

**CSE303: Database Management System**

**Section: 01**

**Project Name: Student Performance Monitor**

**Group Number: 03**

|  |  |  |
| --- | --- | --- |
| **SL** | **Name** | **ID** |
| **1** | **Rasula Makbul** | **1730019** |
| **2** | **Md. Shadman Shakib Talukder** | **1810762** |
| **3** | **Tamanna Shabnam** | **1821801** |
| **4** | **Md. Musfiqur Rahman** | **1721684** |
| **5** | **Ahmed Jubair Nijum** | **1730265** |

**TABLE OF CONTENT**

**Chapter 1: Introduction**

Section 1.1**:** Background of the Organization

Section 1.2: Background of the Project

Section 1.3: Objectives of the Project

Section 1.4: Scope of the Project

**Chapter 2: Requirement Analysis**

Section 2.1: Rich Picture (AS-IS)

Section 2.2: Six Elements Analysis (AS-IS)

Section 2.3: Process Diagram (AS-IS)

Section 2.4: Problem Analysis

Section 2.5: Rich Picture (TO-BE)

Section 2.6: Six Elements Analysis (TO-BE)

Section 2.7: Process Diagram (TO-BE)

**Chapter 3: Logical System Design Section**

3.1: Business Rules Section

3.2: ERD Section

3.3: Relational Schema Section

3.4: Normalization Section

3.5: Data Dictionary

**Chapter 4: Physical System Design Section**

4.1.1: Input Forms-Add User

Purpose

Related SQL Used

3 Section 4.1.2: Input forms-Add program & PLO

Purpose

Related SQL Used

Section 4.1.3: Input forms- Add Course with CO

Purpose

Related SQL Used

Section 4.1.4: Input Forms -Add Marks Individual

Purpose

Related SQL Used

Section 4.1.5: Input forms-Add Marks Mass

Purpose

Related SQL Used

Section 4.2.1: Output Query and Reports – Higher Management Dashboard

Purpose and Use

Section 4.2.2: Output Query and Reports – PLO Achievement

Purpose and Use

Section 4.2.3: Output Query and Reports – Progress View

Purpose and Use

Section 4.2.4: Output Query and Reports – Student Result

Purpose and Use

Section 4.3: System Design Architecture

**Chapter 5: Conclusion Section 5.1: Problem & Solution**

Section 5.2: Additional Features and Future Development

Section 5.3: Conclusion & Recommendation

4 Contribution of Each Member

References

**CHAPTER 1 – INTRODUCTION**

**1.1: BACKGROUND OF THE ORGANIZATION**

Independent University – Bangladesh (IUB) founded in 1993 and presently has more than 7,378 undergraduate students and more than 11,556 alumni, is one of Bangladesh's oldest private institutions. The population of students is anticipated to increase by 10% per year. Over the years, IUB has been able to succeed in producing employable graduates only via being up to date and improved relentlessly. The school has also grown to a well-funded research center that carries out many Research Projects and, more particularly, to the department of Computer Science. Due to the assistance of the Government of Bangladesh and UGC, IUB has been able to set up state-of-the-art laboratory facilities in its department. Due to the approach of the IUB to academics as a philosophy of 'application-oriented learning' the students "do not only learn the basic concepts of computers, but also provide them with practical instruction in the diverse applications of this knowledge."

**1.2: BACKGROUND OF THE PROJECT**

Our basic aim is to create a program that can enable colleges around the globe to help evaluate student performance more productively and effectively. We presented the notion of mapping learning outcomes (PLOs) program with courses for the very first time in our project. Then the result of each PLO map with cos. The aim of the project is to assess the pupils efficiently whether or not each student meets the PLO requirement linked to the CO. The technology will provide the Department with access to the PLO. The department then enters the COs to allow the system to map the COs properly into the PLO. The system will allow the faculty member to update the software in line with the course requirements, the user will be able to access data on the departments or course assessments in terms of completing coaching in a particular course and overall PLOs at the Independent University in Bangladesh in one semester. This software can examine its success in the program in the annual student assessment report. This helps to obtain correct student information from majors, departments and institutions.

**1.3: OBJECTIVE OF THE PROJECT**

The objective of this project is for the monitoring of university academic achievement by a more effective and less error propensity method. By postponing the conventional file system database architecture, a club-based, relational database management system (OBE) is implemented in the new and enhanced Student Performance Monitoring System (SPMS). The Independent University, Bangladesh will utilize the system. Current statistics data representation in Independent University, Bangladesh on student performance assessment. Upgradable design to expand further. The system is open to students, faculty, administrators and interested parties. Create a system more efficient to determine PLO & CO values, monitor historical scores, and provide future forecasts than the present method.

**1.4: SCOPE OF THE PROJECT**

The project will be a substitute for the present manual system. Our solution consists in creating, editing, adding and updating data needed for surveying students' performance, as well as producing and saving relevant OBE data, reports and documents utilizing a cloud-based relational database management system.

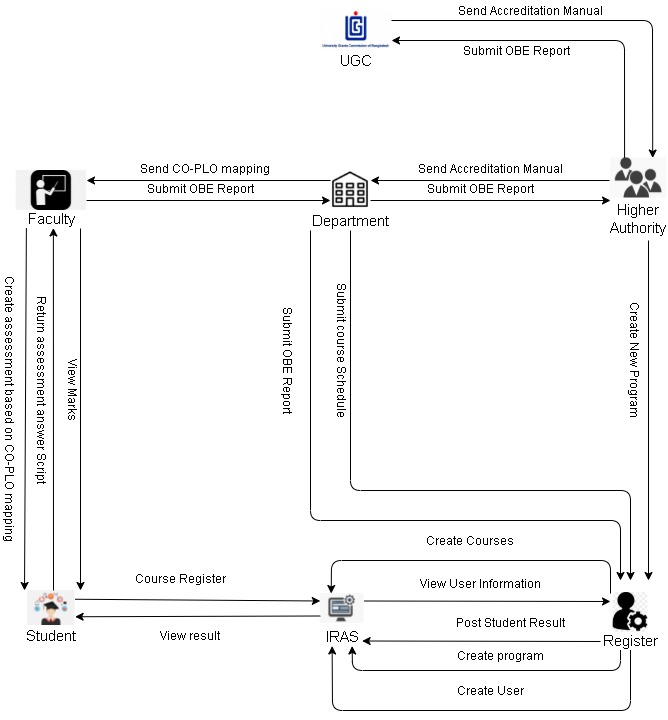
We have manufactured possible consumers for the web-based SPMS system and specified how the system and the information and data it needs to be accessed are used. Since the problems might arise from various areas of all business procedures, all the stakeholders who will be users of this system also will have unique user interfaces and login capabilities.

Because we are able to utilize SPMS to store data, it is very easy to retrieve essential files, tabular data, layouts and reports and to use them in real time. In order to conveniently access and use these data to produce and download reports, we also build interfaces for all users.

We provide an interface for faculty to work with one other to generate curriculum descriptions, course reports, leaflets, assessments, mapping evaluations of COs and PLOs for PLO accomplishment and record evaluations of students during all their courses throughout the semester. Data will also be secured and only data relevant to each stakeholder will be presented.

