Student Performance Monitoring System

Good Morning!

Group 27
Team Data Armour

Group 27
Team Data Armour

Team Members

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

Background

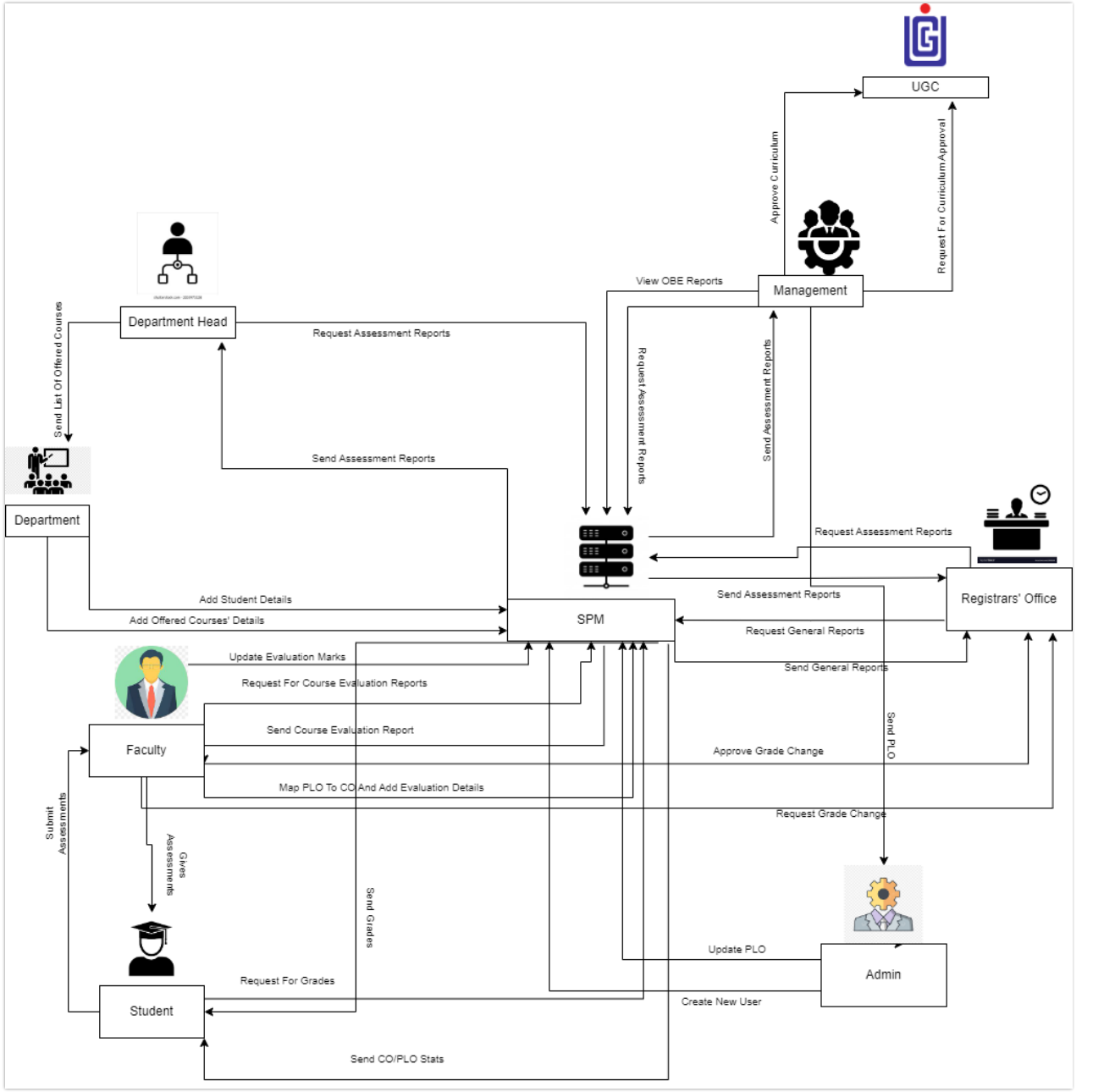
- Our project's goal is to create, develop, and distribute software that, in our opinion, will assist universities worldwide in promoting a more fruitful and efficient method of student evaluation.
- We've introduced the notion of Course Outcomes (COs) and Program Learning Outcomes (PLOs), where each CO is mapped to a PLO, and each PLO represents a particular valuable skill that students are expected to acquire or improve at the conclusion of that course.
- The project will determine whether each student has successfully completed the PLOs that are linked to the COs requirements in order to evaluate them effectively through tools such as spider charts.
- Students can monitor their progress in each area and identify their areas for growth and improvement. Our program also aims to help the institutional bodies, including faculty, administrative, and departmental bodies, track student development, departmental performance, and better distribute and allocate resources.

Objectives And Scope

- Our project aims to develop an interactive, user-friendly program that will serve as a platform for university staff, faculty, and other participants to assist in enhancing the standard of instruction and revolutionizing how we incorporate technology into our education.
- Our approach entails building a Web application called SPMS 2 that makes use of a Relational Database Management System (RDMS) to store, edit, add, and update the data required for tracking student performance as well as for producing and archiving related OBE data, reports, and documents
- We created hypothetical users for the web based SPMS system and made assumptions about their usage patterns and the information and data they would require
- We will create unique user interfaces and login options for various stakeholders who will also be using this system.
- We develop user interfaces that allow all users to quickly access relevant files, tabular data and reports and the use of RDMS offers real time interaction with the relevant data.
- We create a platform through which faculties may work together to create course outline, course reports, mark sheets, assessments, map assessments to COs and PLOs for PLO successes, and keep track of student evaluations for all their courses throughout the semester and upload questions in the question bank for the students

