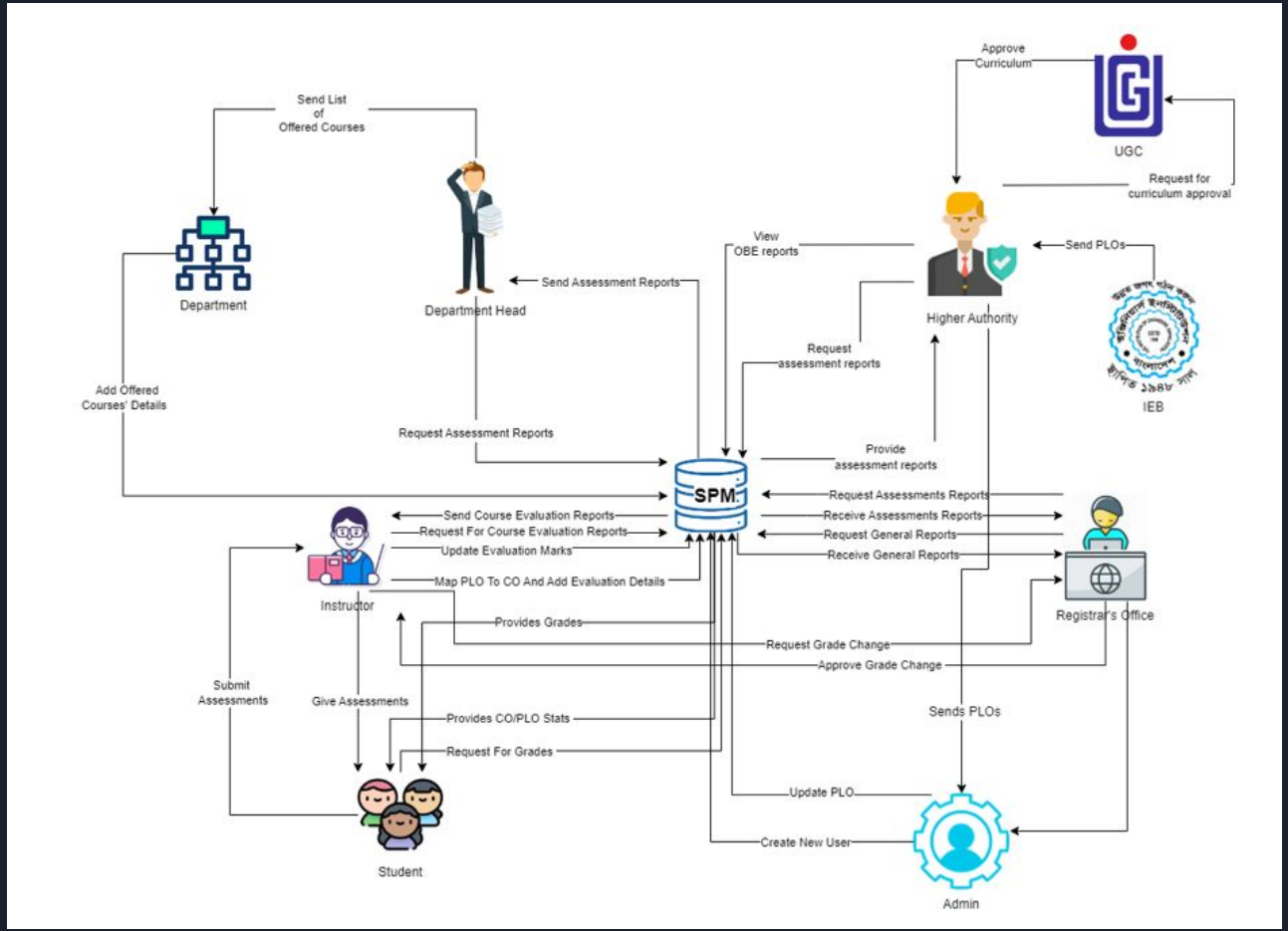
CSE303 Project Presentation

Done by

Faiza Omar Arpita	2010319
Injamam Ul Haque	2021397
Istiaq Ahmed	2021462
Jaima Jahan Khan	2030183
Showrov Mallick	2022185
Syed Niaz Mohtasim	2021607