**CHAPTER 2: REQUIREMENT ANALYSIS**

**2.1: RICH PICTURE**



**2.2: SIX ELEMENT ANALYSIS (AS IS)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Process | System Roles | | | | | |
| Human | Non-  Computing  Hardware | Computing  Hardware | Software | Database | Network  &  Communication |
| Mapping Course outcomes(COs) to Program Learning Outcomes (PLOs) | **UGC :**  1.Provides the manual of accreditation to higher authorities with stated PLOs  **Higher Authorities:**  1.Receive Accreditation Manual from UGC  2. Send the Accreditation Manual to the Department.  3. Directs Department to tell the faculties to design Course Outline and Course Assessment Reports.  **Department :**  1. Receive Accreditation Manual from Higher Authority.  2. Map Cos with PLOs.  3. Send the CO-PLO mapping to respected faculty.  **Faculty:**  1. Receive CO-PLO mapping from department.  2. List Course content and Outcomes.  3. Create a course plan based on CO-PLO mapping.  4. Continue the semester according to the course plan. | **Pen and paper:**  1. Is used to record brainstorming ideas.  **Board and marker:**  1.Is used to noting down the ideas | **Computer:**  1. Computers are used by faculty to produce softcopies of course results (COs) of certain courses in which they are skilled.  **Printer:**  1. Printing copies of the course results (COs). | **MS Word:**  1.Faculty uses MS Word to map the program learning results with full course contours and course assessment reports with course results (COs) (PLOs).  **Excel Sheet:**  1. Faculties use the Excel Sheet to correlate certain questions to specific course results for mid-term, final exams, and project work (COs). |  | **Internet & Email:**  1.To address crucial subjects relating to planning of course results for program learning outcomes, use the Internet and e-mails to communicate with UGC or other stakeholders.  **Others:**  1.To address key issues relating to the mapping courses, use phones or physical means with stakeholders for outcomes in the learning of programs. |
| Student Assessment | **Faculty** :  1. Create assessment as per course plan.  2. Assign students Assessments.  3. Receive completed answer script from students.  4. Evaluate the answer script.  **Student:**  1. Receive assessment from faculty.  2. Answer the assessment as assigned.  3. Return the answer script to faculty. | **Pen and paper:**  1. Use pen & paper to record physical paper data and marks in tabular format (hardcopies) | **Computer:**  1.Computers are able to generate soft copies of all evaluation data for certain courses | **Excel Sheet:**  1. Record the relevant evaluation data on Excel sheets and their final grades.  **IRAS :**  1.To view students and the registrar's office, upload the final grades of students to IRAS | **Department Storage:**  1. The departmental and registrar's offices may store records of student evaluation data and final grades for future reference.  **IRAS Database server:**  1. In order to record and retain student grade information, IRAS uses a server database | **Internet:**  1.In order to store the final grades, the Internet is needed to communicate with IRAS |
| **Creating & Submitting OBE Report** | **Faculty:**  1. Calculate the total marks received by each CO computing the marks received by mapping to COs for questions and/or further assessments.  2. Total percentage calculations receivable on the OBE Mark sheet, for each Course Results (COs).  3. State if a student has a certain CO (if CO percentage is greater than or equal to 40).  4. Report whether a PLO for an associated CO has been given to a student.  5. Make a table to assess how many students could receive a specific CO and PLO as well as other documents including facts and data.  6. Report for design course evaluation using course outlines, course content and course results. Design course evaluation report.  7. Send the final version of the OBE Mark sheet to the Department.  **Department:**  1. Send the OBE mark sheet, Course Assessment Report and other reports to the Registrar’s Office.  2. Store the OBE Mark sheet and Course Assessment Report to Higher Authority. | **Pen and paper:**  1. OBE mark sheet stored in hardcopy. Additional markings may be made to further separate between students | **Computer/ Phone:**  1.Uses computers to make softcopies of the OBE Mark sheet and Course Assessment Reports.  **Printer:**  1. Print hardcopies of final versions of the OBE Mark sheets and Course Assessment Reports. | **Coded Excel sheet:**  1. Faculty /Course Coordinator uses automated excel sheets to calculate the student’s success/ failure in achieving PLOs.  **MS Word:**  1. Used to make course assessment reports. | **Department Storage:**  1. Records of students’ assessment data and final grades will be saved in the department for future reference.  **Registrar’s Office Storage:**  1. OBE Mark sheets, Course Assessment Reports and other documents submitted by the department is stored for future reference. | **Internet/Mail:**  1**.** An Online platform (such as Google Sheets) may be used for processing the OBE assessment data spreadsheet. |
| **Preview Grades and Transcripts** | **Students:**  1. Log into IRAS.  2. Search semester wise result for intended semester.  3. See grades for specific semesters.  3. Download transcript through browser into hard disk.  **Registrar’s Office:**  1. Access IRAS.  2. View students’ grades if and when it’s necessary.  3. Download their transcripts | **Pen and Paper**  1. Transcripts can be printed on paper with tabulated transcripts. During applications and other paperwork, Hardcopy is the primary source of truth. | **Computer/**  **Phone:**  1. Used for accessing IRAS.  **Printer:**  1. Used to print the tabulated transcripts. | **IRAS:**  1. Storesletter grades of each completed course  2. Provides the online user interfacefor viewing grades and transcripts | **Registrar’s Office:**  1. Student information is kept in admin in hardcopies for future reference.  **IRAS Database:**  1. A Database Management Service is used to store, maintain, edit and receive student grades information in IRAS.  **Web Server:**  1. User interface and website pages are served using a remote web server | **Internet/ Email** 1. In order to store the final grades, the Internet is needed to communicate with IRAS.  2. Softcopies may be mailed to the stakeholders and students. |
| **Student and Faculty account details** | **Admin :**  1. The registration processes collect information for new students.  2. New faculty information is received from Higher Authorities.  3. Creates an account for students and Faculties.  4.If necessary for student or Faculty, customize some account details | **Pen and Paper:**  1. May be used for writing/ copying student/ faculty’s vital login information for account creation. | **Computer:**  1. Used for accessing and adding/editing data to IRAS. | **IRAS:**  1. User interface is provided to interact with student/faculty data. | **Registrar’s Office Storage:**  1. Student/ Faculty information is kept in admin in hardcopies for future reference.  **IRAS Database Server:**  1. A Database Management Service is used to store, maintain, edit and receive student/faculty information in IRAS.  **Web Server:**  1. User interface and website pages are served using a remote web server. | **Internet:**  1. The internet is needed to interact with IRAS to store account information on a remote database server.  2. User interface  and website pages are served using internet access. |

