CHITKARA UNIVERSITY

FA4 PROJECT

Institute InfoPanel

An Institute Management System

Team Leader:

- UJJWAL (2010991754)

Team Members:

- UJJWAL (2010991754)
- URVASHI CHOUBEY (2010991756)
- VIPUL KUMAR (2010991776)

Guide:

- Dr Vikas Solanki

Abstract

Institute InfoPanel deals with the maintenance of any Institute's faculty, student information within the Institute. Institute InfoPanel has a relational database which is used to store the college, faculty, student, courses and other information of institute. Starting from registration of a new student in the institute, it maintains all the details regarding the attendance and marks of the students. The project deals with retrieval of information through an internet-based campus wide portal.

Institute InfoPanel focuses on the basic need of accomplishing the task of maintaining the large stock of information in Institute by creating a database.

This project report will provide a detailed account of the functionalities of the user interface which is taken as a reference to manage a institute. Each subsection of this phase report will feature the important functionalities of the database design.

Development process of the system starts with System analysis. System analysis involves creating a formal model of the problem to be solved by understanding requirements.

Acknowledgements

We would like to express our gratitude to all of those who made it possible to complete this Database Management Project , in particular to our supervisor Dr. Vikas Solanki .

We would also like to thank our family and team members for their continuous support and guidance .

Table Of Contents

Abstract 2

Acknowledgements 3

Chapter 1: Introduction 5

- **Database Management Systems** 5
- **Relational database management systems** 5

Normalization 7

Objective 7

Chapter 2: Institute InfoPanel 8

2.1 Informal Description 8

2.2 Logical Model 13

2.3 Physical Model 14

2.4 Interactive Queries 27

Chapter 3: Conclusion & Future Work 34

BIBLIOGRAPHY 35

Chapter 1: Introduction

- Database Management Systems (DBMS) are software systems used to store, retrieve, and run queries on data. A DBMS serves as an interface between an end-user and a database, allowing users to create, read, update, and delete data in the database.

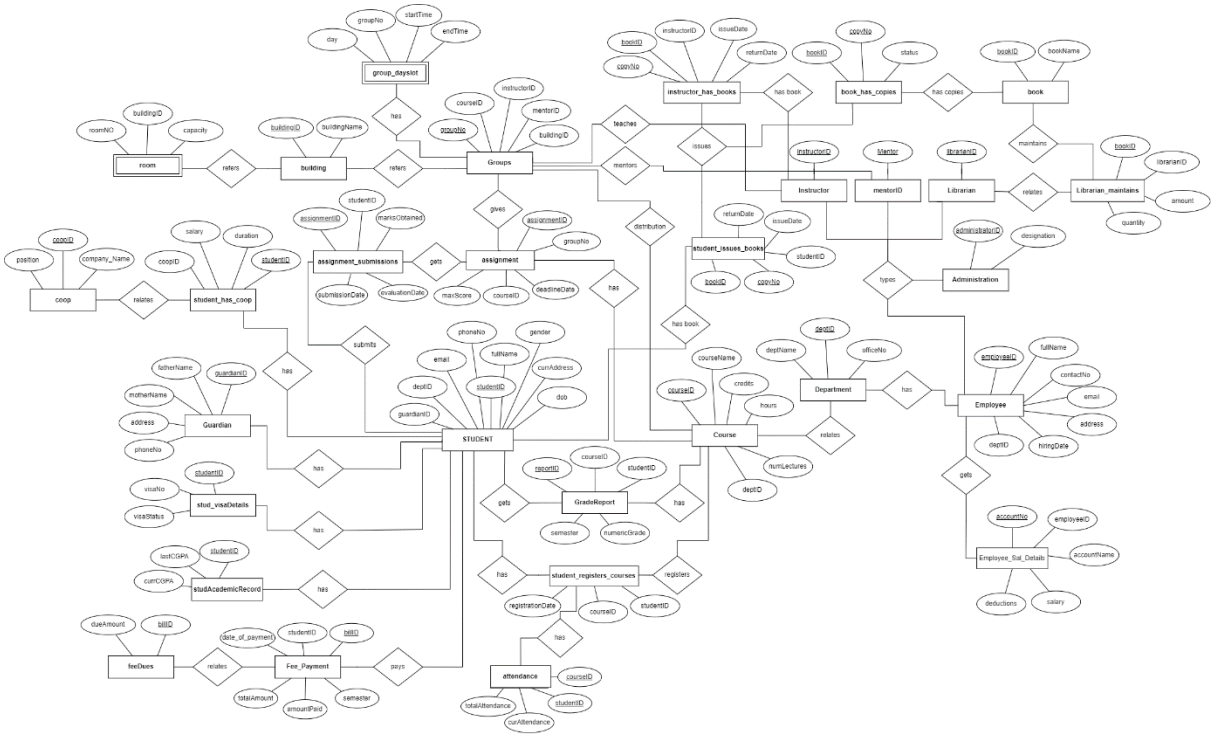
  DBMS manage the data, the database engine, and the database schema, allowing for data to be manipulated or extracted by users and other programs. This helps provide data security, data integrity, concurrency, and uniform data administration procedures.

