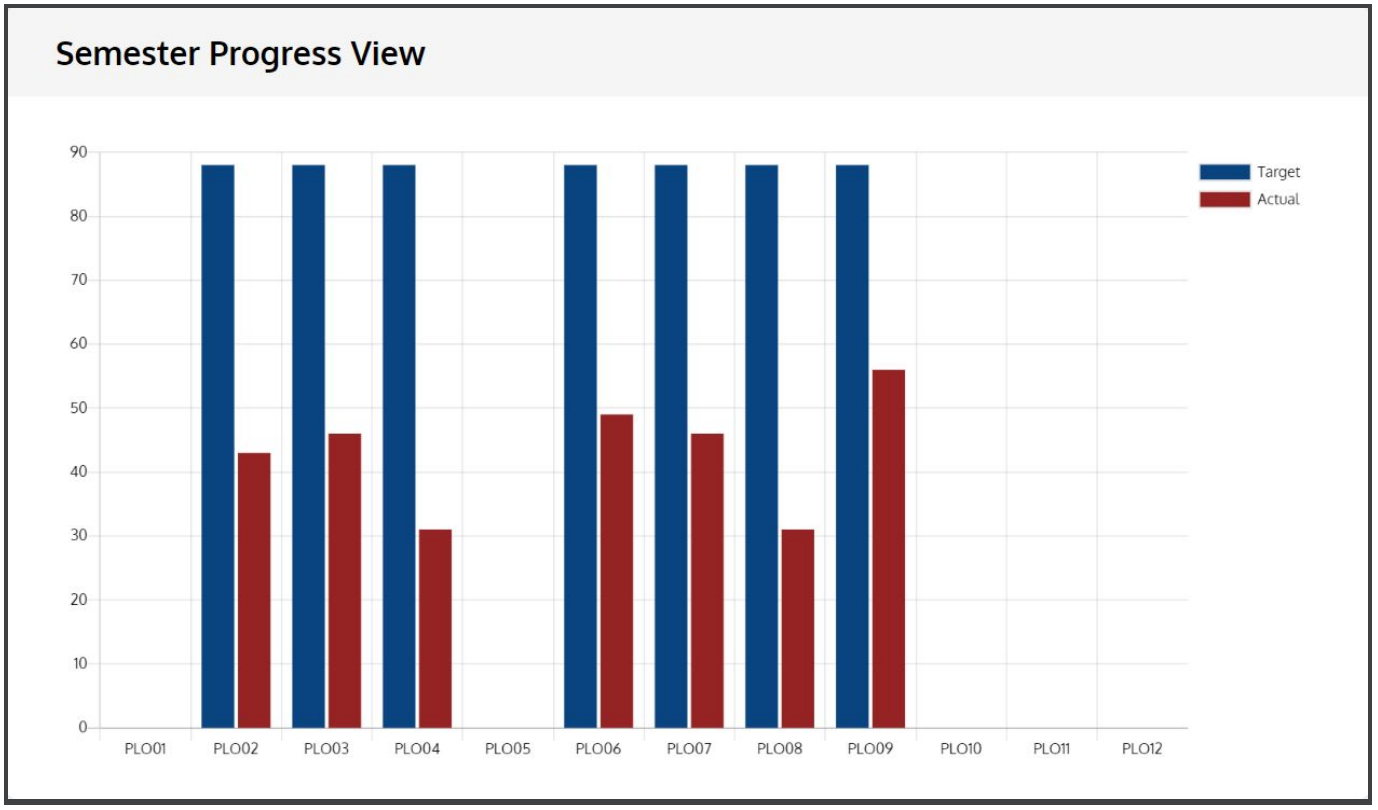

```
SELECT COUNT(Acheived.ActualPlo)
                    FROM (
                            SELECT AVG(TotalPlo.PLOpercentage) AS ActualPlo
                            FROM (
                                    SELECT student_id,(PLO / TotalComark * 100) AS
PLOpercentage
                                    FROM (
                                            SELECT  en.student_id,SUM(DISTINCT
e.obtainedMarks) AS PLO, SUM(DISTINCT a.marks) AS TotalCoMark
                                            FROM mainapp_enrollment_t en,
                                                    mainapp_evaluation_t e,
                                                    mainapp_assessment_t a,
                                                    mainapp_co_t c,
                                                    mainapp_plo_t p
                                            WHERE en.semester ='{}'
                                                AND en.year = '{}'
                                                AND en.enrollmentID = e.enrollment_id
                                                AND e.assessment_id = a.assessmentNo
                                                AND a.co_id = c.id
                                                AND c.course_id = '{}'
                                                AND c.plo_id = '{}'
                                            GROUP BY en.student_id
                                        ) ploPer
                                GROUP BY student_id
                                ) TotalPlo
```

```
                    GROUP BY student_id
                        ) Acheived
                    WHERE Acheived.ActualPlo >= 40;

                    SELECT COUNT(Acheived.ActualPlo)
                    FROM (
                            SELECT AVG(TotalPlo.PLOpercentage) AS ActualPlo
                            FROM (
                                    SELECT student_id,(PLO / TotalComark * 100) AS
PLOpercentage
                                    FROM (
                                            SELECT  en.student_id,SUM(DISTINCT
e.obtainedMarks) AS PLO, SUM(DISTINCT a.marks) AS TotalCoMark
                                            FROM mainapp_enrollment_t en,
                                                    mainapp_evaluation_t e,
                                                    mainapp_assessment_t a,
                                                    mainapp_co_t c,
                                                    mainapp_plo_t p
                                            WHERE en.semester ='{}'
                                                AND en.year = '{}'
                                                AND en.enrollmentID = e.enrollment_id
                                                AND e.assessment_id = a.assessmentNo
                                                AND a.co_id = c.id
                                                AND c.course_id = '{}'
                                                AND c.plo_id = '{}'
                                            GROUP BY en.student_id
                                        ) ploPer
                                GROUP BY student_id
                                ) TotalPlo
                    GROUP BY student_id
```




```
SELECT COUNT(Acheived.ActualPlo)
                    FROM (
                        SELECT AVG(TotalPlo.PLOpercentage) AS ActualPlo
                        FROM (
                                SELECT student_id,(PLO / TotalComark * 100) AS
PLOpercentage
                                FROM (
                                        SELECT  en.student_id,SUM(DISTINCT
e.obtainedMarks) AS PLO, SUM(DISTINCT a.marks) AS TotalCoMark
                                        FROM mainapp_enrollment_t en,
                                                mainapp_evaluation_t e,
                                                mainapp_assessment_t a,
                                                mainapp_co_t c,
                                                mainapp_plo_t p,
                                                mainapp_program_t pr
                                        WHERE p.program_id = pr.programID
                                            AND pr.programID = '{}'
                                            AND en.enrollmentID = e.enrollment_id
                                            AND e.assessment_id = a.assessmentNo
                                            AND a.co_id = c.id
                                            AND c.plo_id = '{}'
                                        GROUP BY en.student_id
                                    ) ploPer
                            GROUP BY student_id
                            ) TotalPlo
                GROUP BY student_id
```

```
                ) Acheived
            WHERE Acheived.ActualPlo >= 40;

SELECT COUNT(Acheived.ActualPlo)
            FROM (
                    SELECT AVG(TotalPlo.PLOpercentage) AS ActualPlo
                    FROM (
                            SELECT student_id,(PLO / TotalComark * 100) AS
PLOpercentage
                            FROM (
                                    SELECT  en.student_id,SUM(DISTINCT
e.obtainedMarks) AS PLO, SUM(DISTINCT a.marks) AS TotalCoMark
                                    FROM mainapp_enrollment_t en,