Rich Picture

Existing System



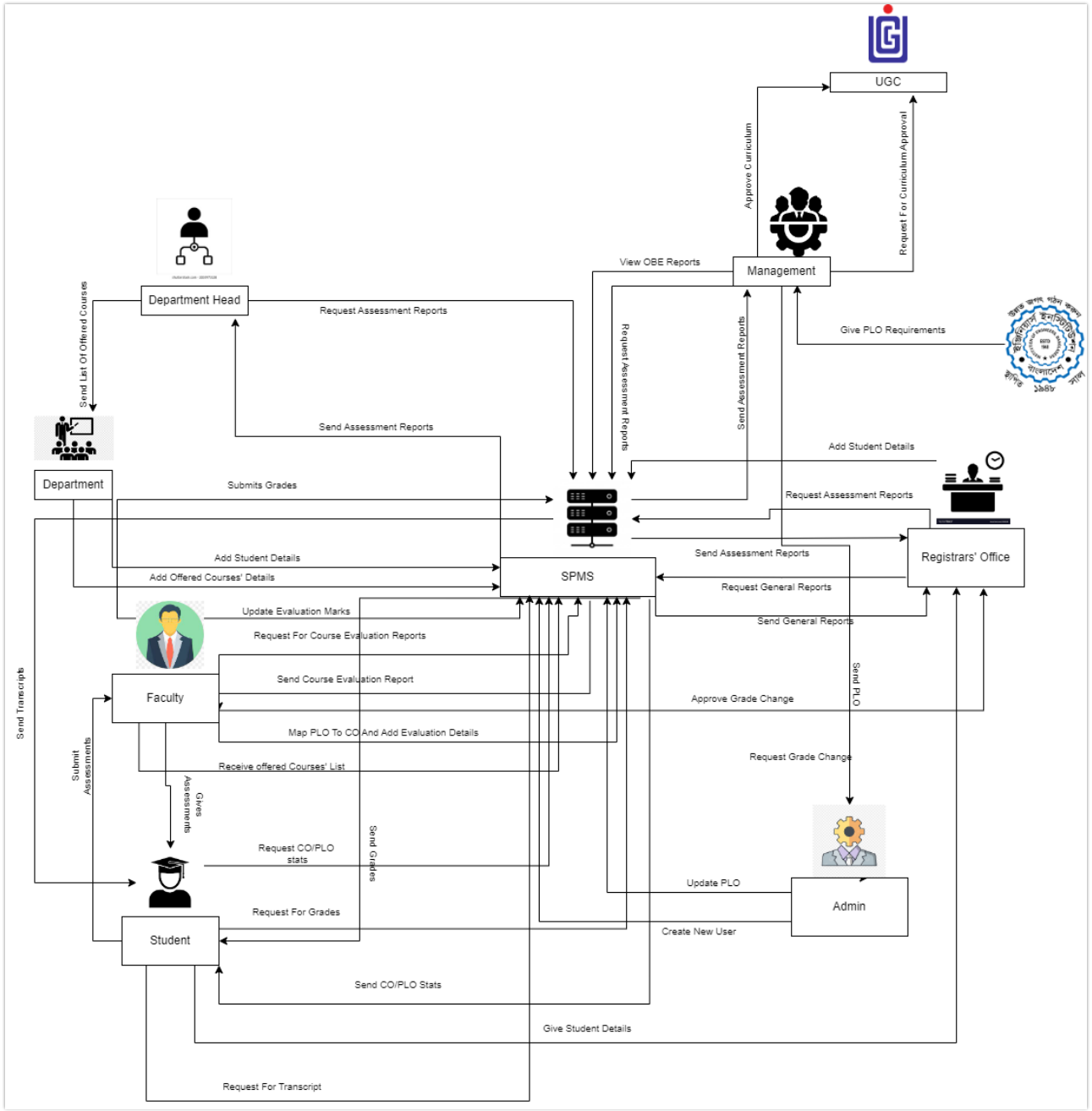
Problem Analysis

EXISTING PROBLEMS & ANALYSIS OF THE PROBLEM

Process Name	Stakeholders	Concerns (Problems)	Analysis (Reason of the Problem)	Proposed Solution
Student Enrollment	1. Student 2. Department Head 3. Registrar's Office 4. Faculty 5. Admin	Comparison of Student who have Enrolled in each Department with respect to a given Time Period/Semester	Student enrolled stat is recorded school-wise, department-wise, and program-wise but was not compared with respect to time period or semesters.	We want to keep the record in the count of students enrolled along with a visual comparison of the student stats as per school-wise, department-wise, program-wise and semester-wise.
Assessments and Grading	1. Faculty 2. Students	1) Condition of Question paper and Answer Script 2) Giving and Receiving Process 3) Unreliable Storage 4) Lack of Visibility of Learning and Question Difficulty 5) No method to Submit Assessments and Grades	1) The question papers and answer script which are being stored physically can get damaged or may get lost. 2) The Process of completing the assessment and giving it to the teacher in person is slow. 3) There may be a shortage of physical space due to increase number of papers.	The question papers and answer scripts can be stored into the database so there is no problem of storage. Once a question is placed inside the question bank, the question gets its difficulty level and domain of learning automatically assigned. Online submission of
			4) Need to find the domain of learning and difficulty of the question manually and that also takes a lot of time. 5) Adding method to Insert Grades and Cos of a course.	assessment saves time as it negates the necessity to submit a physical copy in person. And Adding Method to Submit Grading and CO assessments by importing CSV file.
Course Outline	1. Department 2. Faculty 3. Student	1) Waiting Delay for receiving Necessary Resources 2) Creating a Course Outline	1) The faculty needs to send requests to department and wait for them to send back the necessary materials. 2) It requires a lot of time to create a course outline manually.	A feature can be installed to generate the course outline automatically according to the things the faculty wants to add. It is stored in the database, and it can be downloaded by the stakeholders in a pdf file.
Student Performance based on CGPA	1. Student 2. Department Head 3. Registrar's Office 4. Faculty	Comparison of Student CGPA between Schools, Departments, Programs and Courses	The CGPA of students can only be observed individually but can be compared between different schools, departments, programs, and courses.	A system should be in place which will allow the stakeholders to analyze the CGPA not only individually but also based on different schools, departments, programs, and courses for a given time or semester.
CO-PLO Achievement	1. Student 2. Faculty 3. Admin	1) PLO Achievement of a Student for each Courses 2) Comparison of PLO Achievement within a Department 3) PLO Achievement Rate and Score 4) Reports based on CO-PLO	1) Students are unable to monitor progress of their PLO achievement for respective courses as it is only available to higher authorities and is done manually 2) The PLO and corresponding CO of all courses	A system should be implemented which will record the PLO' and COs in the database which will give easier access to the stakeholders. Comparisons regarding PLO achievements can then be made
			a student does is never compared with cumulatively along with the departmental average performance. 3) PLO achieved versus attempted, and the actual score is done manually which can be extremely time consuming. 4) Reports based on PLO and CO may not be enough to give a clear picture.	automatically which will save time. Charts can then be generated for better analysis.

Rich Picture

TO BE



Six Elements Analysis

TO BE

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

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

Six Elements Analysis (continued)