  DBMS optimizes the organization of data by following a database schema design technique called normalization, which splits a large table into smaller tables when any of its attributes have redundancy in values. DBMS offer many benefits over traditional file systems, including flexibility and a more complex backup system.

  Database management systems can be classified based on a variety of criteria such as the data model, the database distribution, or user numbers. The most widely used types of DBMS software are relational, distributed, hierarchical, object-oriented, and network.

- Relational database management systems (RDBMS) are the most popular data model because of its user-friendly interface. It is based on normalizing data in the rows and columns of the tables. This is a viable option when you need a data storage system that is scalable, flexible, and able to manage lots of information.

- Institute InfoPanel is an Institute Database Management System that has been implemented using Oracle 11g XE Database. The purpose of this Project is to implement the CRUD (Create, Read, Update, Delete) operations related to database on a Real-Time example. It allows you to keep the normalised student records and manage them when needed.



Entity-Relationship (ER) model of our project is constructed using diagrams.net , an open- source cross-platform graph drawing software .
The oracle database schema is produced based on the ER model.

➢ Feasibility study :-

- ### Technical Feasibility :-

  We can strongly say that it is technically feasible, since there will not be much difficulty in getting required resources for the development and maintaining the system as well. All the resources needed for the development of the software as well as the maintenance of the same is available in the organization and are mentioned below :-

  - Oracle 11g XE
  - 64-bit Win10/11/Linux OS

- ### Economical Feasibility :-

  Development of this application is highly economically feasible .The organization need not to spend much money for the development of this system. The only thing is to be done is making an environment for the

development with an effective supervision. If we are doing so, we can attain the maximum usability of the corresponding resources .Even after the development, the organization will not be in condition to invest more in the organization .Therefore, the system is economically feasible.

Normalization

- The attribute studentID is removed from the coop table and is placed in another table student_has_coop to normalize the tables.
- The salary details of an employee are removed from employee table and is taken separately in table Employee_Sal_Details where the details of salary slip of each employee will be recorded.
- The academic record is removed from the student table as there existed partial dependencies of attributes on academic records. Hence , the table is in 3NF.
- The info of copies of books are removed from the table Books and are taken into new table book_has_copies since a book can have multiple copies.
- A course can be taught in many groups (sections) so this info is placed in another table Groups.
- A group can have lectures in different buildings at different timings , hence , a new table named group_dayslot is created. Similarly , to remove other dependencies and complexities building and rooms information is also stored in different tables.

Objective :- The main objective of this project is to store and maintain the student records efficiently. Without a Student information System, managing and maintaining the details of the student is a tedious job for any organization. The database will store all the details of the students including their academic records , other necessary details. At administrator side , we manage and update all the records.

Chapter 2: Institute InfoPanel

2.1 Informal Description

- PURPOSE OF THE SYSTEM:

  Institute InfoPanel deals with the maintenance of institute faculties, students and other information. This project involved the automation of student information that can be implemented in different institute managements.