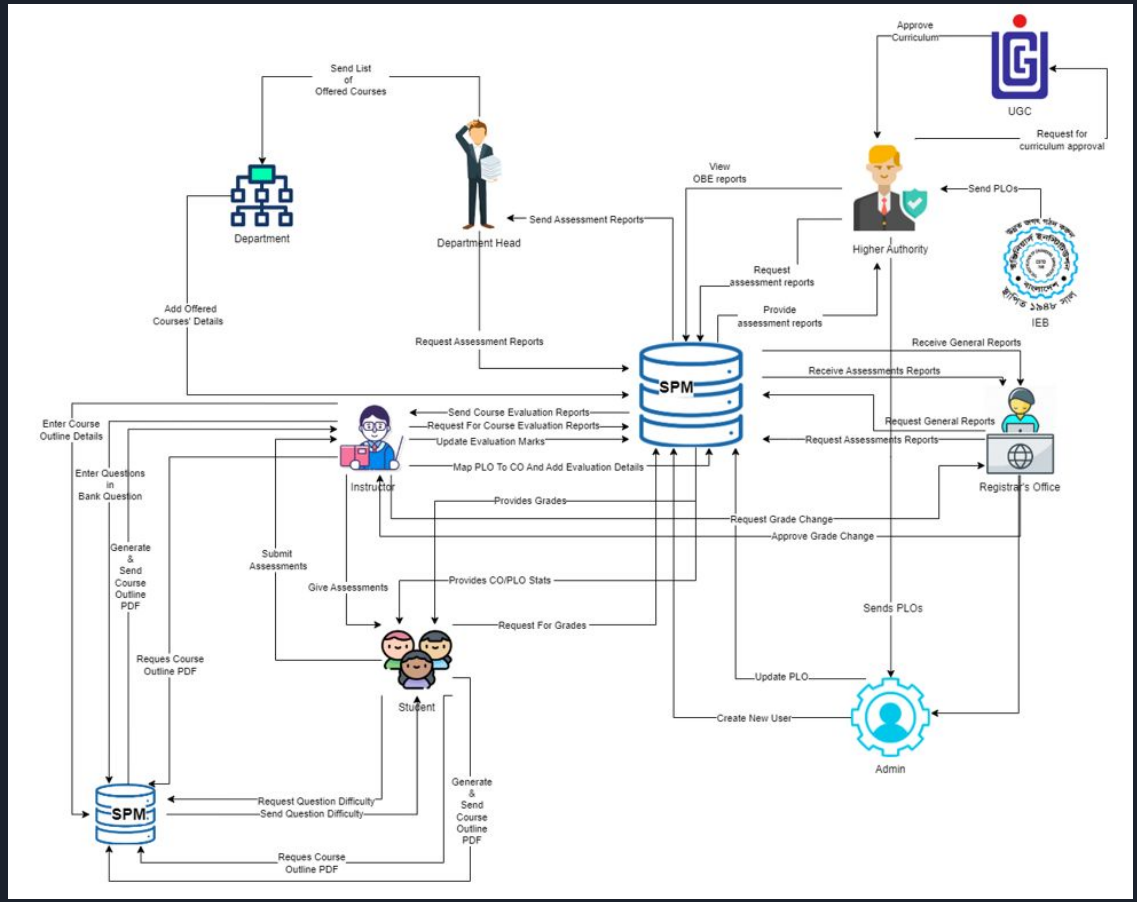
Rich Picture (AS-IS)



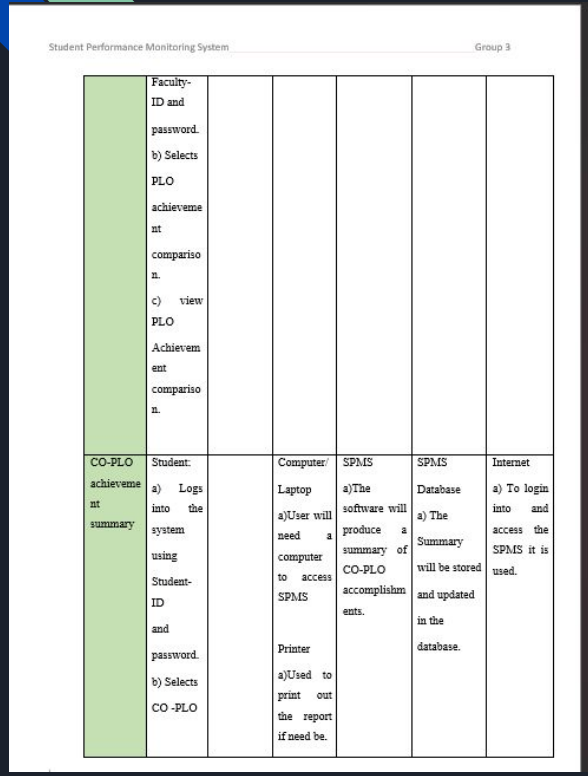
Problem Analysis

Process Name	Stakeholders	Concerns (Problems)	Analysis (Reason of the Problem)	Proposed Solution
Data Entry	A. Weather station B. Forest Ministry Representative C. Forest Ministry D. City Corporation Representative	1. Time Consuming 2. Too many entities involved 3. No direct way to entry data.	1. Data has to be collected and then sent from one source to another and then another source for verification before entry. 2. There is no process for data collectors to directly entry data into the system and to know whether their sensors are working perfectly.	1. Make the process more streamlined, no need to involve so many entities. 2. Create a form through which data can be entered by data collectors directly.
Data Verification	A. City Corporation Representative	1. Manual Checking 2. Lack of relevant communication.	1. Data has to be collected and then sent from one source to another and then another source for verification before entry. 2. There is no process for data collectors to directly entry data into the system and to know whether their sensors are working perfectly.	1. This verification can be done automatically by the system through checking software. 2. Any issues or changes in data or sensors can be caught and can be informed to the data collectors through reports.
Generate reports	A. Forest Ministry B. City Corporation C. Weather stations D. Users	1. No way to edit templates of reports. 2. Reports do not have information regarding data verification.	1. There is no way to make changes or additions to the reports in case more information is asked to be represented on the report. 2. After data verification, if there is problem with sensors there is no way to inform data collectors.	1. Allow option for city corporation to make changes to template of reports as required. 2. Include information about data verification and sensors to the reports to be downloaded by data collectors.
Propose changes to system	A. Forest Ministry B. City Corporation Representative C. City Corporation	1. Time Consuming. 2. Too many entities involved	1. Information about changes has to be proposed by ministry and then sent to representative and then to city corporation for actually implementing changes, very slow process.	1. Make the process more streamlined, no need to involve so many entities. 2. Make a comment box in the system where city corporation can directly send in proposed changes.