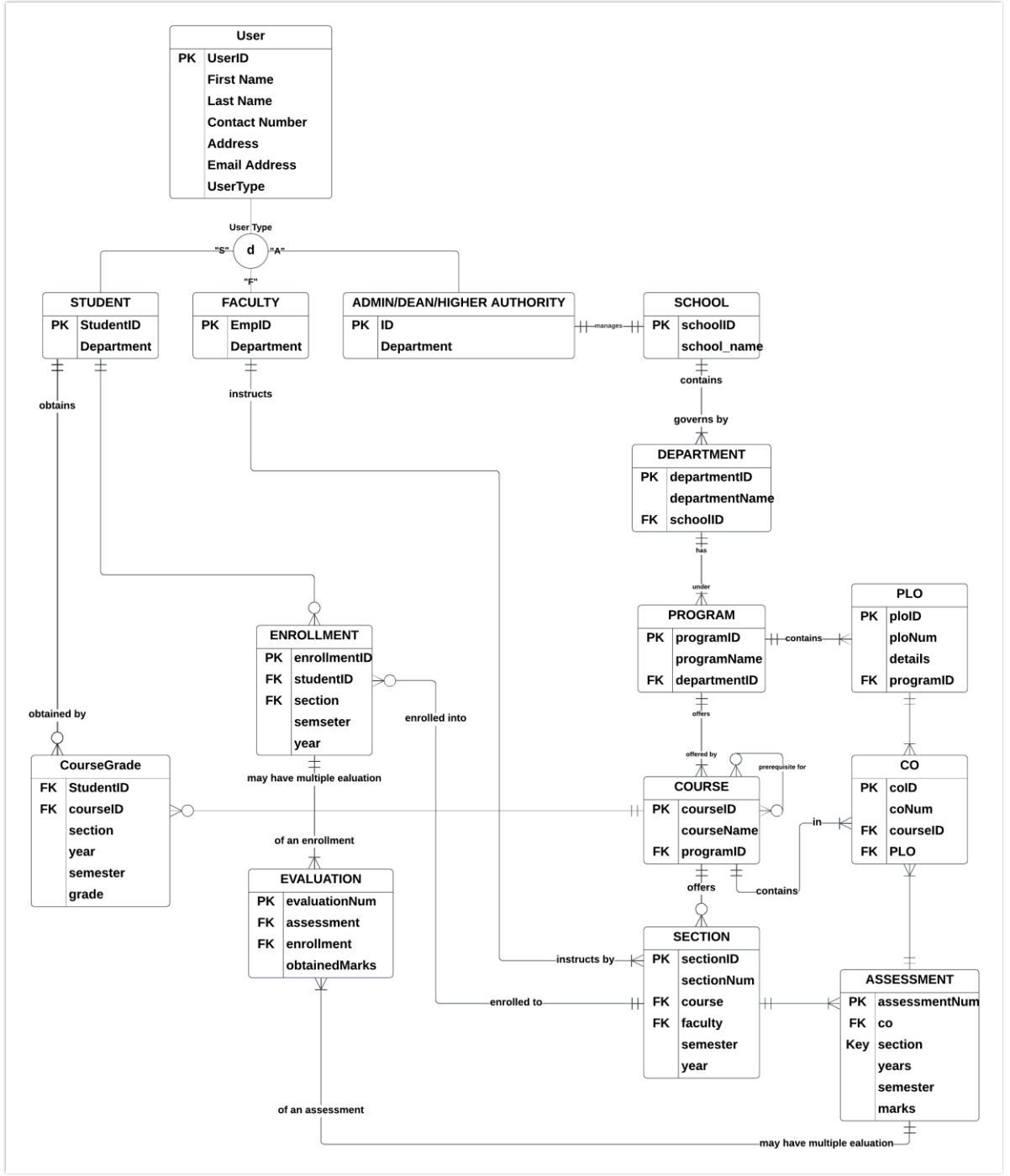
	Admin: a) Logs into the system using user-ID and password. b) Selects CO percentage based on the obtained grades for each course summary. c) View CO percentage based on the obtained grades for each course summary. Faculty: a) Logs into the system using Faculty-ID and password. b) PLO Achievement Analysis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

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

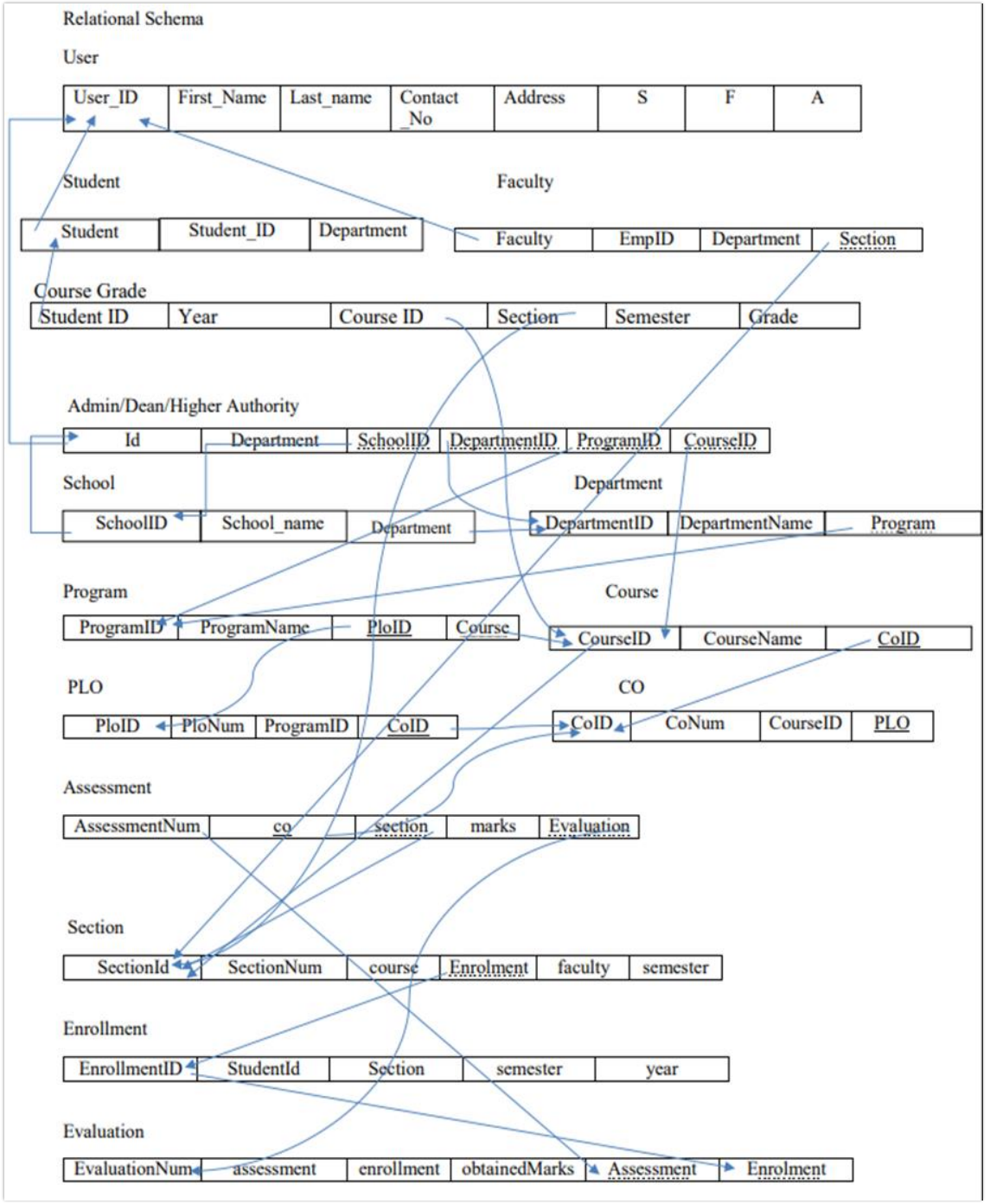
Six Elements Analysis (continued)