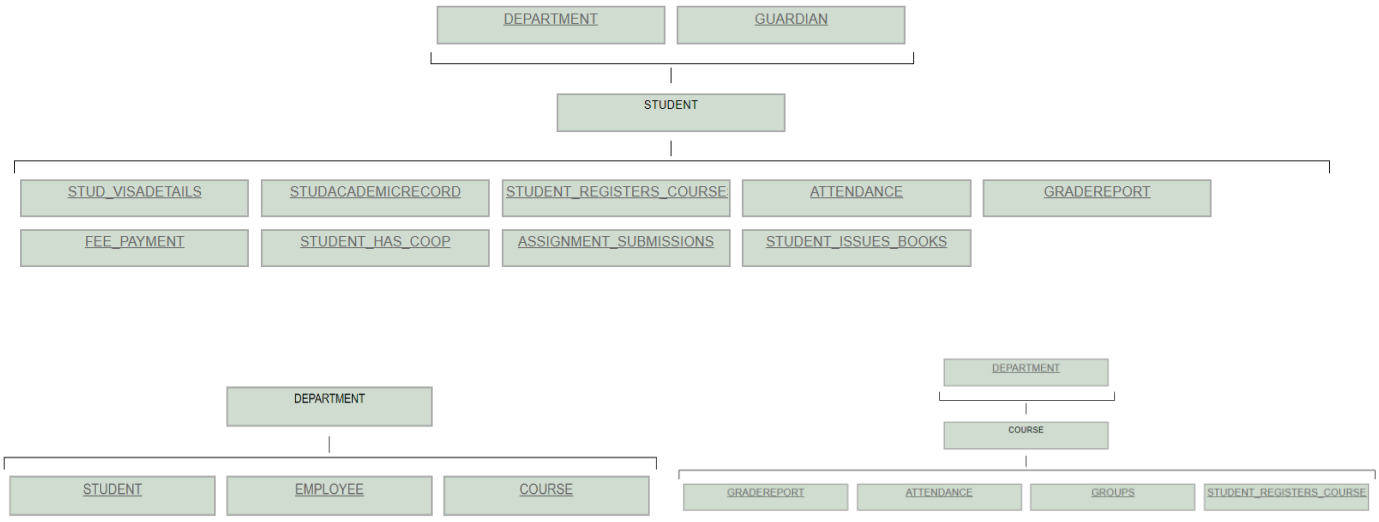
  The project deals with retrieval of information using database. It collects related information from all the departments of an organization and maintains database, which are used to generate reports in various forms to measure individual and overall performance of the students.

- PROPOSED SYSTEM:
  - Institute InfoPanel makes management to get the most updated information always by avoiding manual accounting process.
  - This system has the following functional divisions:
    - Administrator User (Students / Faculties /Department Staff) University Administrator has the functionality of registering new colleges and courses. He has the rights of creating department, allocating courses to departments, creating faculties, students and allocating subjects to faculties and modifications in the data entered by the user can also be done by the college administrator. User of this may be faculty or students or department staff.

    - Faculty has the facility of entering the marks and attendance of the students. Students can check their marks and attendance but there is no chance of modifications.

    - Department staff can maintain records respective to their roles. Reports must be generated for the existing data i.e., for attendance and marks of the students, which are used to assess

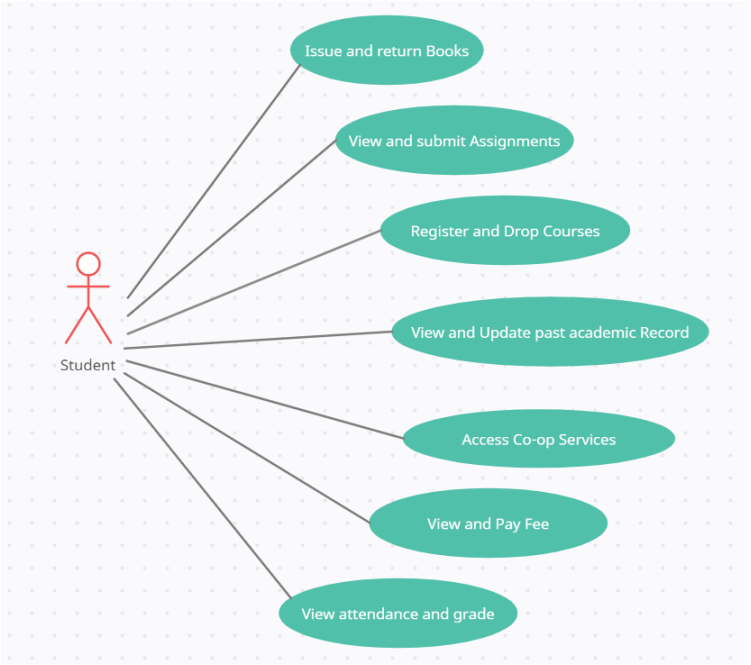
the performance of the students. These reports should be viewed by the faculty and user.