Rich Picture (TO-BE)

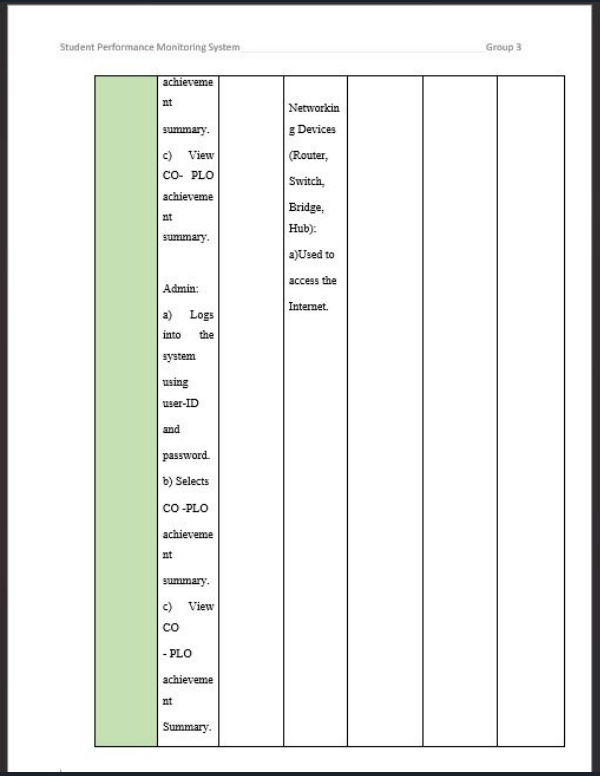


Six Element Analysis - TO BE System



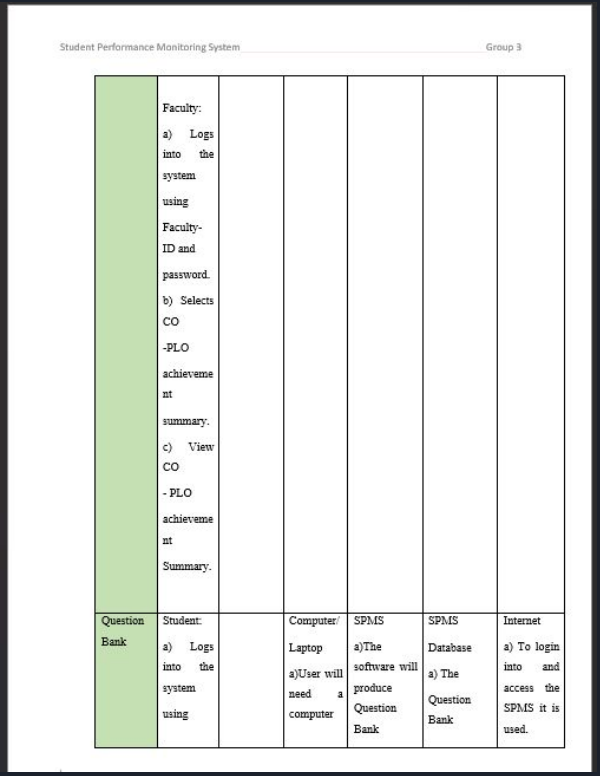
Student Performance Monitoring System Group 3

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



Student Performance Monitoring System Group 3

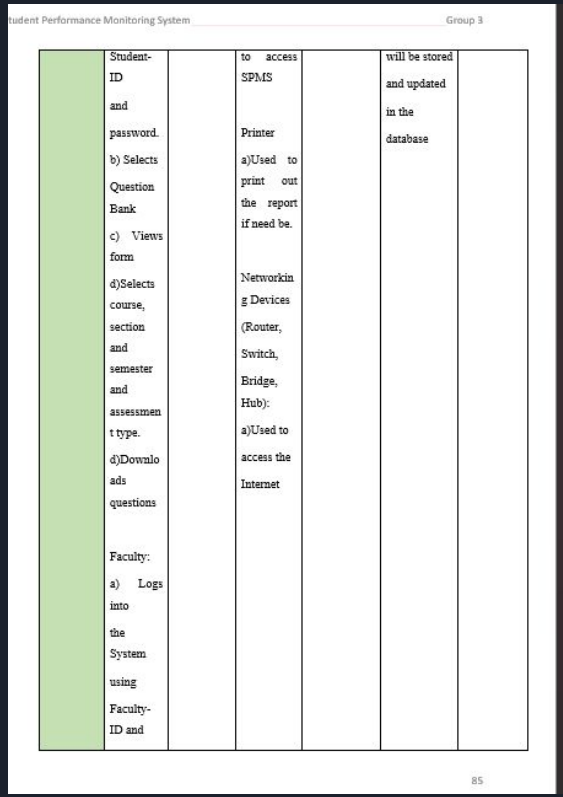
	achievement summary. c) View CO- PLO achievement summary. Admin: a) Logs into the system using user-ID and password. b) Selects CO -PLO achievement summary. c) View CO - PLO achievement Summary.		Networking Devices (Router, Switch, Bridge, Hub): a)Used to access the Internet.			