**2.3: PROCESS DIAGRAM(AS IS)**

**Diagram

Description automatically generated**

**Diagram

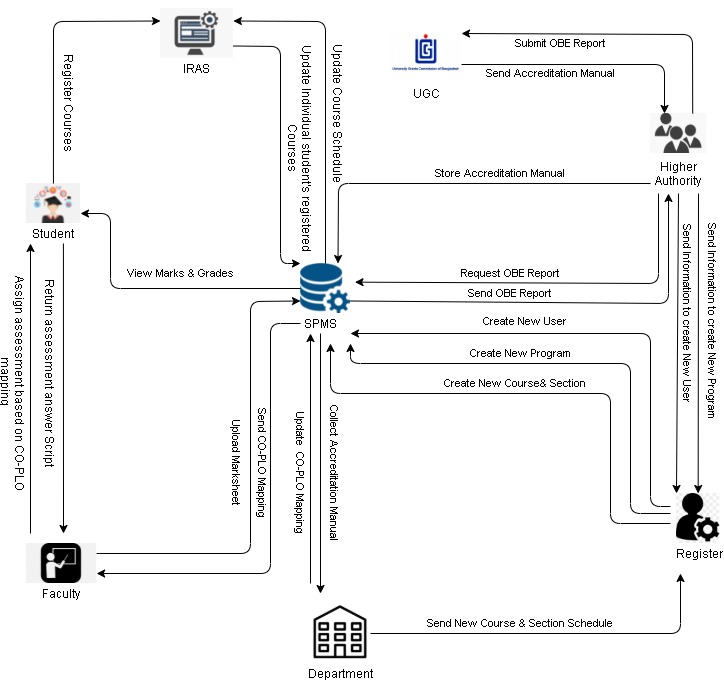
Description automatically generated**

**Diagram

Description automatically generated**

**2.4: PROBLEM ANALYSIS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process Name** | **Stakeholders** | | **Concerns (Problems)** | **Analysis (Reason of the problem)** | **Proposed Solution** |
| |  | | --- | | **Mapping Course outcomes(COs) to Program Learning Outcomes (PLOs)** | | **Higher Authorities** | | 1. If hardcopy is dispatched, Head of Department or Dean of the School may not get the PLO Accreditation Manual Definite by UGC because of a lost mail. After mailing the manual, it can even be received lengthy time.  2. Email-sent soft copies may also cause the manual to lay down or lose time.  3. The Head/Dean could neglect to direct its personnel to work on the tasks necessary for circulation of the Accreditation Handbook (for example, forget which course Faculties should take care of the course evaluation courses). | This delay could cause huge delays to the business process when receipt of the Handbook and/or the possibility of its loss via Email or traditional emails. Failure to understand or lack of communication between the Higher authorities and Departmental personnel can also hinder or postpone the operation. | Make this process entirely easier on a private online platform (call it SPMS App) and create an interface to our suggested system where UGC Staff can immediately communicate files, documents and comments to the Superior Authorities. |
|  | **Department** | | 1. Unintentionally, Department personnel could not transmit the Accreditation Manual or send it to the wrong columns or the instructor/coordinator. | Due to the increasing number of faculties and training sessions, it may be more difficult to track which faculties take courses for which semesters and thus cause a time lag in the business process, while keeping track of all the information in so many documents. | Our website can be linked to a database containing information about all courses, faculties and updated tables each half year, which track the courses to which faculties in the semester have been assigned. |
|  | **Faculty** | | 1. The process of listing the content and course results (COs), mapping it to and from the PLOs and designing the course review report and course description can be vulnerable to mistakes. | The report and the course contour as well as the course outcomes may fail without the supervision of the expert or stakeholder involved (UGC) or both (COs). | The ongoing contact between the higher authorities and faculties demands that the Course Coordinators and Instructors of each course constantly update and reference. Download Course Outcome Mapping as a table for our system to subsequently use in the production of OBE Mark sheet and OBE Analysis Report, to address specific questions of special assessments. |
| **Student Assessment** | **Faculty** | | 1. Some of the questions in the question paper might go incorrect.  2. Students can cheat on their homework from any type of website.  3. When computing markings for evaluations, the faculty might make mistakes.  4. Sometimes faculty don't equally or on a curriculum/course basis examine the response scripts.  5. There are distinct grading methods in different faculties. | In the same document a duplication issue may occur or certain typographical errors might occur. There may be insufficient room to write the answers.  In their tasks/quizzes/project work, students might commit plagiarism.  Errors may occur during computation manually or through automation in Excel sheets during insertion.  Papers are not examined by other faculty and only a few faculties evaluate and dislike themselves.  Not everyone has the same grading methods, and students and other stakeholders that wish to monitor student performance in these courses may be confused. | Faculties should recheck the question paper before printing it. They should provide extra sheets to the students during the exam.  Faculties should examine whether or not they have done any plagiarism and act accordingly with a plagiarism inspection.  Before the marks and formulae for automation calculations are placed in mark sheets, the faculty should double verify. Before updating them on the brochure, the faculty should present the students with the marks.  The grading scale must be kept by the system and the system must have computational capabilities and thus generate the grade. The ratings are derived using the marks of several examinations given to it by the system itself.  Create a database to save a grading system for all courses and required user interfaces that will allow students, university/high management, and officials of UGC to examine grading regulations on one page. |
| **Creating & Submitting OBE Report** | **Faculty** | | 1. Total mark calculation obtained for each CO; total percentages of each CO may be susceptible to mistakes.  2. Sending the storage department a hardcopy and softcopy of the OBE Mark sheet and Course Assessment Report might lead to increased time consumption, delays or lost information and essential information. | This calculation is done manually or with an Excel sheet during the formats to automate calculation columns is subject to mistakes that can lead to an indetermination in the analysis and verdict phase of the reports of the students receiving codes and associated PLOs.  As hard copying and soft copying may easily lead to misunderstanding, data loss and reports to be sent to the Department and Registrar's Office. | A mapped table containing particular questions on a specific evaluation of a course under one faculty can be stored for students getting COs and associated PLOs on the customized interface offered by our SPMS App (was entered into the SPMS App during the Mapping of COs to PLOs process).  The faculty can upload to our database and examine the faculties utilizing user interfaces for their particular purposes by interested departments. |
|  | **Department** | | It becomes difficult and difficult to manage to storage softcopies and hardcopies. | When the company has been running for years, soft copying and hard copying might be exceedingly tough to handle.  It also becomes progressively tiresome for student performance trends to follow papers for a particular time frame. Updating information to a particular document would involve monitoring it, which would make it more difficult to obtain it and need staff to update different versions. | The maintenance of the data tables in our database and the appropriate authority’s access can solve these difficulties through a user interface to see and print them. |
| **Preview Grades and Transcripts** | **Students** | | If grades are sent to IRAS, IRAS is slowed down and ready for students to view. | Many students that use IRAS and request their browser at once produce a significant loading and slower than typical of the system and the server resulting in business delay. | 1. Better server maintenance and scaling for a rising number of users in all aspects of web technology. Conduct frequent updates for a competent, committed and in-house IT team.  2. Email each student at the conclusion of each semester with a full transcript document. Students can therefore verify their qualifications each time without having to log into IRAS. This reduces internet traffic to IRAS for students. |
| **Student and Faculty account details** | **Admin** | | 1. Student and faculty information may be lost.  2. Admin may make/miss information when copying student/faculty data, typing errors may be produced.  3. Data entry from students and teachers may at first lead to issues and delays in being able to carry out online university related activity and must be done manually via physical management channels.  4. Customization of information for students and faculty when needed could cause operation overhead for Admins. | As early data on hardcopies are obtained, this lazy paperwork and the documents included might be easily misplaced. Because a wide range of student/faculty information has been inserted into the system, typing mistakes might easily occur. It might be laborious, frightening, time consuming and prone to error/information to enter huge amounts of student/faculty data by copying it from hardcopies.  Many students and faculties may have to update their data on a day and hinder other vital administrative activities as well as take time to modify the information for every student/ professor. | Many students and faculties may have to update their data on a day and hinder other vital administrative activities as well as take time to modify the information for every student/ professor.  Allow students and teachers to update their own system information by utilizing the SMSP App interface. |
|  | |