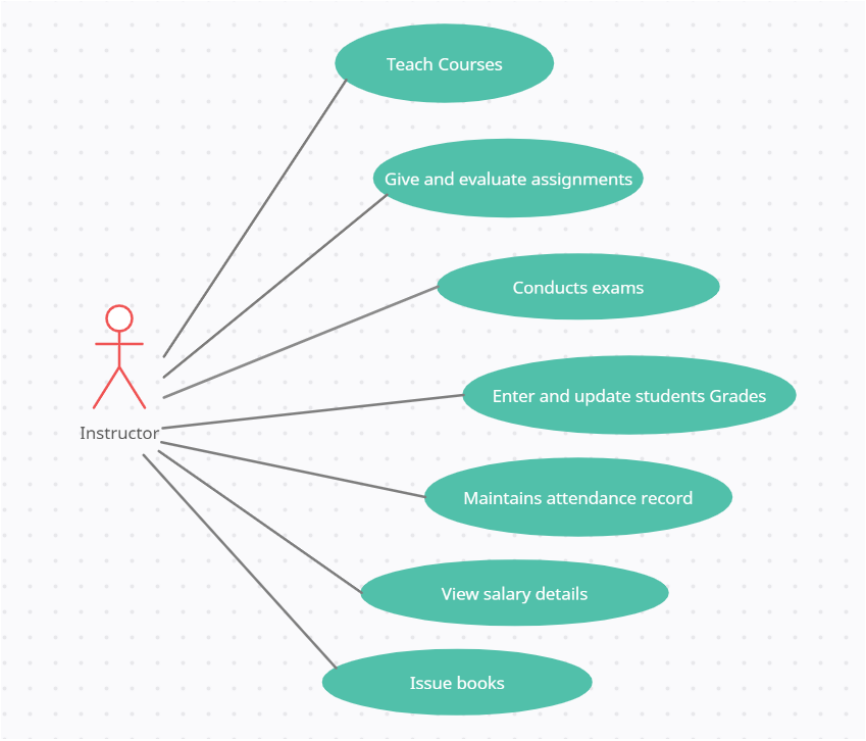
➢ ROLES AND RESPONSIBILITIES:

There are various roles played by different people:-

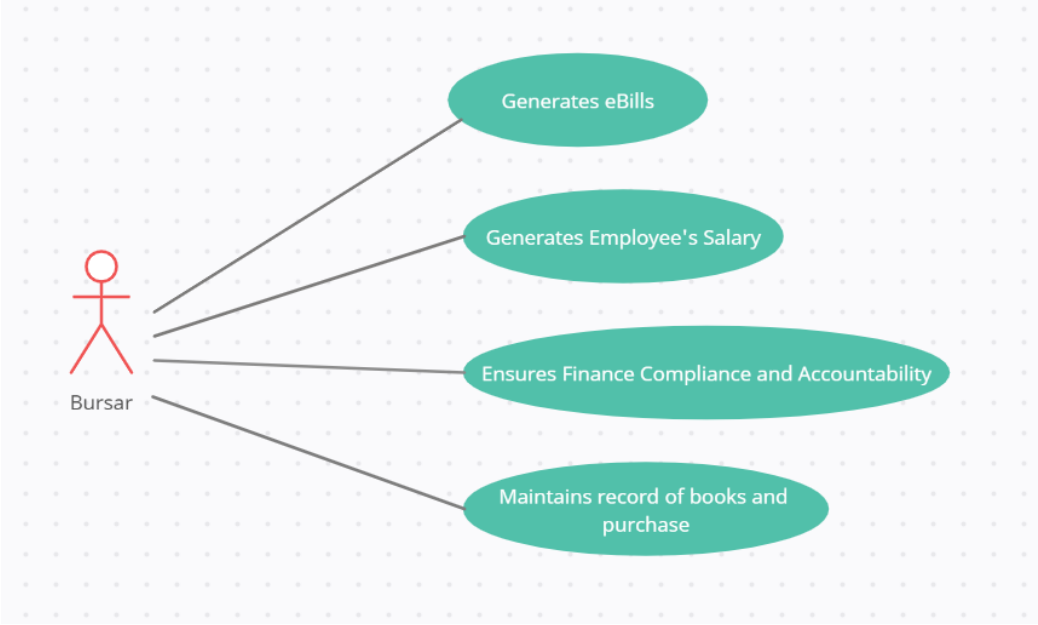
a) **Student:** The person who uses the application to interact with institute and associated employees, access his/her career opportunities & also get professional counselling from experienced mentors. It is instructed by instructors by taking section under various courses offered by the institution. It maintains attendance, undertakes examination and pass courses while submitting assignment and minimum grades.