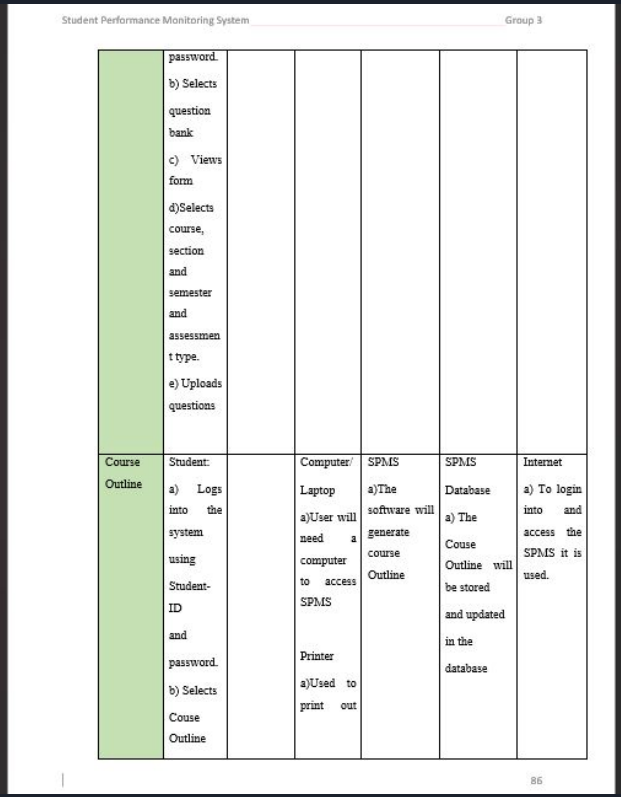
Student Performance Monitoring System Group 3

	Faculty: a) Logs into the system using Faculty-ID and password. b) Selects CO -PLO achievement summary. c) View CO - PLO achievement Summary.					
Question Bank	Student: a) Logs into the system using		Computer/ Laptop a)User will need a computer	SPMS a)The software will produce Question Bank	SPMS Database a) The Question Bank	Internet a) To login into and access the SPMS it is used.

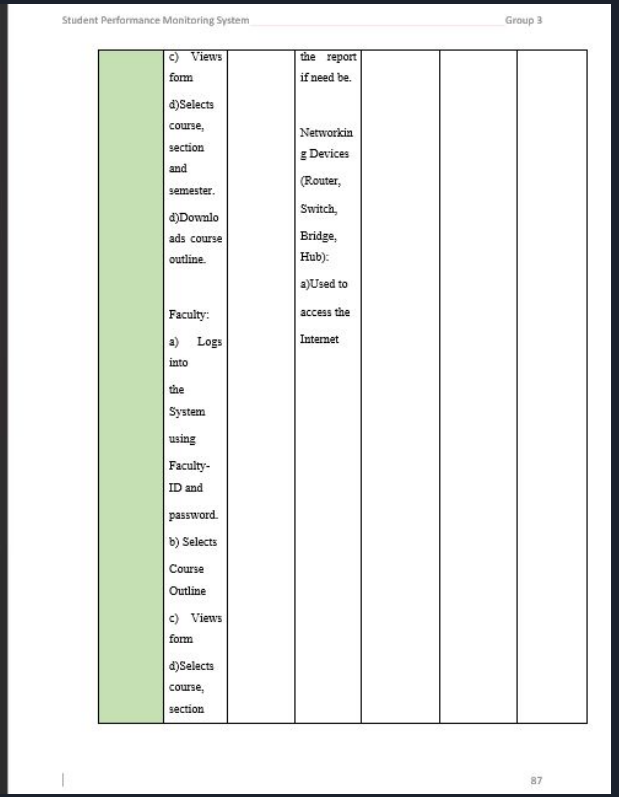
Continued



	Student-ID and password. b) Selects Question Bank c) Views form d)Selects course, section and semester and assessment type. d)Downloads questions Faculty: a) Logs into the System using Faculty-ID and		to access SPMS Printer a)Used to print out the report if need be. Networking Devices (Router, Switch, Bridge, Hub): a)Used to access the Internet		will be stored and updated in the database	



	password. b) Selects question bank c) Views form d)Selects course, section and semester and assessment type. e) Uploads questions					
Course Outline	Student: a) Logs into the system using Student-ID and password. b) Selects Couse Outline		Computer/ Laptop a)User will need a computer to access SPMS Printer a)Used to print out	SPMS a)The software will generate course Outline	SPMS Database a) The Couse Outline will be stored and updated in the database	Internet a) To login into and access the SPMS it is used.



	c) Views form d)Selects course, section and semester. d)Downloads course outline. Faculty: a) Logs into the System using Faculty-ID and password. b) Selects Course Outline c) Views form d)Selects course, section		the report if need be. Networking Devices (Router, Switch, Bridge, Hub): a)Used to access the Internet			

BPMN (TO BE)

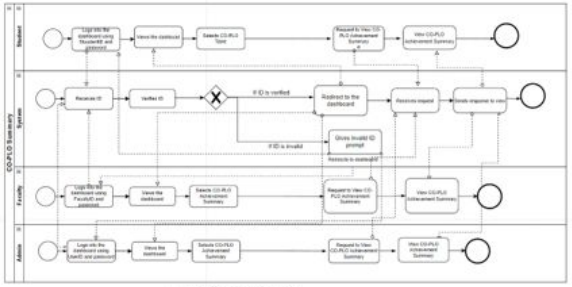


Figure 2.7: CO-PLO Summary (Process)

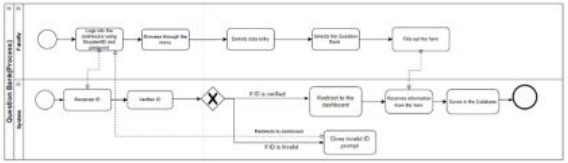


Figure 2.8:Question Bank(Process)