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

Entity Relationship Diagram



Relational Schema



Normalization

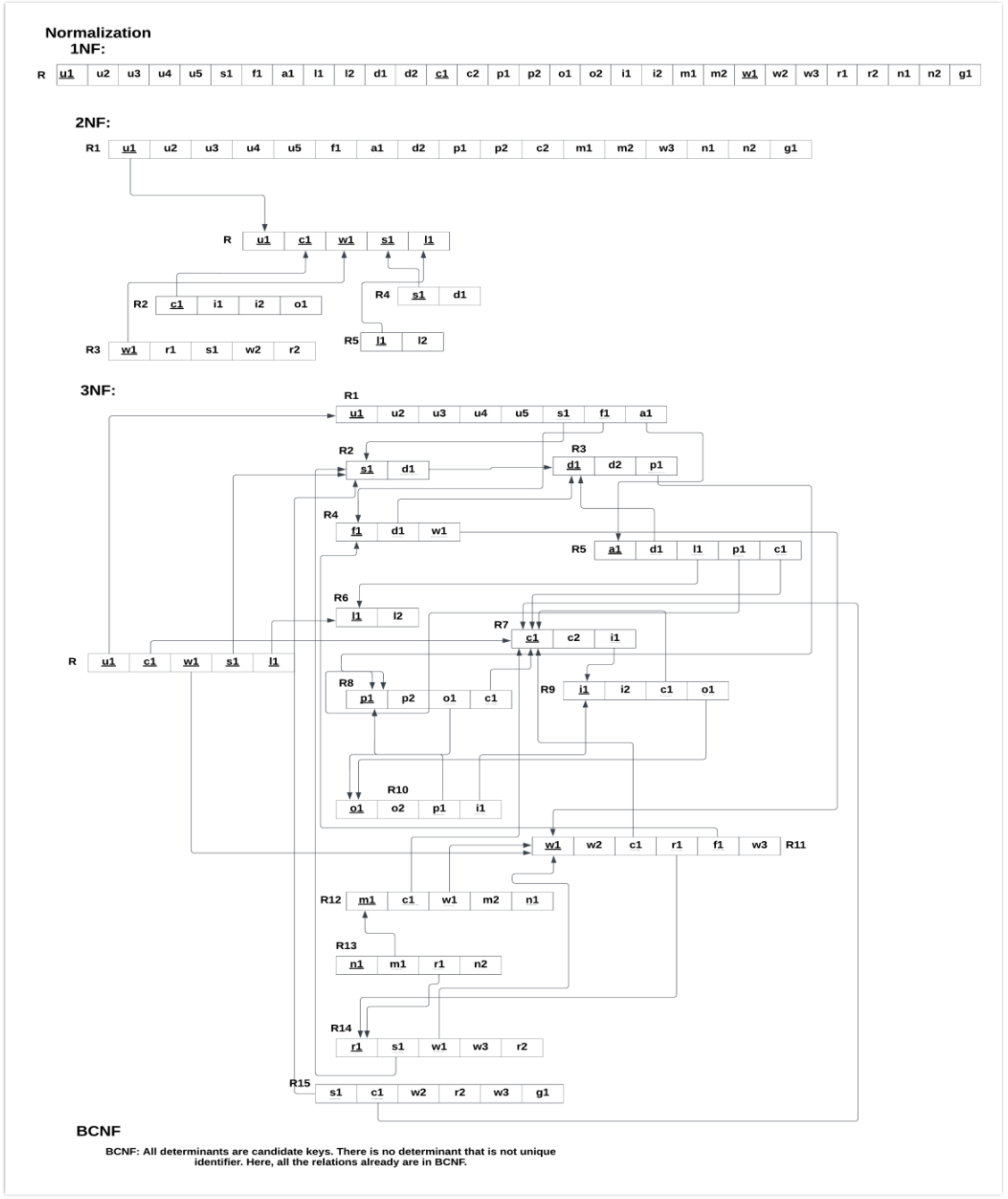
NORMALZATION

User	User_ID	u1	Program	Program_ID	p1
	First_Name	u2		Program_Name	p2
	Last_Name	u3			
	Contact_No	u4		Course_ID	c1
	Address	u5			
	Student_ID	s1		Plo_ID	o1
	Faculty_ID	f1			
	Admin_ID	a1			
Student	Student_ID	s1	PLO	Plo_ID	o1
	Department_ID	d1		Plo_Num	o2
Faculty	Faculty_ID	f1		Program_ID	p1
	Department	d1		Course_ID	c1
	Section_ID	w1	CO	Co_ID	i1
Admin	Admin_ID	a1		Co_Num	i2
				Couse_ID	c1
	School_ID	l1		Plo_ID	o1
	Department_ID	d1	Assessment	Assessment_Num	m1
	Program_ID	p1		Course_ID	c1
	Course_ID	c1		Section_ID	w1
School	School_ID	l1		Marks	m2
	School_Name	l2		Evaluation	n1
	Department_ID	d1		Section_ID	w1

Department	Department_Name	d2	Section	Section_Num	w2
	Program	p1		Course_ID	c1
Course	Course_ID	c1		Enrolment_ID	r1
	Course_Name	c2		Faculty_ID	f1
	CO_ID	i1		Semester	w3
Enrollment	Enrollment_ID	r1	Evaluation	Evaluation_Num	n1
	Student_ID	s1		Assessment	m1
	Section_ID	w1		Enrollment_ID	r1
	Semester	w3		Obtain_Marks	n2
	Year	r2			
CourseGPA	Student_ID	s1			
	Course_ID	c1			
	Section_Num	w2			
	Year	r2			
	Semester	w3			
	Grade	g1			

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

Normalization
(continued)



Data Dictionary

DATA DICTIONARY

School_T

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

Program_T

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

Department_T

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

Course_T

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

CLO_T

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

PLO_T

Name	Datatype	Size	Remarks
nPLOID	INTEGER		This is the primary key for Program Learning Outcome. E.g.: "1"
nPLONum	INTEGER		This is the PLO number. E.g.: "1"

Data Dictionary
(continued)

cDetails	VARCHAR	255	This is the details for Program Learning Outcome. E.g.: "An ability to select and apply the knowledge, technique, skills and modern tools of the computer science and engineering discipline"
cProgramID	VARCHAR	5	This is the foreign key from the pPogram_T. E.g.: "BSC1"

Assessment_T

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

Evaluation_T

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

Student_T

Name	Data Type	Size	Remarks
nStudentID	INTEGER		This is the primary key for the student table. E.g.: "1921834".
cFirstName	VARCHAR	30	This is the first name of the student. E.g.: "Farsheedl".
cLastName	VARCHAR	30	This is the last name of the student. E.g.: "Rahman".
dDateOfBirth	DATE	DD MM YYYY	This is the birth date of the student. E.g.: "21-12-1996".
cEmail	VARCHAR	30	This is the email of the student. E.g.: "1830360@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.: "Hou 1, Road 4, Block D, Bashundhara RA".
cProgramID	INTEGER		This is the foreign key from the progra table. E.g.: "BSc1"
cDepartmentID	VARCHAR	3	This is the foreign key from t Department table. E.g.: "CSE"

Section_T

Name	Datatype	Size	Remarks
cSectionID	VARCHAR	255	This is the Primary Key for Section. E.g.: "summer23csc10101"
nSectionNum	INTEGER		This is the section number. E.g.: "1"
cCourseID	VARCHAR	6	This is the foreign key from the Course table. E.g.: "CSE101"
dYear	YEAR	yyyy	This is the year of registration. E.g.: "2019"
cSemester	VARCHAR	10	This is the semester of the section. E.g.: "Summer"
cFacultyID	NUMERIC	4	This is the foreign key from Faculty table. E.g.: "1801"

Data Dictionary
(continued)