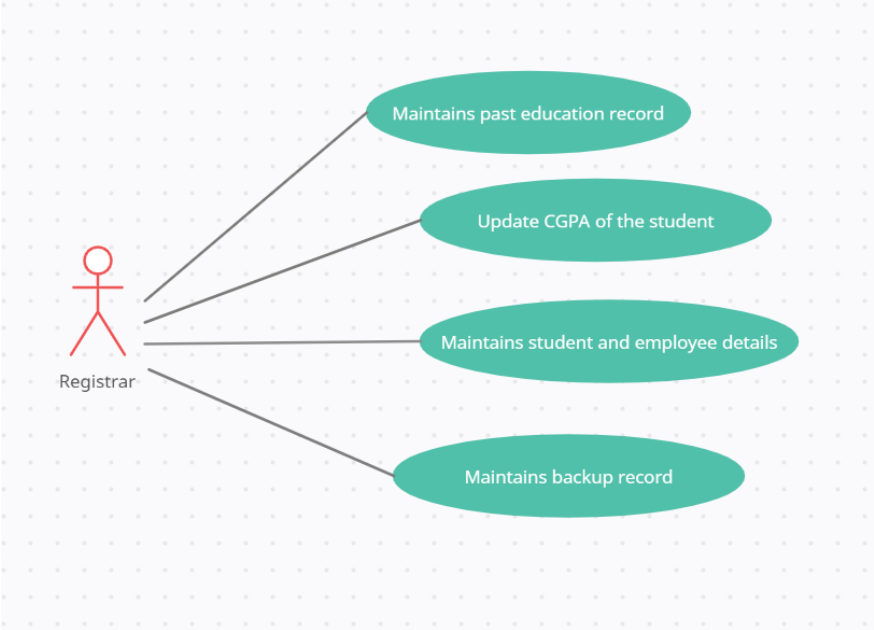
b) **Instructor:** The person who teaches students and is employed under a department, works under administration supervision and works for student welfare.



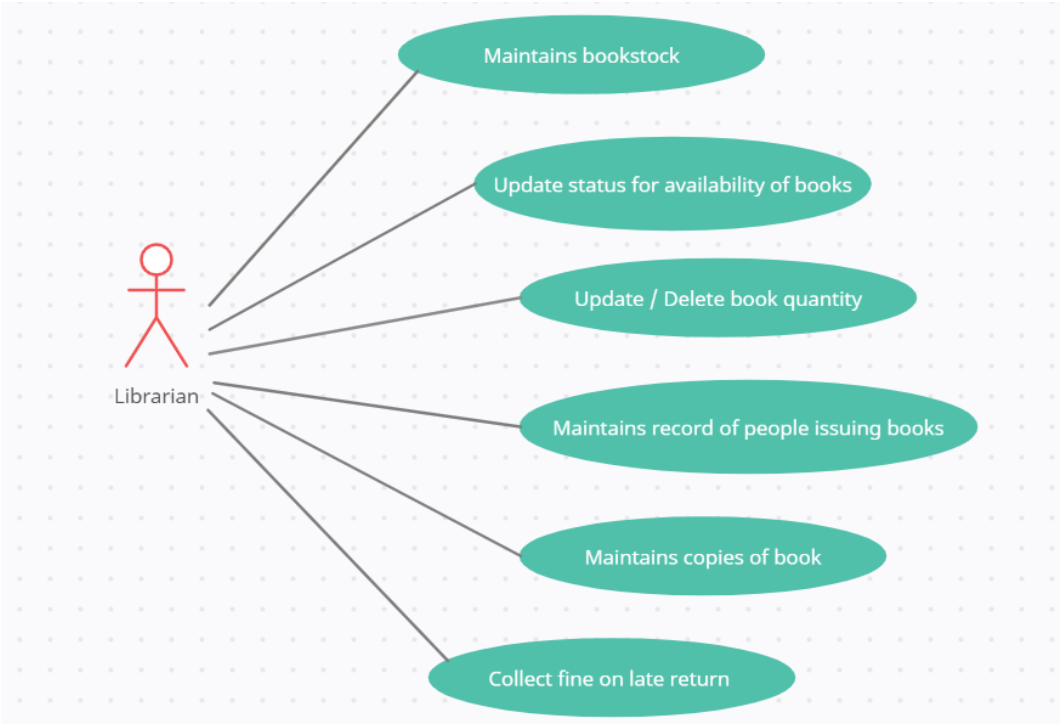
c) **Bursar:** The person who supervises the fiscal matter of the institution employees and student, responsible for generation of bills and remitting salary on monthly basis. It also keeps an eye on the dues of students.