**2.5: RICH PICTURE (TO BE)**

**2.6: SIX ELEMENT ANALYSIS (TO BE)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **System Roles** | | | | | | |
| **Process** | **Human** | **Non-computing hardware** | **Computing hardware** | **Software** | **Database** | **Communication & Network** |
| |  | | --- | | **Mapping Course outcomes(COs) to Program Learning Outcomes (PLOs)** | | **Higher Authorities:**  1. Receive the accreditation manual from UGC.  2. Upload the accreditation manual to SPMS.  **Department :**  1. Receive the accreditation manual from SPMS.  2. Updates the accreditation manual with required PLOs.  **Faculty:**  1. Receive updated accreditation manual through SPMS server.  2. List course content.  3. List Course Outcomes (COs).  4.Map Course Content to Course Outcomes(COs)  5.Make first  Version of Course Assessment Report.  6.Map assessments and assessment  questions to COs and PLOs in related  tables in the front end for each  Course.  7. Design course assessment using Course Outline, Course Content and Course Outcomes in the SPMS. | **Pen and Paper:**  1. Used for taking notes for new ideas. | **Computer/ Phone:**  1. Used for Higher Authorities to upload accreditation manual on SPMS.  2. Used for Department to update accreditation manual with required PLOs.  3.Used by Faculty  To make softcopies of Course Outcomes (COs) of the specific courses they are experts in and to map COs with PLOs.  **Printer:**  1. To print out hardcopies of Course Outcomes (COs) if needed. | **SPMS Higher Authorities frontend:**  1. Higher Authorities uses it to upload accreditation manual from the UGC to SPMS.  **SPMS Department frontend:**  1. Department uses it to update accreditation manual with required PLOs.  **SPMS Faculty frontend:**  1.Used by  The faculties to list, edit and map COs and PLOs and display course assessment report. | **SPMS :**  1. The required files, documents and reports created by the authorities, the department and the faculty for the map of COs and PLOs are stored in a data base. | **Internet:**  1. SPMS is an online web application for which all packets and requests online are transmitted.  **Mail:**  1. Email is primarily used for communication.  **Phone:**  1. May be used for communication. |
| **Student Assessment Data** | **Faculty:**  1. Log into SPMS faculty dashboard.    2. Faculty will be shown the courses they have/had for every semester under.  3. Select course (and section thereof).  4. Create (quiz/ exam/ project)  5. For each student, each student’s score for each question  6. Save the scores for that assessment. | **A4 paper:**  1. Used for recording student’s on-site response to Exam/ Quiz/ Project questions.  **Google Forms**  1. Used for recording a student's remote response to the questions. | **Computer/ Phone:**  1. Used for accessing the SPMS frontend. | **SPMS faculty frontend:**  1. Provides user interface for the faculty to enter student assessment data. | **SPMS :**  1. Stores student assessment data and derives student assessment information.  2. Sends email notifications to students at the end of each major assessment.  **Google Classroom:**  1. Import assessment data from google forms (or classroom, depending on their API), manually or automatically. | **Internet:**  1. SPMS is an online web application for which all packets and requests online are transmitted.  **Email:**  1. Email is the primary method of notifying the students about major assessment stages. |
| **Creating & Submitting OBE Report** | **Faculty:**  1. Uses SPMS App to make an OBE Mark sheet and Course Assessment Reports.  2. Log in SPMS.  3. Go to Courses  4. Each course has the "OBE Mark sheet”  5. The COs and PLOs will be mapped and mapped to specific questions in the CO and PLO Mapping phase.  6. This reports can be saved in the database by the Faculty. |  | **Computer/ Phone:**  1. Used for accessing the SPMS frontend.  **Cloud Server:**  1. Store processed/ grouped data back into SPMS.  2.Calculate CO percentages and determines success | **SPMS Assessment**  **Algorithm:**  1. Dynamically derive course evaluation information (e.g. on the accomplishment of CO and PLO) on the basis of course evaluation data and PLO-CO mapping. | **SPMS :**  1. Recover and table the PLO/CO performance data for a certain course and student(s).  2. The results analysis for students comes from the tabled data.  3. Store and collect required reports | **Internet:**  1. SPMS is an online web application for which all packets and requests online are transmitted. |
| **Preview Grades and Transcripts** | **Students**  1. Log into IRAS.  2. Search semester wise result for intended semester.  3. See grades for specific semesters.  4. Download transcript through browser.    5. Log into SPMS.  6. The option for each course will include its course name, ID and faculty name, alongside your comments.  7. The menu also includes a list of COs and PLOs.  8. Students can check grades and their performance for individual courses. | **Pen and Paper:**  1. Transcripts that are tabulated can be printed on paper. Hard copying is utilized in applications and other documents as the primary source of truth. | **Computer/**  **Phone:**  1. Used for accessing SPMS and IRAS.  **Printer:**  1. Used to print the tabulated transcript. | **SPMS Student frontend:**  1. Stores the list of PLOs and COs for a specific student.  2. Stores grades of each completed course.  **IRAS:**  1. Generate student transcript. | **SPMS:**  1. In order to store, maintain, update and receives each student's list of COs and PLOs, grade information for the student, and transcript in SPMS a database management service is utilized.  2. A remote web server is available to provide the user interface and website pages. | **Internet:**  1. SPMS is an online web application for which all packets and requests online are transmitted.  **Email:**  1. Softcopies may be emailed to students. |
| **Student and Faculty account details** | **Registrar Office:**  1. For creation or change account data sends user information/data to Admin.  **Admin:**  1. Logs into SPMS dashboard.  2. Creating new account/managing existing account.  3. Fills up the information required/edits the information fields of existing account.  4. Saves the new account/edited details. | **Pen and Paper:**  1. May be used for writing/ copying student/ faculty’s vital login information for account creation.  2.Hard copies | **Computer:**  1. Used for accessing and adding/editing data to SPMS/IRAS. | **IRAS:**  1.Provides user interface for add/managing accounts    **SPMS:**  1. Provides UI for the add/edit account data.  2. Shows new account input information fields.  3.If existing account, shows list of accounts & search bar for account search  4.Provides editing of information fields  5. provide option to save account information/data. | **IRAS Database:**  1. To save, maintain, update and receive account data, a database management service is utilized.  **SPMS Database:**  1. To save, maintain, update and receive account data, a database management service is utilized. | **Internet:**  1.SPMS is an online web application for which all packets and requests online are transmitted.  2. The internet access will be used for user interface and website pages.  **Manual**:  1. The Registrar's Office sends physical copies of user information to the Admin. |
| **View OBE report and Mark Sheet** | **Faculty:**  1.Log into SPMS faculty dashboard  3. Specify the scope and duration for the report on the performance assessment (entire course or individual student)  4.Download report in PDF form  **Student:**  1.Log into SPMS student dashboard  2. View OBE mark sheet in browser.  3. Get details about the selected semester performance.  4.Download report in PDF form  **Higher Authorities:**  1. Log into SPMS Higher Authorities dashboard.  2. View OBE Mark sheets, searchable by year, courses, and faculty.  3. View individual student reports. | **Pen and paper:**  1. Used for note-taking. | **Cloud Server:**  1.Receive and process incoming requests  **Computer/ Phone:**  1. View reports.  2. Save reports from browser to device | **SPMS Faculty frontend:**  1.Provide user interface for online Faculty navigation  2. Show specific reports on request.  3. Sort report data in customizable ways (by PLO, by CO, by semester, by course, by time)  **SPMS Student frontend:**  1. Provide user interface for online Student navigation.  2. Show specific reports | **SPMS:**  1. Recover PLO/CO accomplishment statistics from SPMS and compile them for a given course and student(s).  2. The results and analyses are derived from the tabled data. | **Internet:**  1. SPMS is an online web application for which all packets and requests online are transmitted. |