Enrollment_T

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

Faculty_T

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

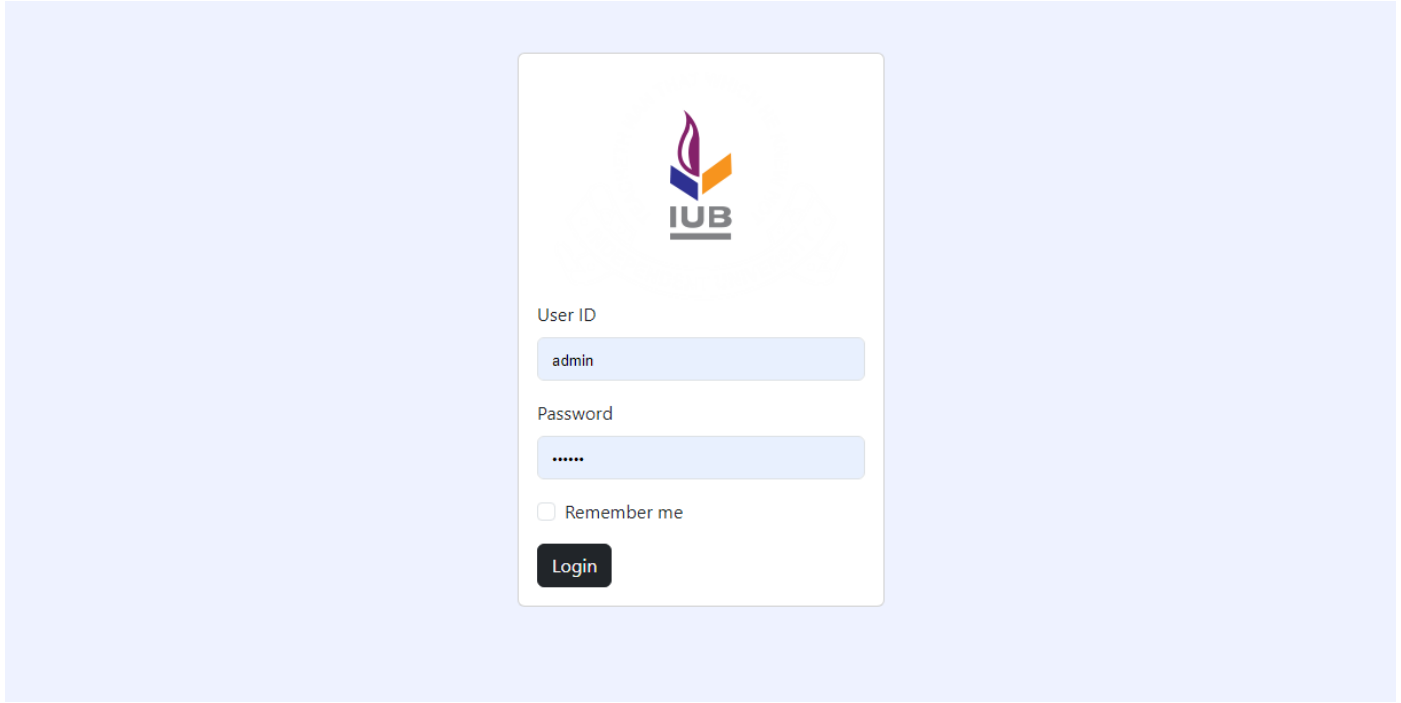
Admin_T

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

CourseGrade_T

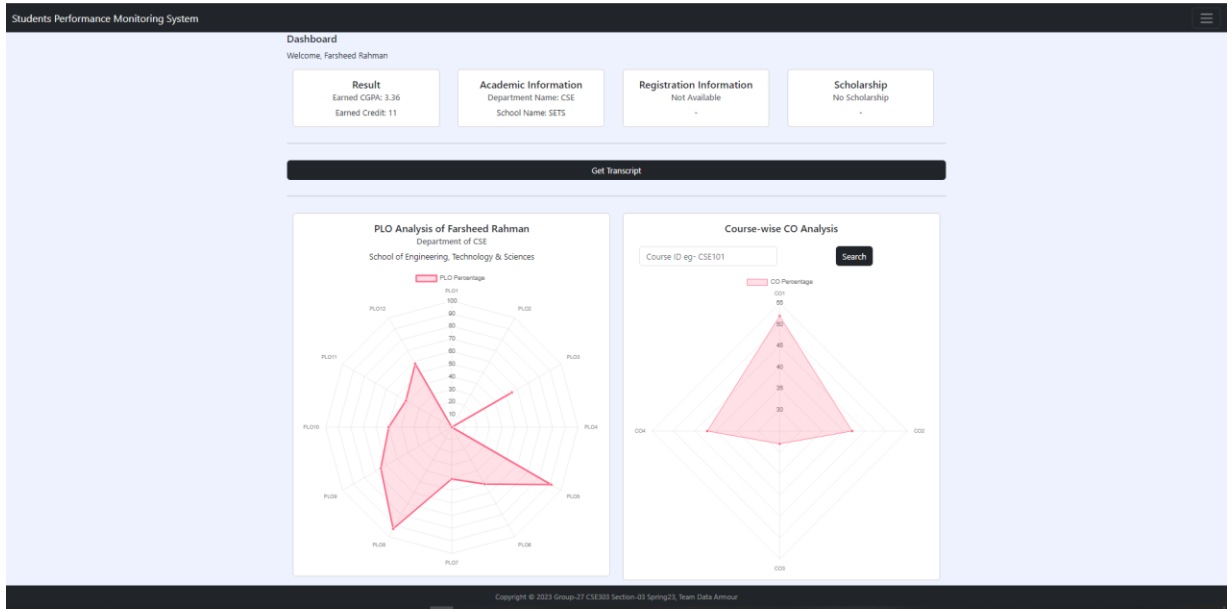
Name	Datatype	Size	Remarks
nID	INTEGER		This is the primary key for the CourseGrade_T table. It increments automatically E.g.: "4250"
cStudentID	NUMERIC	7	This is the foreign key from Student Table extended from User_T. E.g.: "1830398"
dEduYear	YEAR	yyyy	This is the year of registration or Enrollment. E.g.: "2019"
cEduSemester	VARCHAR	10	This is the semester of registration or Enrollment. E.g.: "Spring"
cCourseID	VARCHAR	6	This is the foreign key from the Course table. E.g.: "CSE101"
nSectionNum	INTEGER		This is the section number. E.g.: "1"
cGrade	VARCHAR	2	This is the Grade of a course example: "B".

Physical System Design (Login)




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

Physical System Design
(Student dashboard)