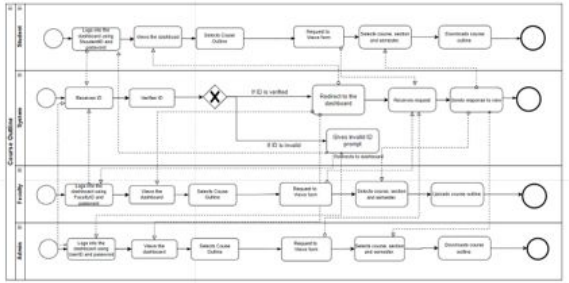
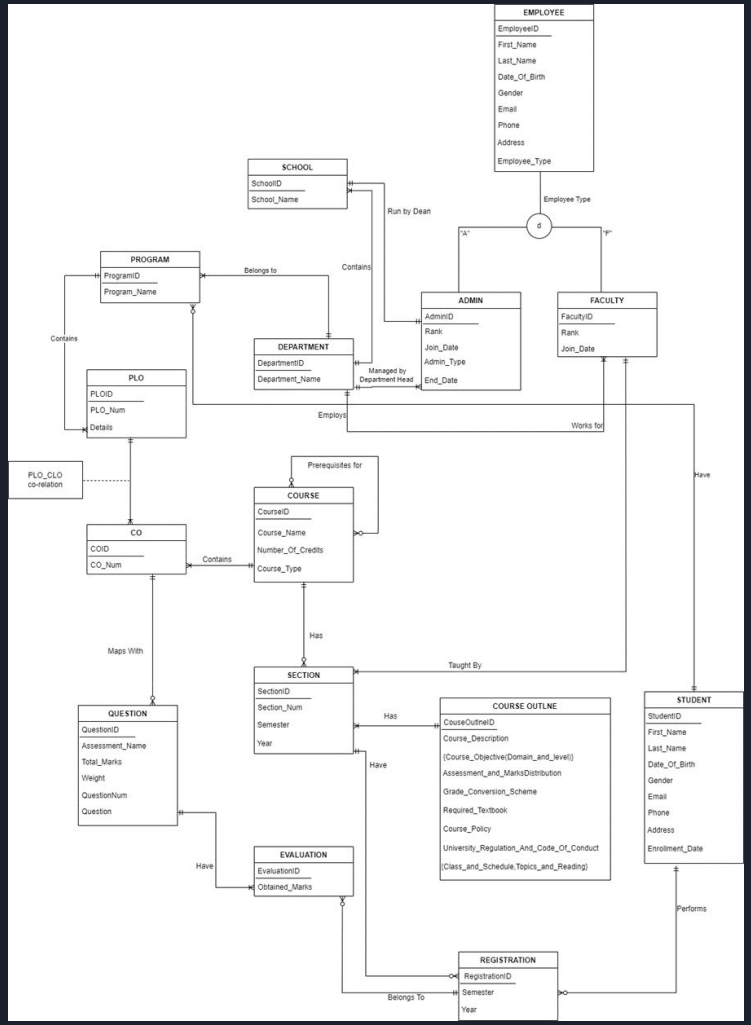
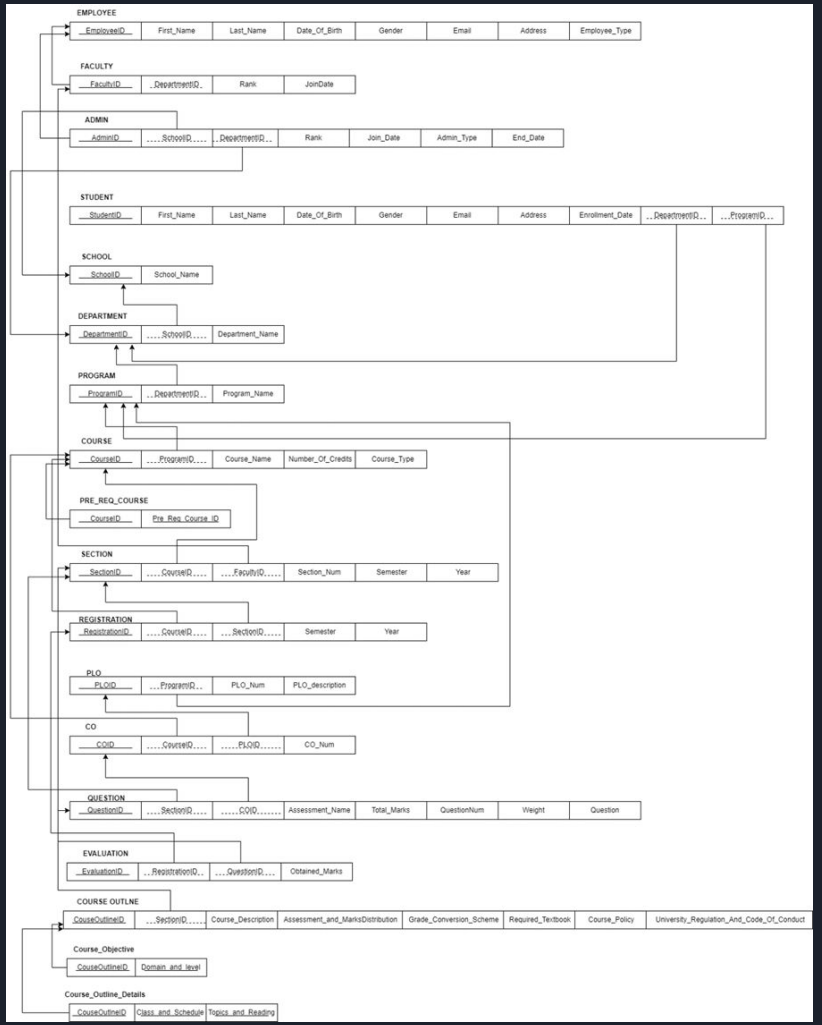