**2.7: PROCESS DIAGRAM(TO BE)**

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

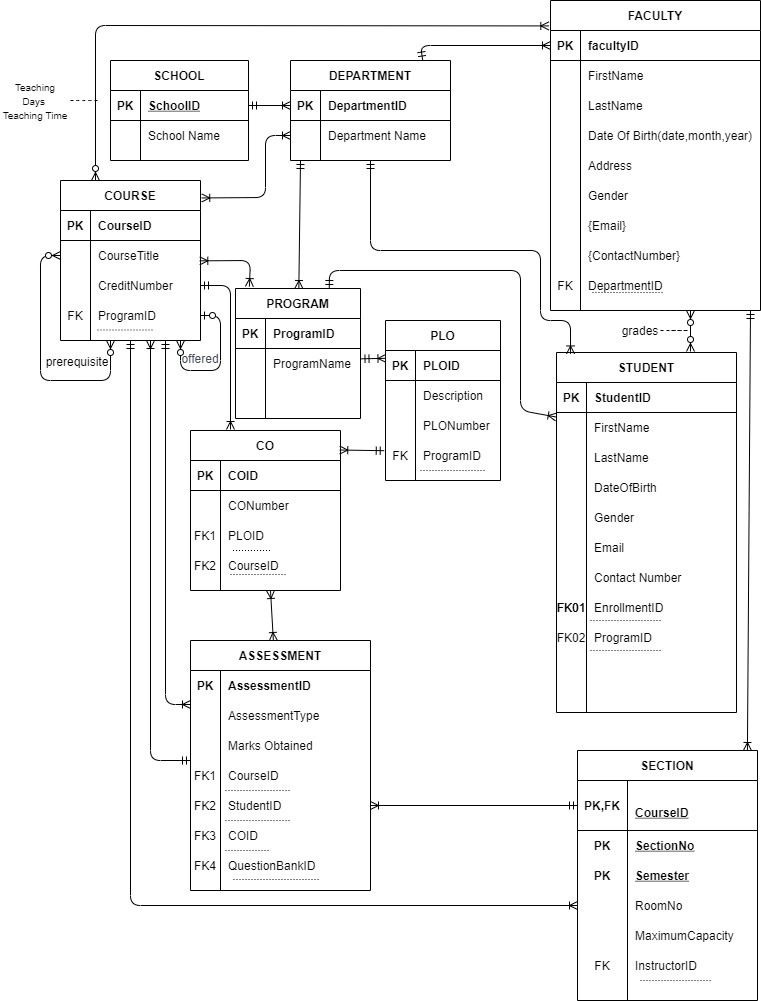
Description automatically generated

**CHAPTER 3: LOGICAL SYSTEM DESIGN**

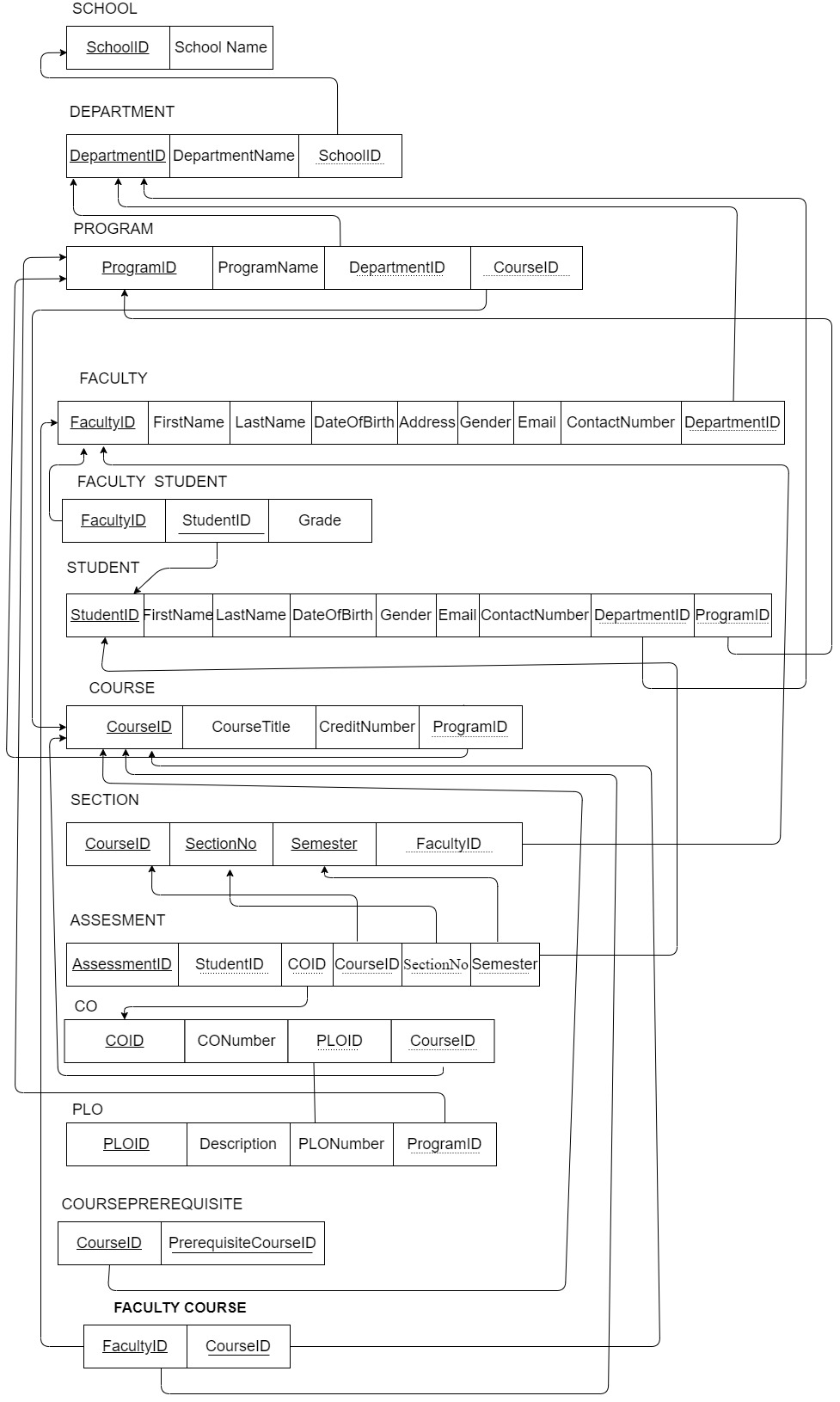
**3.1: BUSINESS RULES:**

1. An Accountis used to log in to our SPMS web application. An Accounthas a name (first name, last name), local address, phone number, email address, password (?), account type.
2. An account type can only be one of the following: Admin, Student, Faculty etc.
3. An Accreditation Manualwill have documentId (identifier), document title, document URL, upload date, upload time, year.
4. A Studentwill have a studentID, studentType (e.g graduate or undergraduate), cgpa, department, major(s), minor(s), date of admission, potential year for graduation.
5. A Studentmust belong to only one department, and a departmentmay have many students.
6. Each Studentmay register for multiple OfferedCourses. An Offered Coursemay be registered by multiple Students. Studentscan only register for exactly one Sectionof an Offered Course.
7. A Faculty may be assigned to multiple Sections of multiple Offered Courses. A Section will always have one Faculty assigned to it.
8. An Admin will have a role and tasks. Despite having more functionalities in SPMS than any other user type, their own attributes will not be related in any other way.
9. A Department has many Courses, but a Course belongs to exactly one Department.
10. 11. A Course has courseid, course name, course description and a credit hour count.
11. 12. A Course may have multiple courses as prerequisites, and a course may be a prerequisite of another course.
12. A Department can offer certain courses as Offered Courses every Semester.
13. 14. A Semester has multiple Offered Courses but Offered Courses will belong to only one Semester every semester. A Semester has semester season (e.g. Autumn , Summer, etc.), year, start date and end date.
14. A Course may have multiple other courses as their prerequisites, and a course may be a prerequisite of multiple Courses.
15. A Course Assessment belongs to only one Section of an Offered Course. A Course Assessment must have multiple Questions. With each Question mapped to a specific Course Outcomes. Course Assessment will have course assessment id, course assessment title, total marks.
16. A Question will have courseAssessmentID and questioNo as its identifier. It will also have full marks, description and only one assigned course ID because a Question may be mapped to only one Course outcome. A student answers many questions, and a question is answered by many students.
17. A Faculty may be assigned to a department.
18. A Course Outcome has course outcome id and courseID as its identifier. Other attributes include course outcome description, domain level and pass threshold.
19. An Offered Course may have multiple Course Outcomes.
20. A course outcome may be mapped to multiple questions within a course.
21. A course outcome can be assigned to exactly one Offered Course. Program Learning Outcomes must be assigned on at least one Course Outcomes. A Course Outcome must be assigned multiple Program Learning Outcomes.
22. A Program Learning Outcomes will have PLOid (identifier), PLO title and PLO description. A program learning outcome must be mapped to at least one course learning outcome.

**3.2: ENTITY RELATIONSHIP DIAGRAM**



**3.3: RELATIONAL SCHEMA**



**3.4: NORMALIZATION**

**Text

Description automatically generated**

**Text, application, table, Word

Description automatically generated**

**3.5: DATA DICTONARY**

**tblschool:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Size** | **Remark** |
| schoolid | varchar | 10 | This is the primary key of this relation. This contains the school id. Example: ‘SECS’ |
| schoolname | varchar | 100 | This contains the school name of a particular school. Example: ‘The School of Engineering & Computer Science’ |

**tbldepartment**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Size** | **Remark** |
| departmentid | varchar | 100 | This is the primary key of this relation. This contains the department ID. Example: ‘CSE’ |
| departmentname | varchar | 100 | This contains the name of a particular DEPARTMENT. Example: ‘Computer Science & Engineering’. |
| schoolid | varchar | 100 | This is a foreign key from Table SCHOOL |

**tblprogram**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Size** | **Remark** |
| programid | varchar | 100 | This is the primary key of this relation. This contains the program id. |
| programname | varchar | 100 | This contains the program name of a particular program. |
| departmentid | varchar | 100 | This is a foreign key from Table DEPARTMENT. |
| courseid | varchar | 100 | This is a foreign key from Table COURSE. |

**tblfaculty:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Size** | **Remark** |
| facultyid | varchar | 100 | This is the primary key of this relation. This contain the ID of instructor. Example: ‘4021’ |
| firstname | varchar | 100 | This is the first name of the instructor. Example: ‘Rafiq’ |
| lastname | varchar | 100 | This is the last name of the instructors. Example: ‘Mohammad’ |
| dateofbirth | varchar | 100 | This contains date of birth of the instructor. Example: ‘26/07/90’ |
| address | varchar | 100 | This is the home address of the instructor. |
| gender | varchar | 100 | This contains the gender of the instructors. Example: ‘Male’ or ‘Female’. |
| email | varchar | 100 | This is the email address of the instructor. |
| contactnumber | varchar | 100 | This is the contact number of the instructor. Example: ‘01930456781’. |
| departmentid | varchar | 100 | This is a foreign key from Table DEPARTMENT. |

**tblfacultystudent**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Size** | **Remark** |
| facutyid | varchar | 100 | This is the composite key of this relation. This contains the instructor ID. |
| studentid | varchar | 100 | This is the composite key of this relation. This contain the ID of student. Example: ‘1611014’ |
| Grade | varchar | 100 | This is the grade assigned by the Instructor. |