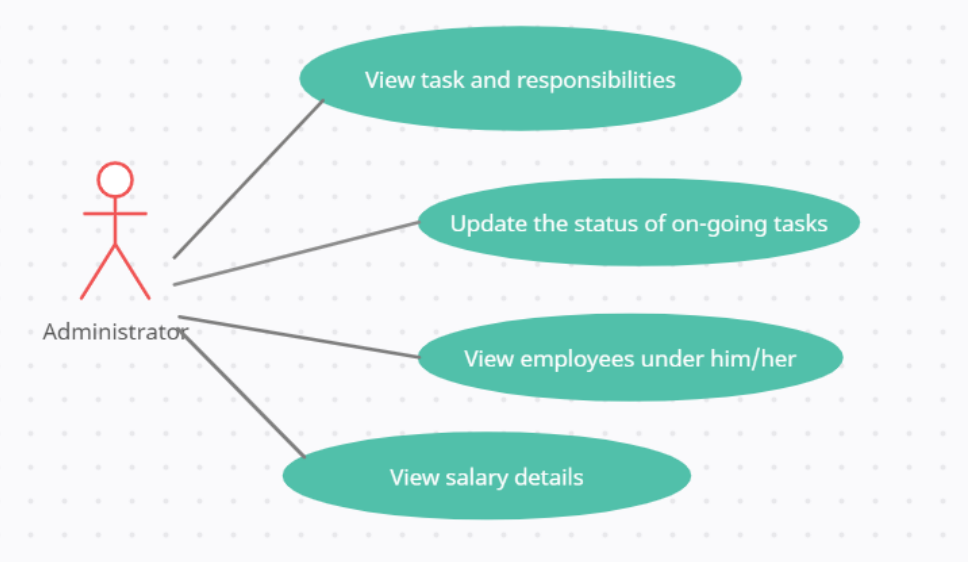
d) **Registrar:** The person who maintains student details. It also keeps a record of employee and administration staff. It also maintains a backup of the data in case it is lost. It is also responsible for updating the grades and marks of students in accordance with his registered courses.



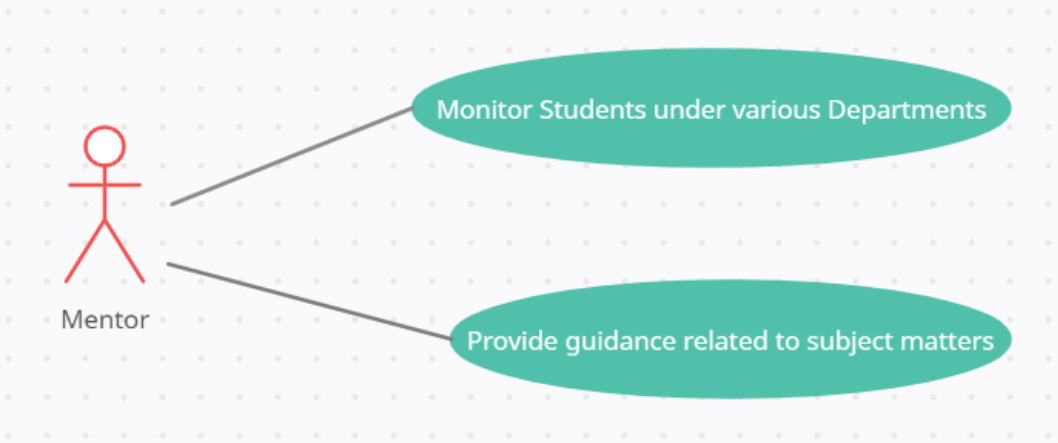
e) **Librarian:** The person who maintains the library, purchases books to serve as reference material for the students as well as instructors, maintains the quantity of books as per the need and demand. It maintains a record of people who issues books, and update the status of availability for the convenience of students .