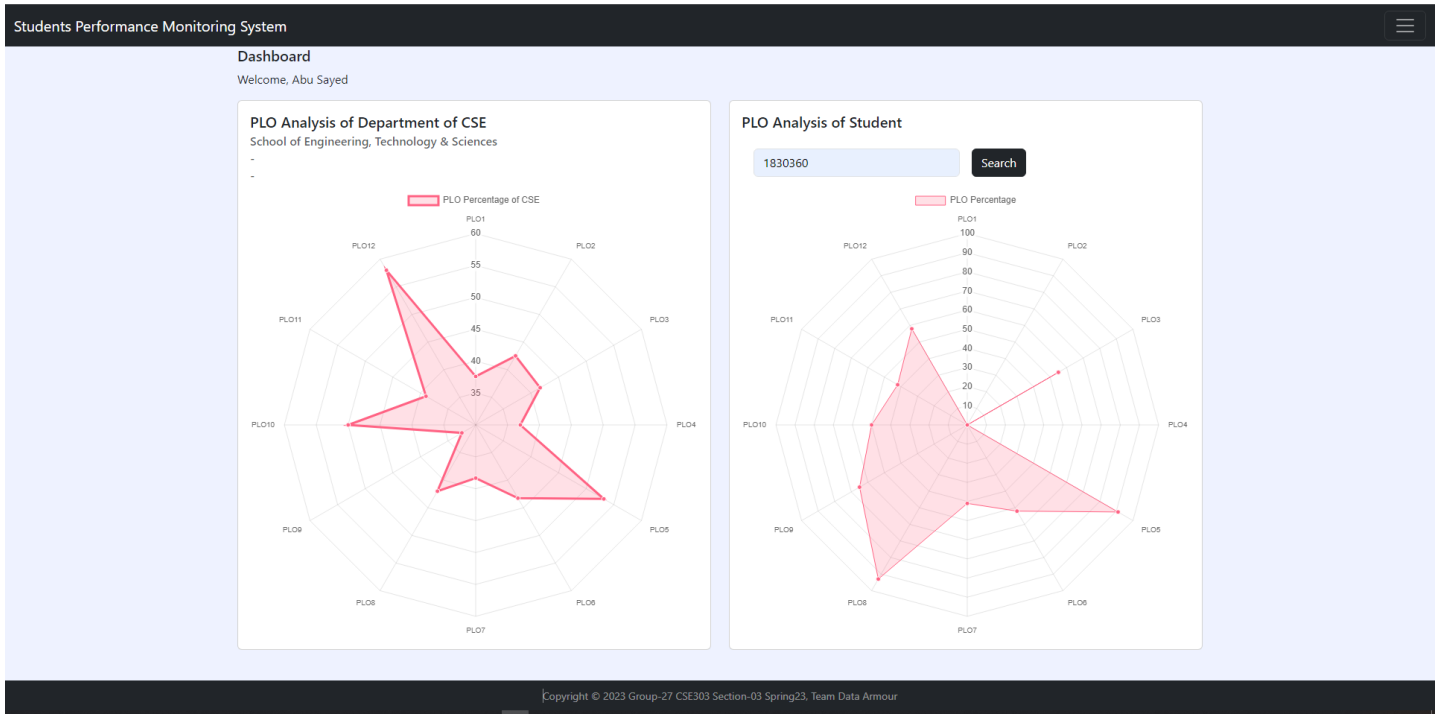



```
158             # Get all the grades from CourseGrade_T filtered by a specific student_id
159             grades = CourseGrade_T.objects.raw("SELECT * FROM app_coursegrade_t WHERE studentID_id = %s;", [request.user.id])
160             attempted_credit = 0
161             total_cum_credit = 0
162 
163             for grade in grades:
164                 if grade.grade == 'A':
165                     #course = Course_T.objects.get(pk=grade.course)
166                     course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
167                     attempted_credit+=int(course.creditNo)
168                     total_cum_credit+=float(int(course.creditNo)*4.00)
169                 elif grade.grade == 'A-':
170                     course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
171                     attempted_credit+=int(course.creditNo)
172                     total_cum_credit+=float(int(course.creditNo)*3.70)
173                 elif grade.grade == 'B+':
174                     course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
175                     attempted_credit+=int(course.creditNo)
176                     total_cum_credit+=float(int(course.creditNo)*3.30)
177                 elif grade.grade == 'B':
178                     course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
179                     attempted_credit+=int(course.creditNo)
180                     total_cum_credit+=float(int(course.creditNo)*3.00)
181                 elif grade.grade == 'B-':
182                     course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
183                     attempted_credit+=int(course.creditNo)
184                     total_cum_credit+=float(int(course.creditNo)*2.70)
185                 elif grade.grade == 'C+':
186                     course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
187                     attempted_credit+=int(course.creditNo)
188                     total_cum_credit+=float(int(course.creditNo)*2.30)
189                 elif grade.grade == 'C':
190                     course = Course_T.objects.raw("SELECT * FROM app_course_t WHERE courseID = %s;", [grade.course.courseID])[0]
191                     attempted_credit+=int(course.creditNo)
192                     total_cum_credit+=float(int(course.creditNo)*2.00)
193                 elif grade.grade == 'C-':
```

Physical System Design
(Student dashboard) Continued

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

Physical System Design
(Faculty dashboard)




```
223     # IF THE USER IS FACULTY
224     if request.user.role == 'Faculty':
225         ploS = None
226         if request.method == 'POST':
227             try:
228                 ploS = getPLO(request.POST['searchStudent'])
229             except:
230                 pass
231         return render(request, 'home/home.html', {  'plo': getDeptWisePLO(request.user.department.departmentID),
232                                                     'ploStudent': ploS})
233
234     if request.user.role == 'Admin':
235         departments=Department_T.objects.raw("SELECT * FROM app_department_t;")
236         if request.method == 'POST':
237             return render(request, 'home/home.html', {  'ploDepartment': getDeptWisePLO(request.POST['department']),
238                                                         'departments': departments})
239         return render(request, 'home/home.html', { 'departments': departments})
240     return render(request, 'home/home.html', {})
241 else:
242     return redirect('login')
```