**tblstudent:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Size** | **Remark** |
| studentid | varchar | 100 | This is the primary key of this relation. This contain the ID of student. Example: ‘1611014’ |
| firstname | varchar | 100 | This is the first name of the student. Example: ‘Rafia’ |
| lastname | varchar | 100 | This is the last name of the student. Example: ‘Mohammad’ |
| dateofbirth | date | 9999-12-31 | This contains date of birth of the student. Example: ‘16/07/97’ |
| address | varchar | 100 | This is the home address of the student. |
| gender | varchar | 100 | This contains the gender of the student. Example: ‘Male’ or ‘Female’. |
| email | varchar | 100 | This is the email address of the student. |
| contactnumber | varchar | 100 | This is the contact number of the student. Example: ‘01930456781’. |
| departmentid | varchar | 100 | This is a foreign key from Table DEPARTMENT. |
| programid | varchar | 100 | This is a foreign key from Table PROGRAM. |

**tblcourse**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Size** | **Remark** | |
| couresid | varchar | 100 | This is the Primary Key for the Course.  Example: “CSE303” | |
| coursetitle | varchar | 100 | This is the name of the Course.  Example:” Database Management” | |
| creditnumber | varchar | 100 | This is the credit for the Course.  Example: “3.00” | |
| programid | varchar | 100 | This is the Foreign Key from Program table  Example: “B.Sc”. | |
| offeredcourseid | varchar | 100 | This is the list of course IDs that are offered in the semester. |

**tblsection**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data type** | **Size** | **Remark** |
| coureseid | varchar | 100 | This is the composite key. Example: “CSE303” |
| sectionno | varchar | 100 | This is the composite key. This is the section number.  Example: “1” |
| semester | varchar | 100 | This is the composite key. This is the semester.  Example: “May” |
| facultyid | varchar | 100 | This is the foreign key from the instructor  Table.  Example: “13400435” |

**tblassessment:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Data type** | **Size** | **Remark** | |
| assessmentid | varchar | 100 | This is the primary key of this relation. This contains the assessment id. | |
| assessmenttype | varchar | 100 | This contains the assessment marks. | |
| marksobtained | varchar | 100 | This contains the total marks achievable. | |
| courseid | varchar | 100 | This is a foreign key from Table COURSE. | |
| studentid | varchar | 100 | This is the foreign key of this relation. This contains the student id. | |
| coid | varchar | 100 | This is a foreign key from Table CO. | |
| sectionno | varchar | 100 | This is the foreign key from Table SECTION. This is the section number.  Example: “1” |
| semester | varchar | 100 | This is the foreign key from Table SECTION. This is the semester.  Example: “May” |

**tblco:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Size** | **Remark** |
| Coid | varchar | 100 | This is the primary key of this relation. This contains the co id. |
| conumber | varchar | 100 | This contains the co number of a particular course. Example: ‘Database Management’. |
| Ploid | varchar | 100 | This is a foreign key from Table PLO. |
| courseid | varchar | 100 | This is a foreign key from Table COURSE. |

**tplo**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Size** | **Remark** |
| ploid | varchar | 100 | This is the primary key of this relation. This contains the plo id. |
| description | varchar | 100 | This is the plo description. |
| plonumber | varchar | 100 | This contains the plo number of a particular course. Example: ‘Database Management’. |
| programid | varchar | 100 | This is a foreign key from Table PROGRAM. |

**tblcourseprerequisite:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Size** | **Remark** |
| courseid | varchar | 100 | This is the composite key of this relation. This contains the course id of a particular course. Example:’CSE303’ |
| prerequisitecourseid | varchar | 100 | This is the composite key of this relation. This contains the course id of the prerequisite course. |

**tblfacultycourse**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Size** | **Remark** |
| facultyid | varchar | 100 | This is the composite key of this relation. This contains the instructor ID . |
| courseid | varchar | 100 | This is the composite key of this relation.  Example: CSE303 |

**CHAPTER 4: PHYSICAL SYSTE DESIGN**

**4.1: FRONT END LOOK**

**A screenshot of a computer

Description automatically generated**

Figure 1: Users

**Graphical user interface, text, application

Description automatically generated**

Figure 2: Add new User

**Graphical user interface, application, email, Teams

Description automatically generated**

Figure 3 : Courses

Graphical user interface, application

Description automatically generated

Figure 4: Add New Course

A screenshot of a computer

Description automatically generated

Figure 5 : Program

Graphical user interface, application

Description automatically generatedFigure 6: Add new Program

Graphical user interface, application

Description automatically generated

Figure 7: Entry Marks

Graphical user interface, application, Teams

Description automatically generated

Figure 8: Entry Marks Mass

Graphical user interface, text, application

Description automatically generated

Figure 9: Higher Management Dashboard

Graphical user interface, text, application

Description automatically generated

Figure 10: Student Result

**4.2: RELATED SQL USED**

## Section 4.1.1: Input Forms – Add User

### Purpose

Here admin user will create other user profiles such as student, faculty etc.

<?php

    require 'mysql.php';

    // geting post requests

    $id = $\_POST['userID'];

    $fName = $\_POST['firstName'];

    $lName = $\_POST['lastName'];

    $prog = $\_POST['programID'];

    $email = $\_POST['email'];

    $password = $\_POST['password'];

    $role = strtolower($\_POST['role']);

   $store = "INSERT INTO tbluser (userID, firstName, lastName, programID, email, password, role) VALUES

          ('$id', '$fName', '$lName', '$prog', '$email', '$password' , '$role')";

    if($mysql->query($store)){

        header("Location: ../admin/add-user.php");

    }else{

        header("Location: ../admin/add-user.php");

    }

?>

## **Section 4.1.2: Input Forms – Add Program and PLO**

### Purpose

Since this system tracks PLO achievement of students of any courses in any program, so we must input the PLO data with their corresponding program. The system cannot collect the data automatically, so manually inputting the programs with their PLO’s are done using input form.

<?php

    require 'mysql.php';

    $id = $\_POST['programID'];

    $programName = $\_POST['programName'];

    $school = $\_POST['school'];

    $sql = "INSERT INTO tblprogram2(programID, programName, school) VALUES ('$id', '$programName', '$school')";

    //echo $sql;

    if($mysql->query($sql) == FALSE){

        header("Location: ../admin/add-program.php?failed=1");

    }

    $i = 1;

    while(isset($\_POST['title'.$i])){

        $name = $\_POST['title'.$i];

        $sql = "INSERT INTO plo(programID, plo\_no, plo\_name) VALUES

            ('$id', $i, '$name')";

        if($mysql->query($sql) == FALSE){

            header("Location: ../admin/add-program.php?failed=1");

        }

        $i++;

    }

    header("Location: ../admin/add-program.php?success=1");

?>

**Section 4.1.3: Input Forms – Add Course with CO**

Purpose

Each individual course must be added to the system with their respective CO’s. In order to do that, an input form must be used to give inputs to the system.

<?php

    require 'mysql.php';

    $id = $\_POST['courseID'];

    $program\_id = $\_POST['programID'];

    $credit = $\_POST['credit'];

    $total\_co = $\_POST['totalCO'];

    $title = $\_POST['courseTitle'];

    $sql = "INSERT INTO tblcourse (courseID, programID, courseTitle, credit, totalCO) VALUES

            ('$id', '$program\_id', '$title', $credit, $total\_co)";

    if($mysql->query($sql) == FALSE){

        header("Location: ../admin/add-course.php?failed=1");

    }

    for($i=1; $i<=15; $i++){

        if(isset($\_POST["plo-co".$i])){

            $sql = "SELECT sl FROM tblplo WHERE programID = '$programID' AND ploNo = $i";

            $plo\_id = $mysql->query($sql)->fetch\_assoc()['sl'];

            $data = $\_POST["plo-co".$i];

            $field = ""; $val ="";

            foreach($data as $co){

                $field .= 'co'. $co . ', ';

                $val .= '1, ';

            }

            $sql = "INSERT INTO tblco (courseID, ploID, ".substr($field, 0, -2).") VALUES ('$id', $plo\_id, ".substr($val, 0, -2).")";

            if($mysql->query($sql) == FALSE){

                header("Location: ../admin/add-course.php?failed=1");

            }

        }

    }

    header("Location: ../admin/add-course.php?success=1");

?>

**Section 4.1.4: Input Forms – Add Marks Individual**

### Purpose

This input form requires marks entry process with which a students’ data can be entered based on a course in a particular semester. This is then stored in the database which can be retrieved when required. Marks entry form can input all the assessments of a particular student.