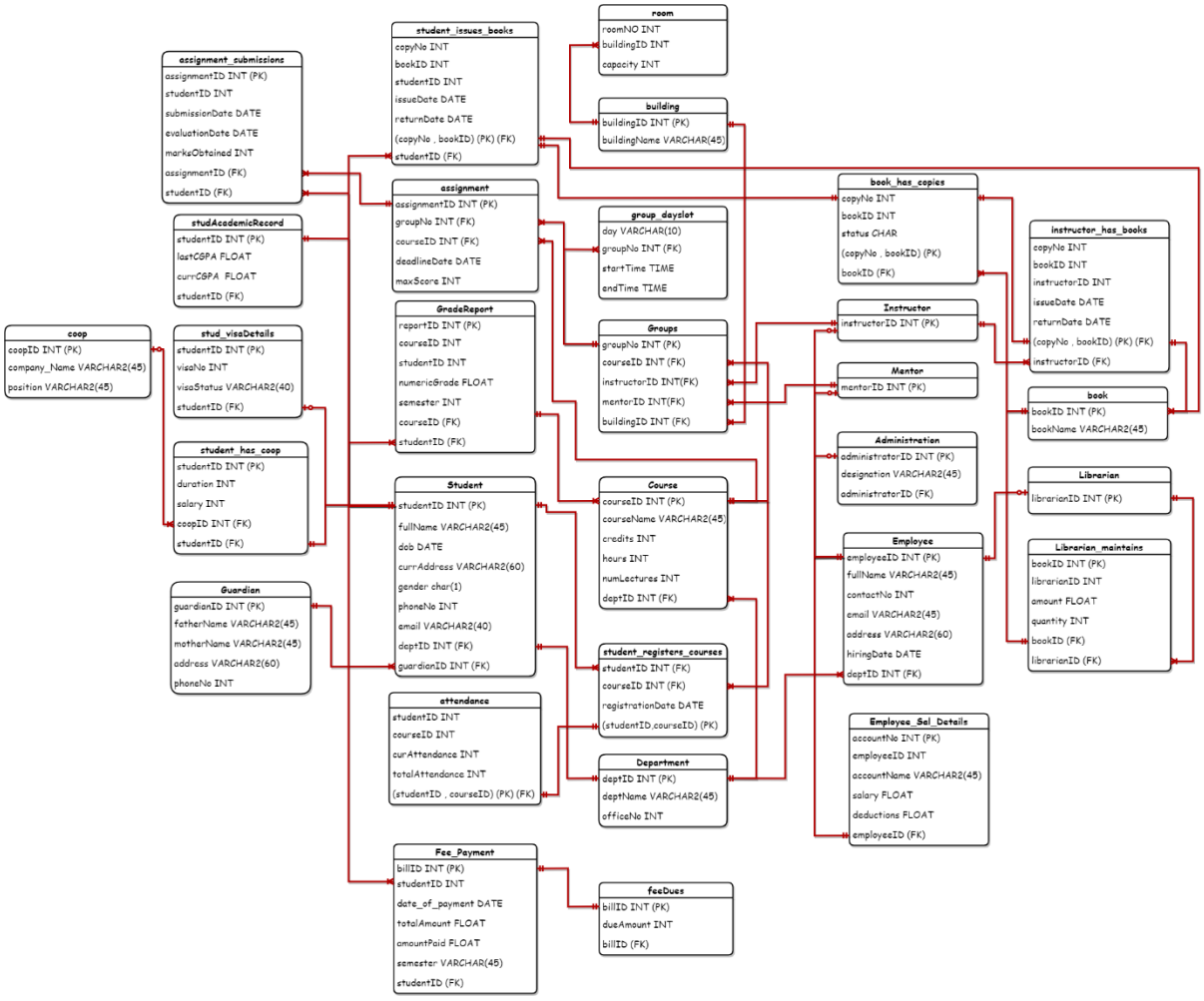
f) **Administrator:** An employee at a senior level and position, experienced and responsible for running the department for which he is accounted for. He can view the employee details of his department working under him .



g) **Mentor:** A person who provide guidance and monitors the student's term at the institute.



2.2 Logical Model



Above relationship schema of our system is also constructed using the web-based tool diagrams.net . It helped us to model the structure of the database , all tables , their attributes etc. By using this , we can also understand the outcomes of the database.