                                            mainapp_evaluation_t e,
                                            mainapp_assessment_t a,
                                            mainapp_co_t c,
                                            mainapp_plo_t p,
                                            mainapp_program_t pr
                                    WHERE p.program_id = pr.programID
                                        AND pr.programID = '{}'
                                        AND en.enrollmentID = e.enrollment_id
                                        AND e.assessment_id = a.assessmentNo
                                        AND a.co_id = c.id
                                        AND c.plo_id = '{}'
                                    GROUP BY en.student_id
                                ) ploPer
                            GROUP BY student_id
                            ) TotalPlo
            GROUP BY student_id
                ) Acheived
```




```
SELECT COUNT(Acheived.ActualPlo)
                    FROM (
                            SELECT AVG(TotalPlo.PLOpercentage) AS ActualPlo
                            FROM (
                                    SELECT student_id,(PLO / TotalComark * 100) AS
PLOpercentage
                                    FROM (
                                            SELECT  en.student_id,SUM(
e.obtainedMarks) AS PLO, SUM( a.marks) AS TotalCoMark
                                            FROM mainapp_enrollment_t en,
                                                    mainapp_evaluation_t e,
                                                    mainapp_assessment_t a,
                                                    mainapp_co_t c,
                                                    mainapp_plo_t p
                                            WHERE en.enrollmentID = e.enrollment_id
                                                AND en.semester = '{}'
                                                AND en.year = '{}'
                                                AND e.assessment_id = a.assessmentNo
                                                AND a.co_id = c.id
                                                AND c.plo_id = '{}'
                                            GROUP BY en.student_id
                                        ) ploPer
                                GROUP BY student_id
                                ) TotalPlo
                    GROUP BY student_id
                        ) Acheived
                    WHERE Acheived.ActualPlo >= 40;

SELECT COUNT(Acheived.ActualPlo)
                    FROM (
```

```
                        SELECT AVG(TotalPlo.PLOpercentage) AS ActualPlo
                        FROM (
                                SELECT student_id,(PLO / TotalComark * 100) AS
PLOpercentage
                                FROM (
                                        SELECT  en.student_id,SUM(
e.obtainedMarks) AS PLO, SUM( a.marks) AS TotalCoMark
                                        FROM mainapp_enrollment_t en,
                                                mainapp_evaluation_t e,
                                                mainapp_assessment_t a,
                                                mainapp_co_t c,
                                                mainapp_plo_t p
                                        WHERE en.enrollmentID = e.enrollment_id
                                            AND en.semester = '{}'
                                            AND en.year = '{}'
                                            AND e.assessment_id = a.assessmentNo
                                            AND a.co_id = c.id
                                            AND c.plo_id = '{}'
                                        GROUP BY en.student_id
                                    ) ploPer
                            GROUP BY student_id
                            ) TotalPlo
                GROUP BY student_id
                    ) Acheived
```

CHAPTER 5

CONCLUSION

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

PROBLEM AND SOLUTION

1. The limited amount of marksheets and info provided on students and faculties, we had to limit a lot of our calculations and working. If provided with more resources and data to work with, we could believe we could have achieved much more reliable and accurate results, representations and predictions.

2. The bounded timeframe of the semester has hindered our ability to achieve the full potential of this software. We believe we have created the best possible software from the limited resources and time provided, and hope to come up with improvements with better analysis when allowed more time.

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

We believe that we have designed, built and implemented the best possible version of the idea we had for our SPM software. Through the proper usage of this software, we are hopeful to achieve a drastically improved quality of education that universities provide. This software is serviceable to students who want to improve themselves as better and more competent scholars, for faculties to keep better track of their students and improve their teaching methods accordingly, and for the members of the institution to better regulate their resources.