Physical System Design

(Faculty dashboard) Continued

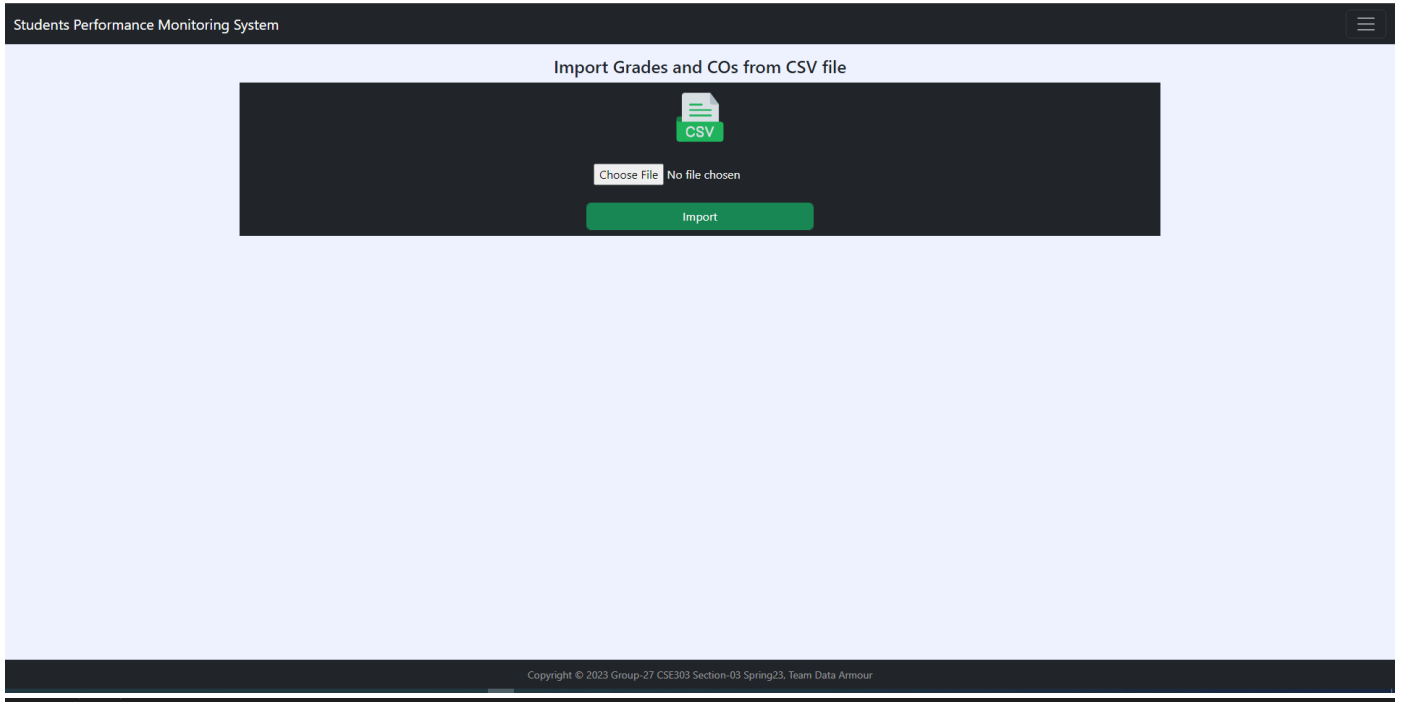
```python
# FUNCTION OF GETTING DEPARTMENT-WISE PLO
def getDeptWisePLO(dept):
    plodata = [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
    plodataC = [0,0,0,0,0,0,0,0,0,0,0,0]
    # Get all the assessments from the assessment_t
    plos = Assessment_T.objects.raw('SELECT * FROM app_assessment_t;')

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

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

Physical System Design

(Faculty CSV import Forms)




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

Physical System Design

(Faculty CSV import Forms) continued

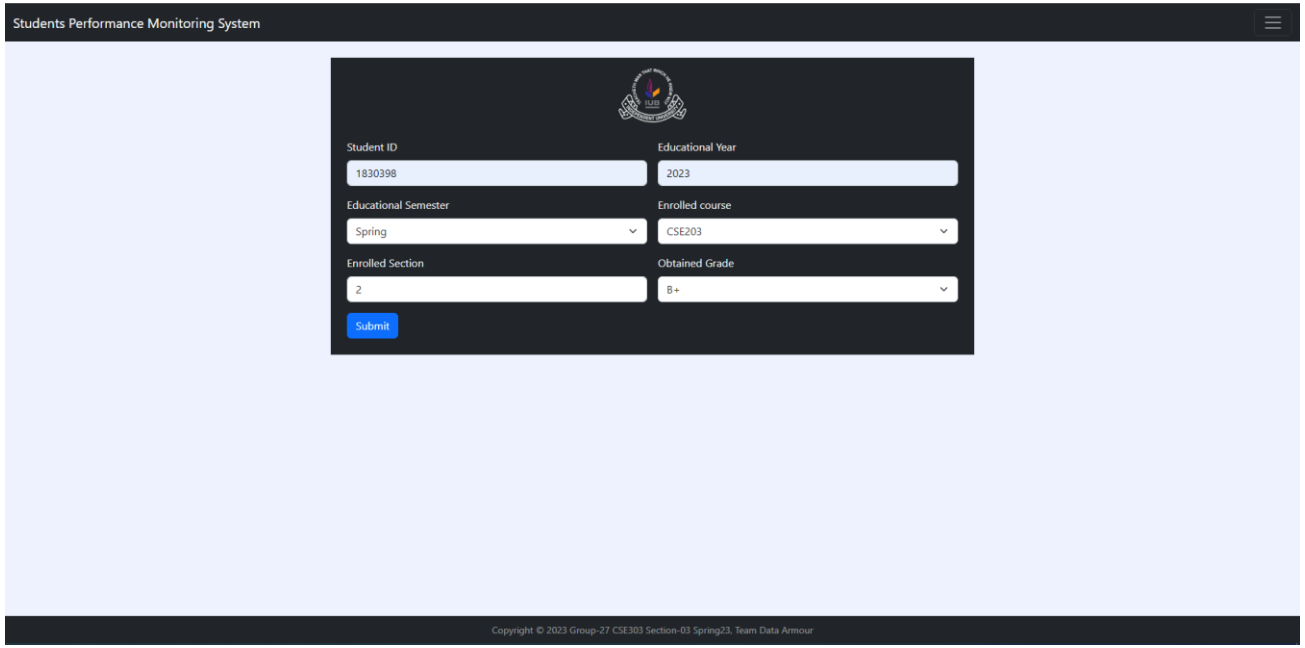
```
# Filter all the COs by the CourseID
cos = CO_T.objects.raw("SELECT * FROM app_co_t WHERE course_id=%s;", [courseT.courseID])
for cot in cos:
    if cot.coNo == 1 and str(row[5]) != '':
        form = Assessment_T(
            studentID=student,
            semester=str(row[2]),
            year=str(row[1]),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

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

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

Physical System Design

(Faculty Grade input Forms)