Figure 2.9:Course Outline

EERD


EMPLOYEE
EmployeeID
First_Name
Last_Name
Date_Of_Birth
Gender
Email
Phone
Address
Employee_Type
SCHOOL
SchoolID
School_Name
Run by Dean
Employee Type
d
"A"
"F"
PROGRAM
ProgramID
Program_Name
Belongs to
Contains
ADMIN
AdminID
Rank
Join_Date
Admin_Type
End_Date
FACULTY
FacultyID
Rank
Join_Date
Contains
DEPARTMENT
DepartmentID
Department_Name
Managed by
Department Head
PLO
PLOID
PLO_Num
Details
Employs
Works for
PLO_CLO
co-relation
Prerequisites for
COURSE
CourseID
Course_Name
Number_Of_Credits
Course_Type
Have
CO
COID
CO_Num
Contains
Maps With
Has
Taught By
SECTION
SectionID
Section_Num
Semester
Year
STUDENT
StudentID
First_Name
Last_Name
Date_Of_Birth
Gender
Email
Phone
Address
Enrollment_Date
QUESTION
QuestionID
Assessment_Name
Total_Marks
Weight
QuestionNum
Question
Has
COURSE OUTLNE
CouseOutlneID
Course_Description
{Course_Objective(Domain_and_level)}
Assessment_and_MarksDistribution
Grade_Conversion_Scheme
Required_Textbook
Course_Policy
University_Regulation_And_Code_Of_Conduct
{Class_and_Schedule,Topics_and_Reading}
Have
EVALUATION
EvaluationID
Obtained_Marks
Have
Performs
REGISTRATION
RegistrationID
Semester
Year
Belongs To

Relational Schema



Normalization

1NF: A relation that has a primary key and in which there are no repeating groups.

1NF

a1	a2	a3	a4	a5	a6	a7	b1	b2	b3	**c1**	c2	c3	c4	c5
d1	d2	d3	d4	d5	d6	d7	d8	e1	e2	f1	f2	g1	g2	h1
h2	h3	h4	h5	i1	i2	i3	i4	j1	j2	j3	k1	k2	k3	l1
l2	l3	m1	m2	m3	m4	m5	m6	**n1**	n2	**o1**	o2	o3	o4	o5
o6	o7	o8	o9	o10										

2NF: A relation in first normal form in which every non-key attribute is fully functionally dependent on the primary key.

2NF

a1	a2	a3	a4	a5	a6	a7

b1	b2	b3	e1	e2	f1	f2	g1	g2	h1	h2	h3	h4	h5
j1	i2	i3	i4	j1	j2	j3	k1	k2	k3	l1	l2	l3	m1
m2	m3	m4	m5	m6	**n1**	n2							

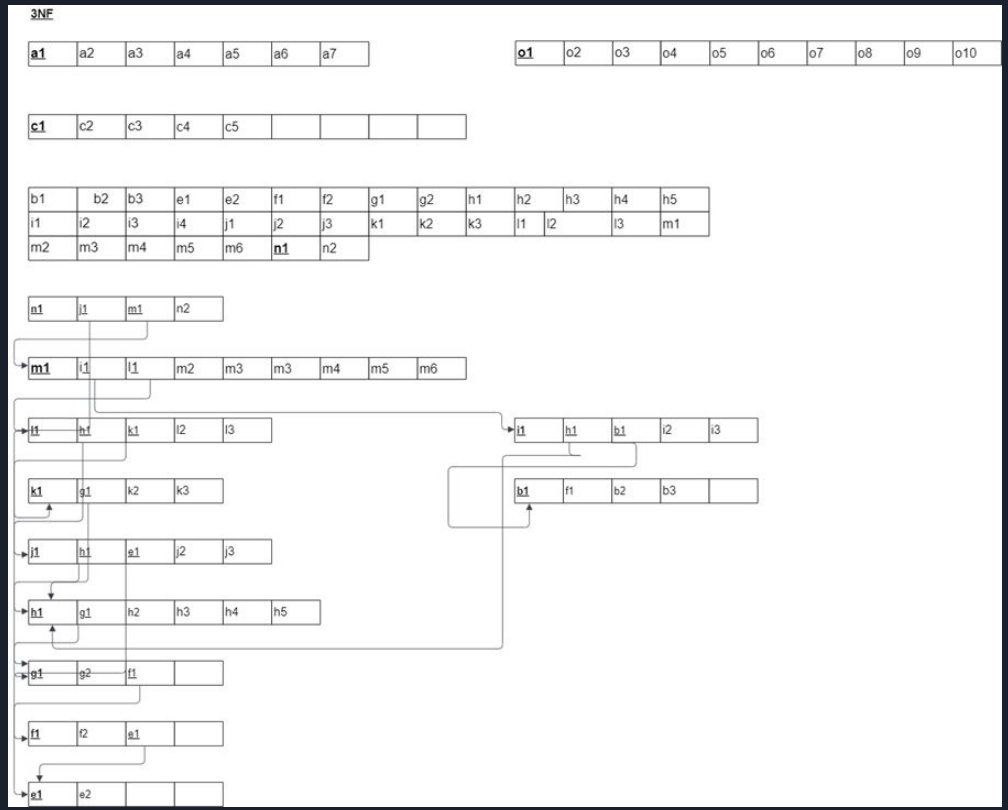
c1	c2	c3	c4	c5				

d1	d2	d3	d4	d5	d6	d7	d8	

o1	o2	o3	o4	o5	o6	o7	o8	o9	o10

Normalization - Continued

3NF: A relation that is in second normal form and has no transitive dependencies.