    require 'mysql.php';

    $student\_id = $\_POST['studentID'];

    $course\_id = $\_POST['courseID'];

    $exam\_name = $\_POST['examName'];

    $semester = $\_POST['semester'];

    $section = $\_POST['section'];

    $field = "";

    $val = "";

    $i=1;

    while(isset($\_POST['tblco'.$i])){

        $field .= 'q'.$i.'\_mark, ' . 'q'.$i.'\_co, ' . 'q'.$i.'\_max, ';

        $val .= $\_POST['mark'.$i].', '.$\_POST['co'.$i].', '.$\_POST['max'.$i].', ';

        $i++;

    }

    $sql = "INSERT INTO tblmarks(studentID, courseID, examName, semester, section, ".substr($field, 0, -2).") VALUES

            ('$student\_id', '$course\_id', '$exam\_name', '$semester', '$section', ".substr($val, 0, -2).")";

    echo $sql . '<br>';

    $mysql->query($sql);

    header("Location: ../faculty/entry-marks.php?success=1");

**Section 4.1.5: Input Forms – Add Marks Mass**

Purpose

This input form requires marks entry process with which many students data can be entered on any course in any semester. This is then stored in the database which can be retrieved when required. Marks entry form can input all the assessments of all the students using a .csv format file.

<?php

    require 'mysql.php';

    $course\_id = $\_POST['courseID'];

    $exam\_name = $\_POST['examName'];

    $semester = $\_POST['semester'];

    $section = $\_POST['section'];

    $file = fopen($\_FILES['file']['tmp\_name'], "r");

    fgetcsv($file, 200); // dumping header

    $co = array();

    $f = 1; $i=1;

    foreach(fgetcsv($file, 200) as $c){

        if($f == 1){

            $f = 0;

            continue;

        }

        $co[$i] = $c;

        $i++;

    }

    $max = array();

    $f = 1; $i=1;

    foreach(fgetcsv($file, 200) as $m){

        if($f == 1){

            $f = 0;

            continue;

        }

        $max[$i] = $m;

        $i++;

    }

    while($marks = fgetcsv($file, 200)){

        $student\_id;

        $field = "";

        $val = "";

        $f = 1; $i=1;

        foreach($marks as $m){

            if($f==1){

                $student\_id = $m;

                $f=0;

                continue;

            }

            $field .= 'q'.$i.'\_mark, ' . 'q'.$i.'\_co, ' . 'q'.$i.'\_max, ';

            $val .= $m.', '.$co[$i].', '.$max[$i].', ';

            $i++;

        }

        $sql = "INSERT INTO tblmarks(studentID, courseID, examName, semester, section, ".substr($field, 0, -2).") VALUES

            ('$student\_id', '$course\_id', '$exam\_name', '$semester', '$section', ".substr($val, 0, -2).")";

        $mysql->query($sql);

    }

    header("Location: ../faculty/entry-marks-mass.php?success=1");

?>

**Section 4.2.1: Output Query and Reports – Higher Management Dashboard**

Purpose and Use

It gives the overall summary of the system which includes the total number of courses that uses OBE model, total students who are evaluated with OBE model, number of faculty evaluated using OBD, total number of PLO set in the program. It can be used to gather information about an institution by the higher management.

<?php

    require 'php/middleware.php';

    require 'php/mysql.php';

    $sql = "SELECT \* FROM tblcourse";

    $course = $mysql->query($sql)->num\_rows;

    $sql = "SELECT \* FROM tbluser WHERE role='student'";

    $student = $mysql->query($sql)->num\_rows;

    $sql = "SELECT \* FROM tbluser WHERE role='faculty'";

    $faculty = $mysql->query($sql)->num\_rows;

    $sql = "SELECT \* FROM tblplo";

    $plo = $mysql->query($sql)->num\_rows;

?>

**Section 4.2.2: Output Query and Reports – PLO Achievement**

Purpose and Use

It is used to show student wise PLO analysis which includes PLO total percentage score for each PLO calculated from the scores achieved in each CO associated with the corresponding PLO among all the courses the student has done so far. Upon entering a student id course wise PLO analysis can be viewed. Also, a tabular view of student wise PLO achievement can be viewed.

<?php

    require 'mysql.php';

    if(isset($\_GET['studentID'])){

        $id = $\_GET['studentID'];

        $sql = "SELECT \* FROM tblmarks WHERE studentID = $id";

        $sMarks = $mysql->query($sql);

        //course based total co marks

        $cMarks = array();

        $cTotal = array();

        foreach($sMarks as $marks){

            $course = $marks['courseID'];

            for($i=1; $i<=10; $i++){

                if(isset($marks["q".$i."\_co"]) && $marks["q".$i."\_co"]!=0){

                    $co = $marks["q".$i."\_co"];

                    if(isset($cMarks[$course][$co])){

                        $cMarks[$course][$co] += $marks["q".$i."\_mark"];

                        $cTotal[$course][$co] += $marks["q".$i."\_max"];

                    }else{

                        $cMarks[$course][$co] = $marks["q".$i."\_mark"];

                        $cTotal[$course][$co] = $marks["q".$i."\_max"];

                    }

                }

            }

        }

        $pMarks = array();

        $pTotal = array();

        foreach($cMarks as $c => $v){

            $sql = "SELECT \* FROM tblco WHERE courseID = '$c'";

            $plos = $mysql->query($sql);

            foreach($plos as $plo){

                $pId = $plo['ploID'];

                for($i=1; $i<=10; $i++){

                    if(isset($plo["co".$i]) && $plo["co".$i]==1){

                        if(isset($pMakrs[$c][$pId])){

                            $pMarks[$c][$pId] += $cMarks[$c][$i];

                            $pTotal[$c][$pId] += $cTotal[$c][$i];

                        }else{

                            $pMarks[$c][$pId] = $cMarks[$c][$i];

                            $pTotal[$c][$pId] = $cTotal[$c][$i];

                        }

                    }

                }

            }

        }

        //total marks in plo

        $pfMarks = array();

        $pfTotal = array();

        foreach($pMarks as $c => $v){

            foreach($v as $i => $j){

                if(isset($pfMarks[$i])){

                    $pfMarks[$i] += $j;

                    $pfTotal[$i] += $pTotal[$c][$i];

                }else{

                    $pfMarks[$i] = $j;

                    $pfTotal[$i] = $pTotal[$c][$i];

                }

            }

        }

        //student info

        $sql = "SELECT \* FROM tbluser WHERE userID = $id";

        $student = $mysql->query($sql)->fetch\_assoc();

        //total plo

        $sql = "SELECT \* FROM tblplo WHERE programID = '".$student['programID']."'";

        $ploNum = $mysql->query($sql)->num\_rows;

        $color = ["", "#1FE7C4", "#E45C17", "#06B97B", "#8CE026", "#E1CCFF", "#5BA2CC", "#0A2E82", "#957107", "#80CF18"];

    }

?>

**Section 4.2.3: Output Query and Reports – Progress View**

Purpose and Use

It contains student and course progress views. For a given student, it shows the count of PLO’s expected to be achieved and the counts achieved at the end of each semester. Upon selecting a certain course, it shows the number of students in that course with the percentages of CO achieved of failed.