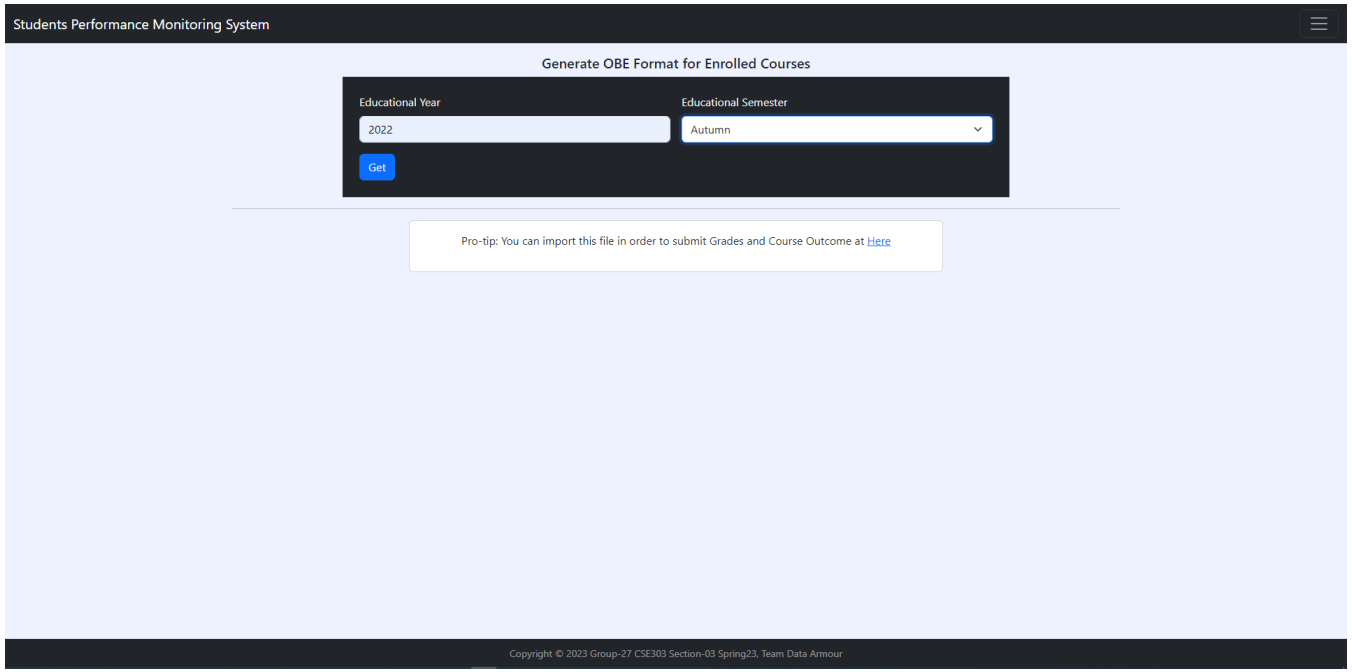


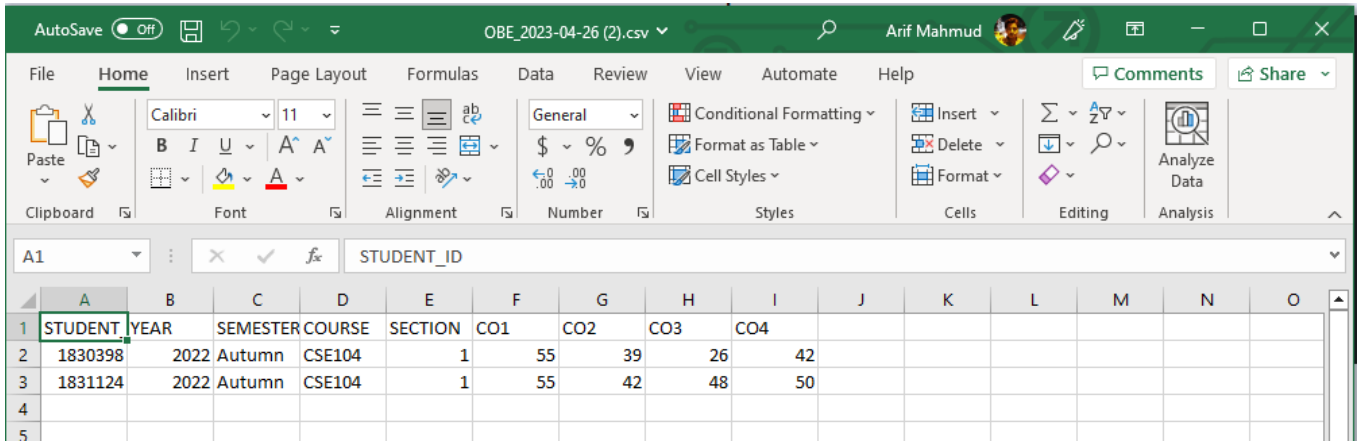

```python
def gradeInputForm(request):
    if request.user.is_authenticated:
        if request.user.role == 'Faculty':
            success = 'success'
            form = GradeInputForm()
            if request.method == 'POST':
                try:
                    # Filter the student from user_t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

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

Physical System Design

(Faculty OBE Generate Form)





Physical System Design

(Faculty OBE Generate Form) Continued [Backend code]

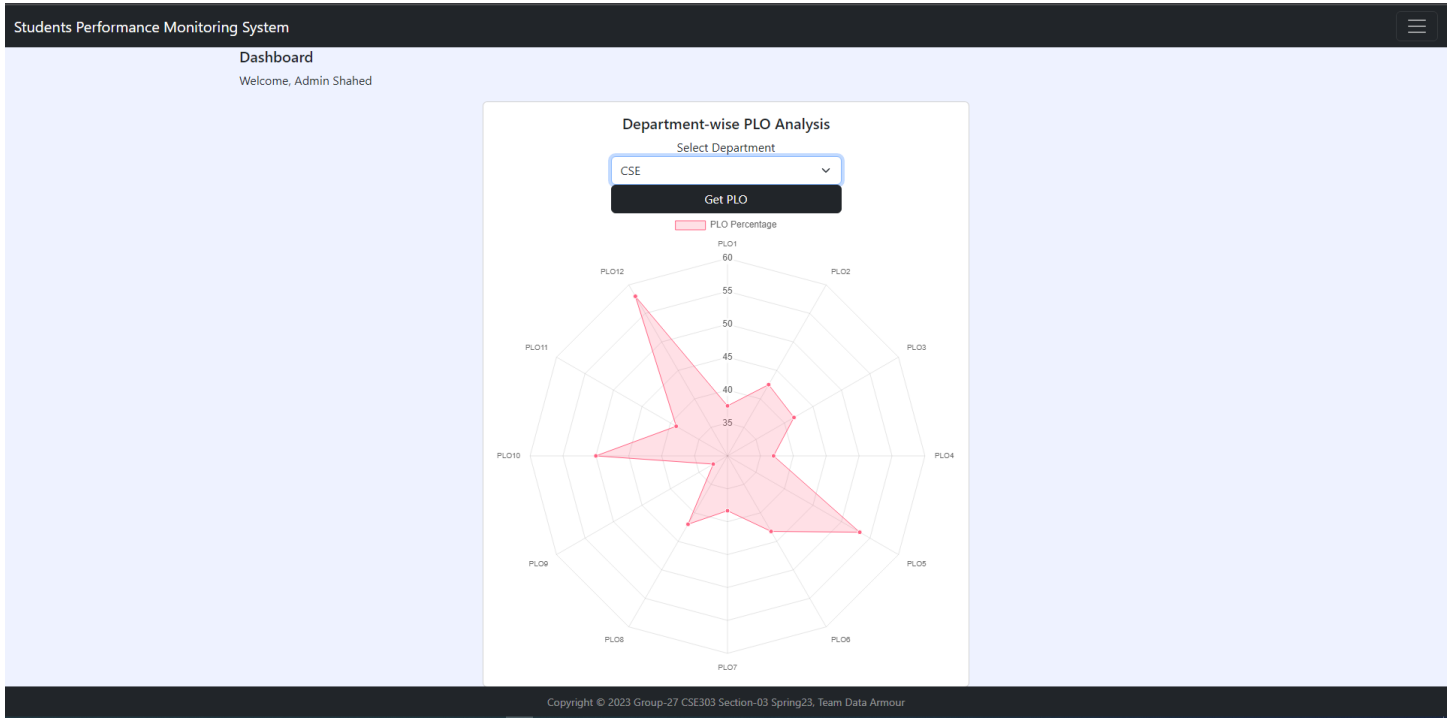
```python
def generate_obe_csv(request):
    if request.user.is_authenticated:
        if request.user.role == 'Faculty':
            if request.method == 'POST':
                response = HttpResponse(content_type='text/csv')
                response['Content-Disposition'] = f'attachment; filename=OBE_{datetime.date.today()}.csv'

                # Create CSV Writer

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

Physical System Design

(Admin/Dean Department wise PLO Analysis)



Physical System Design

(Admin/Dean Department wise PLO Analysis)
Continued [w/Backend code]

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

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

Database Models

spms > app > models.py

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

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

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

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

Database Models

Continued

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

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

Database Models
Continued

```python
# Assessment Table
class Assessment_T(models.Model):
    SEMESTER_CHOICES=(
        ('Spring', 'Spring'),
        ('Summer', 'Summer'),
        ('Autumn', 'Autumn'),
    )
    assessmentNo = models.AutoField(primary_key=True)
    studentID = models.ForeignKey(User_T, on_delete = models.CASCADE)
    semester = models.CharField(max_length=30, choices=SEMESTER_CHOICES, blank=True, null=True)
    year = models.CharField(max_length=4)
    marks = models.FloatField()
    co = models.ForeignKey(CO_T, on_delete=models.CASCADE)
    section = models.ForeignKey(Section_T, on_delete=models.CASCADE)
    def __str__(self):
        return str(self.assessmentNo)
# Evaluation Table
class Evaluation_T(models.Model):
    evaluationNo = models.AutoField(primary_key=True)
    obtainedMarks = models.FloatField()
    assessment = models.ForeignKey(Assessment_T, on_delete=models.CASCADE)
    enrollment = models.ForeignKey(Enrollment_T, on_delete=models.CASCADE)
    def __str__(self):
        return str(self.evaluationNo)+' '+str(self.assessment)+' '+str(self.enrollment)
# Course GPA
class CourseGrade_T(models.Model):
    studentID = models.ForeignKey(User_T, on_delete = models.CASCADE)
    eduYear = models.CharField(max_length=4)
    eduSemester = models.CharField(max_length=25)
    course = models.ForeignKey(Course_T, on_delete=models.CASCADE)
    section = models.IntegerField(default=1)
    grade = models.CharField(max_length=4)

    def __str__(self):
        return str(self.studentID)+' '+str(self.course)+' '+str(self.grade)
```

Problems And Future Development

Problems:

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

Future Developments:

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

Conclusions

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

Thank You!