DATA DICTIONARY

School_T

Name	Data Type	Size	Remarks
cSchoolID	VARCHAR	5	This is the primary key of School. E.g.: "SETS"
cSchoolName	VARCHAR	50	This is the name of the school. E.g.: "School of Engineering, Technology & Science".

Program_T

Name	Data Type	Size	Remarks
cProgramID	INTEGER		This is the primary key for a program. E.g.: "1"
cProgramName	VARCHAR	50	This is the name of the program. E.g.: "Bachelor of Science"

Evaluation_T

Name	Datatype	Size	Remarks
nEvaluationID	INTEGER		This is the Primary Key for Enrollment.
cObtainedMarks	NUMBER		This is the obtained marks of the student. E.g.: "24.5"
cQuestionID	INTEGER		This is the foreign key from the assessment table.
nRegistrationID	INTEGER		This is the Foreign Key from Registration table.

Student_T

Name	Data Type	Size	Remarks
nStudentID	INTEGER		This is the primary key for the student table. E.g.: "1921834".
cFirstName	VARCHAR	30	This is the first name of the student. E.g.: "Rakibul".
cLastName	VARCHAR	30	This is the last name of the student. E.g.: "Hasan".
dDateOfBirth	DATE	DD MM YYYY	This is the birth date of the student. E.g.: "21-12-1996".
cGender	VARCHAR	6	This is the gender of the student. E.g.: "Female".

cEmail	VARCHAR	30	This is the email of the student. E.g.: "1921834@iub.edu.bd"
nPhone	NUMERIC	11	This is the phone of the student. E.g.: "01XXXXXXXXX".
cAddress	VARCHAR	50	This is the address of the student. E.g.: "House 1, Road 4, Block D, Bashundhara R/A".
dEnrollmentDate	DATE	DD MM YYYY	This is enrollment date of the student. E.g.: "1-1-2019"
cProgramID	INTEGER		This is the foreign key from the program table. E.g.: "1"
cDepartmentID	VARCHAR	3	This is the foreign key from the Department table. E.g.: "CSE"

Employee_T

Name	Datatype	Size	Remarks
nEmployeeID	INTEGER		This is the primary key for Employee table. E.g.: "1801"
cFirstName	VARCHAR	30	This is the first name of the faculty. E.g.: "Sadita"
cLastName	VARCHAR	30	This is the last name of the faculty. E.g.: "Ahmed"

dDateofbirth	DATE	DD-MM YYYY	This is the date of Birth of the faculty. E.g:01-01-1992
cGender	VARCHAR	6	This is the gender of the faculty. E.g.: "Female"
cEmail	VARCHAR	30	This is the email address of the student. E.g.: "1675231@iub.edu.bd"
nPhone	NUMERIC	11	This is the phone number of the faculty. E.g.: "01292383111"
cAddress	VARCHAR	30	This is the address of the faculty. E.g.: "House 14, Road 21, Sector 11, Baridara,Dhaka, Bangladesh"
cEmployeeType	CHAR	1	This is the type of the employee. E.g.: "F"

Course_T

Name	Datatype	Size	Remarks
cCourseID	VARCHAR	6	This is the Primary Key for the Course. E.g.: "CSE203"
cCourseName	VARCHAR	40	This is the name of the Course. E.g.: "Discreet Mathematics"
nNumOfCredits	INTEGER		This is the number of credits for the Course. E.g.: "3"

cCourseType	VARCHAR	10	This is the type of the Course. E.g.: "Core"
cPLOID	INTEGER		This is the foreign key from the Program Learning Outcome table. E.g.: "PLO1"

Section_T

Name	Datatype	Size	Remarks
nSectionID	INTEGER		This is the Primary Key for Section. E.g.: "1"
nSectionNum	INTEGER		This is the section number. E.g.: "1"
cCourseID	VARCHAR	6	This is the foreign key from the Course table. E.g.: "CSE101"
cSemester	VARCHAR	6	This is the semester of the section. E.g.: "Summer"
cFacultyID	NUMERIC	4	This is the foreign key from Faculty table. E.g.: "1801"
dYear	YEAR	yyyy	This is the year of registration. E.g.: "2019"

Registration_T

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

Question_T

Name	Datatype	Size	Remarks
nQuestionID	INTEGER		This is the Primary Key for Assessment.
cAssessmentName	VARCHAR	30	This is the name of the assessment. E.g.: "Mid"
cTotalMarks	NUMBER		This is the total marks of the assessment. E.g.: "30"
cQuestion	VARCHAR		This is the question for the assessment. E.g.: "What is SQL"
nQuestionNum	INTEGER		This is the question number E.g.: "1,2,3…."

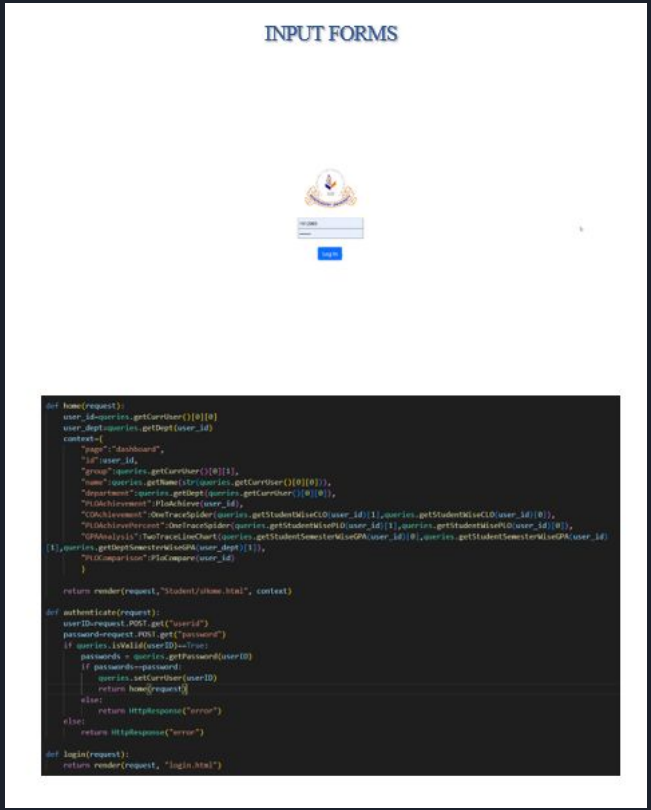
nWeight	INTEGER		This is the percentage range for assessment. E.g.: "Project- 50%, Assessment-50%".
nSectionID	INTEGER		This is the Foreign Key from Section table.
nCOID	INTEGER		This is the Foreign Key from the Course Outcome table.