<?php

    require 'mysql.php';

    if(isset($\_GET['studentID'])){

        $id = $\_GET['studentID'];

        $sql = "SELECT DISTINCT semester FROM tblmarks WHERE studentID = $id";

        $sems =  $mysql->query($sql);

        $ploProg = array();

        foreach($sems as $sem){

            $res = seeker($sem['semester'], $id);

            $ploProg[$sem['semester']]['total'] = $res['p'];

            $ploProg[$sem['semester']]['com'] = $res['t'];

        }

    }if(isset($\_GET['c'])){

        $crs = $\_GET['c'];

        $sql = "SELECT DISTINCT studentID FROM tblmarks WHERE courseID = '$crs'";

        $uList = $mysql->query($sql);

        $totalS = $uList->num\_rows;

        $report = array();

        foreach($uList as $u){

            $usr = $u['studentID'];

            $ret = seeker2($crs, $usr);

            foreach($ret["co"] as $i => $j){

                if($j==1){

                    if(isset($report["co"][$i])){

                        $report["co"][$i]++;

                    }else{

                        $report["co"][$i] = 1;

                    }

                }

            }

            foreach($ret["plo"] as $i => $j){

                if($j==1){

                    if(isset($report["plo"][$i])){

                        $report["plo"][$i]++;

                    }else{

                        $report["plo"][$i] = 1;

                    }

                }

            }

        }

        ksort($report["co"]);

        ksort($report["plo"]);

    }

    function seeker($sem, $uid){

        require 'mysql.php';

        $sql;

        if($sem!="null"){

            $sql = "SELECT \* FROM tblmarks WHERE studentID = $uid AND semester = '$sem'";

        }else{

            $sql = "SELECT \* FROM tblmarks WHERE studentID = $uid";

        }

        $sMarks = $mysql->query($sql);

        $cMarks = array();

        $cTotal = array();

        foreach($sMarks as $marks){

            $course = $marks['courseID'];

            for($i=1; $i<=10; $i++){

                if(isset($marks["q".$i."\_co"]) && $marks["q".$i."\_co"]!=0){

                    $co = $marks["q".$i."\_co"];

                    if(isset($cMarks[$course][$co])){

                        $cMarks[$course][$co] += $marks["q".$i."\_mark"];

                        $cTotal[$course][$co] += $marks["q".$i."\_max"];

                    }else{

                        $cMarks[$course][$co] = $marks["q".$i."\_mark"];

                        $cTotal[$course][$co] = $marks["q".$i."\_max"];

                    }

                }

            }

        }

        $pMarks = array();

        $pTotal = array();

        foreach($cMarks as $c => $v){

            $sql = "SELECT \* FROM tblco WHERE courseID = '$c'";

            $plos = $mysql->query($sql);

            foreach($plos as $plo){

                $pId = $plo['ploID'];

                for($i=1; $i<=10; $i++){

                    if(isset($plo["co".$i]) && $plo["co".$i]==1){

                        if(isset($pMakrs[$c][$pId])){

                            $pMarks[$c][$pId] += $cMarks[$c][$i];

                            $pTotal[$c][$pId] += $cTotal[$c][$i];

                        }else{

                            $pMarks[$c][$pId] = $cMarks[$c][$i];

                            $pTotal[$c][$pId] = $cTotal[$c][$i];

                        }

                    }

                }

            }

        }

        $res = array();

        $res['t'] = 0;

        $pTrack = array();

        foreach($pMarks as $c => $v){

            foreach($v as $i => $j){

                $pTrack[$i]=1;

                if($j \* 100 / $pTotal[$c][$i]>=40){

                    $res['t']++;

                }

            }

        }

        $res['p'] = count($pTrack);

        return $res;

    }

    function seeker2($crs, $uid){

        require 'mysql.php';

        $sql = "SELECT \* FROM tblmarks WHERE studentID = $uid AND courseID = '$crs'";

        $sMarks = $mysql->query($sql);

        $cMarks = array();

        $cTotal = array();

        foreach($sMarks as $marks){

            $course = $marks['courseID'];

            for($i=1; $i<=10; $i++){

                if(isset($marks["q".$i."\_co"]) && $marks["q".$i."\_co"]!=0){

                    $co = $marks["q".$i."\_co"];

                    if(isset($cMarks[$course][$co])){

                        $cMarks[$course][$co] += $marks["q".$i."\_mark"];

                        $cTotal[$course][$co] += $marks["q".$i."\_max"];

                    }else{

                        $cMarks[$course][$co] = $marks["q".$i."\_mark"];

                        $cTotal[$course][$co] = $marks["q".$i."\_max"];

                    }

                }

            }

        }

        $pMarks = array();

        $pTotal = array();

        foreach($cMarks as $c => $v){

            $sql = "SELECT \* FROM tblco WHERE courseID = '$c'";

            $plos = $mysql->query($sql);

            foreach($plos as $plo){

                $pId = $plo['ploID'];

                for($i=1; $i<=10; $i++){

                    if(isset($plo["co".$i]) && $plo["co".$i]==1){

                        if(isset($pMakrs[$c][$pId])){

                            $pMarks[$c][$pId] += $cMarks[$c][$i];

                            $pTotal[$c][$pId] += $cTotal[$c][$i];

                        }else{

                            $pMarks[$c][$pId] = $cMarks[$c][$i];

                            $pTotal[$c][$pId] = $cTotal[$c][$i];

                        }

                    }

                }

            }

        }

        $stats = array();

        foreach($cMarks as $c => $v){

            foreach($v as $i => $m){

                if(($m \* 100 / $cTotal[$c][$i]) >=40 ){

                    $stats["co"][$i] = 1;

                }else{

                    $stats["co"][$i] = 0;

                }

            }

        }

        foreach($pMarks as $p => $v){

            foreach($v as $i => $m){

                if(($m \* 100 / $pTotal[$p][$i]) >=40 ){

                    $stats["plo"][$i] = 1;

                }else{

                    $stats["plo"][$i] = 0;

                }

            }

        }

        return $stats;

    }

?>

**Section 4.2.4: Output Query and Reports – Student Result**

Purpose and Use

It is used to show the students result of PLO achievement in a pie chart for all the courses that student has completed upon entering the students id.

<?php

    require '../php/mysql.php';

    $color = ["", "#1FE7C4", "#E45C17", "#06B97B", "#8CE026", "#E1CCFF", "#5BA2CC", "#0A2E82", "#957107", "#80CF18"];

    // session\_start();

    $id = $\_SESSION['studentID'];

    $sql = "SELECT \* FROM tbluser WHERE userID = $id";

    $uInfo = $mysql->query($sql)->fetch\_assoc();

    $sql = "SELECT \* FROM tblplo WHERE programID = '".$uInfo['programID']."'";

    $totalPlo = $mysql->query($sql)->num\_rows;

    $sql = "SELECT \* FROM tblmarks WHERE studentID = $id";

    $sMarks = $mysql->query($sql);

        //course based total co marks

    $cMarks = array();

    $cTotal = array();

    foreach($sMarks as $marks){

        $course = $marks['courseID'];

        for($i=1; $i<=10; $i++){

            if(isset($marks["q".$i."\_co"]) && $marks["q".$i."\_co"]!=0){

                $co = $marks["q".$i."\_co"];

                if(isset($cMarks[$course][$co])){

                    $cMarks[$course][$co] += $marks["q".$i."\_mark"];

                    $cTotal[$course][$co] += $marks["q".$i."\_max"];

                }else{

                    $cMarks[$course][$co] = $marks["q".$i."\_mark"];

                    $cTotal[$course][$co] = $marks["q".$i."\_max"];

                }

            }

        }

    }

    $pMarks = array();

    $pTotal = array();

    foreach($cMarks as $c => $v){

        $sql = "SELECT \* FROM tblco WHERE courseID = '$c'";

        $plos = $mysql->query($sql);

        foreach($plos as $plo){

            $pId = $plo['ploID'];

            for($i=1; $i<=10; $i++){

                if(isset($plo["co".$i]) && $plo["co".$i]==1){

                    if(isset($pMakrs[$c][$pId])){

                        $pMarks[$c][$pId] += $cMarks[$c][$i];

                        $pTotal[$c][$pId] += $cTotal[$c][$i];

                    }else{

                        $pMarks[$c][$pId] = $cMarks[$c][$i];

                        $pTotal[$c][$pId] = $cTotal[$c][$i];

                    }

                }

**CHAPTER 5: CONCLUSION**

**Analysis Phase**

While creating the rich picture and six elements' analyses, assumptions were made about the organization’s operations. This required requalification from the instructor and thus slowed the pace. Following this, queries are made more explicitly when collecting information from a stakeholder.

**Design Phase**

A large percentage of the initially created entities (and their respective relations) were irrelevant to the needs of the project. Entities such as Grading Policy and Grade Change request had to be removed, and the final iteration of the ERD and Logical Schema were made while keeping only the bare minimum requirements in mind.

**Problem & Solution**

Since we had lack of knowledge on the language (PHP, CSS, JAVASCRIPT, HTML, SQL) so we have faced problem while creating Student Performance Monitoring System. We also had issues on using GitHub which was a new platform for us. So, we tried to gain some help doing some internet searches and gaining information ourselves and using it properly.

**Feature & Future Development**

Future plans for this project are to finish implementing the functionalities that are missing in the implementation stage.