Faculty_T

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

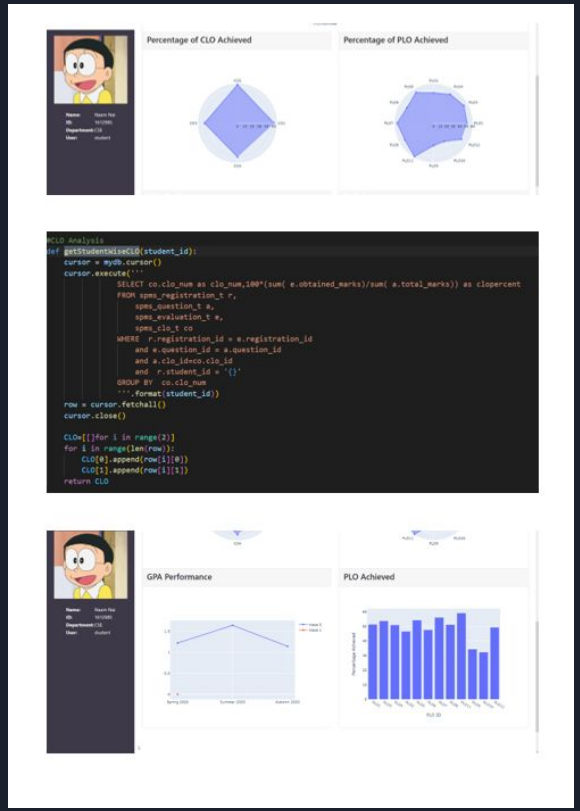
Admin_T

Input Forms

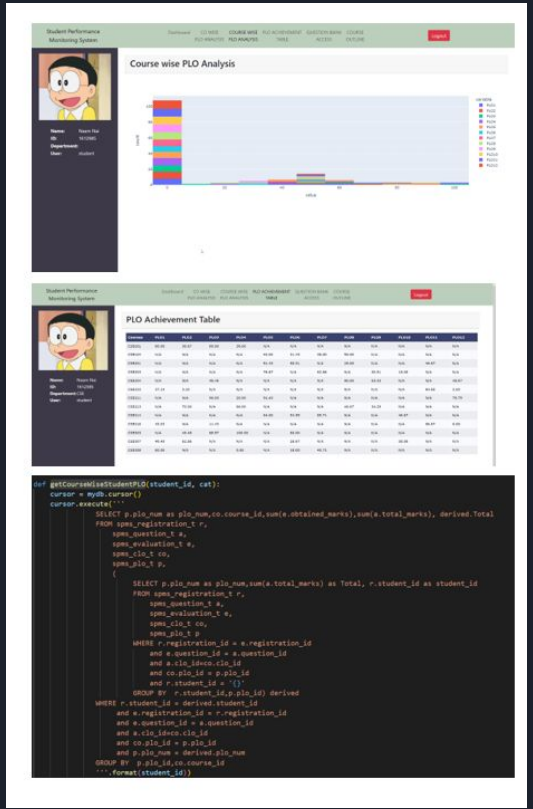



```
def home(request):
    user_id=queries.getCurrUser()[0][0]
    user_dept=queries.getDept(user_id)
    context={
        "page":"dashboard",
        "id":user_id,
        "group":queries.getCurrUser()[0][1],
        "name":queries.getName(str(queries.getCurrUser()[0][0])),
        "department":queries.getDept(queries.getCurrUser()[0][0]),
        "PLOAchievement":PloAchieve(user_id),
        "COAchievement":OneTraceSpider(queries.getStudentWiseCLO(user_id)[1],queries.getStudentWiseCLO(user_id)[0]),
        "PLOAchivePercent":OneTraceSpider(queries.getStudentWisePLO(user_id)[1],queries.getStudentWisePLO(user_id)[0]),
        "GPAAnalysis":TwoTraceLineChart(queries.getStudentSemesterWiseGPA(user_id)[0],queries.getStudentSemesterWiseGPA(user_id)
[1],queries.getDeptSemesterWiseGPA(user_dept)[1]),
        "PLOComparison":PloCompare(user_id)
        }

    return render(request,"Student/uHome.html", context)

def authenticate(request):
    userID=request.POST.get("userid")
    password=request.POST.get("password")
    if queries.isValid(userID)==True:
        passwords = queries.getPassword(userID)
        if passwords==password:
            queries.setCurrUser(userID)
            return home(request)
        else:
            return HttpResponse("error")
    else:
        return HttpResponse("error")

def login(request):
    return render(request, "login.html")
```

Output Forms



Output Forms



Output Forms

```
cursor.close()
table = []
courses = []

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

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

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

Thank you!