2.3 Physical Model

- TABLE DEPARTMENT :-

Name	Type	Size	Constraint
deptID	INT	DEFAULT	PRIMARY KEY
deptName	VARCHAR2	45	NOT NULL
officeNo	INT	DEFAULT	NOT NULL

```
CREATE TABLE Department(
   deptID INT PRIMARY KEY,
   deptName VARCHAR2(45) NOT NULL,
   officeNo INT NOT NULL
);
```

- TABLE EMPLOYEE :-

Name	Type	Size	Constraint
employeeID	INT	DEFAULT	PRIMARY KEY
fullName	VARCHAR2	45	NOT NULL
contactNo	INT	DEFAULT	UNIQUE,NOT NULL
email	VARCHAR2	45	UNIQUE,NOT NULL
address	VARCHAR2	60	NOT NULL
hiringDate	DATE	DEFAULT	
deptID	INT	DEFAULT	NOT NULL,FOREIGN KEY

```
CREATE TABLE EMPLOYEE(
   employeeID INT PRIMARY KEY,
   fullName VARCHAR2(45) NOT NULL,
   contactNo INT UNIQUE NOT NULL,
   email VARCHAR2(45) UNIQUE NOT NULL,
   address VARCHAR2(60) NOT NULL,
   hiringDate DATE,
   deptID INT NOT NULL,
   FOREIGN KEY (deptID) references DEPARTMENT(DEPTID) ON DELETE CASCADE
);
```

➢ TABLE EMPLOYEE_SAL_DETAILS :-

Name	Type	Size	Constraint
accountNo	INT	DEFAULT	PRIMARY KEY
employeeID	INT	DEFAULT	UNIQUE,NOT NULL,FOREIGN KEY
accountName	VARCHAR2	45	NOT NULL
salary	FLOAT	DEFAULT	NOT NULL
deductions	FLOAT	DEFAULT	

```
CREATE TABLE EMPLOYEE_SAL_DETAILS(
   accountNo INT PRIMARY KEY,
   employeeID INT UNIQUE NOT NULL,
   accountName VARCHAR2(45) NOT NULL,
   salary FLOAT NOT NULL,
   deductions FLOAT,
   FOREIGN KEY (employeeID) references EMPLOYEE(employeeID) ON DELETE CASCADE
);
```

➢ TABLE ADMINISTRATION :-

Name	Type	Size	Constraint
administratorID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY
designation	VARCHAR2	45	

```
CREATE TABLE ADMINISTRATION(
   administratorID INT PRIMARY KEY,
   designation VARCHAR2(45),
   FOREIGN KEY (administratorID) REFERENCES EMPLOYEE(employeeID) ON DELETE CASCADE
);
```

➢ TABLE MENTOR :-

Name	Type	Size	Constraint
mentorID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY

```
CREATE TABLE MENTOR(
   mentorID INT PRIMARY KEY,
   FOREIGN KEY (mentorID) REFERENCES EMPLOYEE(employeeID) ON DELETE CASCADE
);
```

➢ TABLE INSTRUCTOR :-

Name	Type	Size	Constraint
instructorID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY

```
CREATE TABLE INSTRUCTOR(
   instructorID INT PRIMARY KEY,
   FOREIGN KEY (instructorID) REFERENCES EMPLOYEE(employeeID) ON DELETE CASCADE
);
```

➢ TABLE GUARDIAN :-

Name	Type	Size	Constraint
guardianID	INT	DEFAULT	PRIMARY KEY
fatherName	VARCHAR2	45	NOT NULL
motherName	VARCHAR2	45	NOT NULL
address	VARCHAR2	60	NOT NULL
phoneNo	INT	DEFAULT	NOT NULL

```
CREATE TABLE GUARDIAN(
   guardianID INT PRIMARY KEY,
   fatherName VARCHAR2(45) NOT NULL,
   motherName VARCHAR2(45) NOT NULL,
   address VARCHAR2(60) NOT NULL,
   phoneNo INT NOT NULL
);
```

➢ TABLE STUDENT :-

Name	Type	Size	Constraint
studentID	INT	DEFAULT	PRIMARY KEY
fullName	VARCHAR2	45	NOT NULL
DOB	DATE	DEFAULT	NOT NULL
currAddress	VARCHAR2	DEFAULT	NOT NULL
gender	CHAR	1	NOT NULL
phoneNo	INT	DEFAULT	UNIQUE,NOT NULL
email	VARCHAR2	40	UNIQUE,NOT NULL
deptID	INT	DEFAULT	NOT NULL,FOREIGN KEY
guardianID	INT	DEFAULT	NOT NULL,FOREIGN KEY

```
CREATE TABLE Student(
   studentID INT PRIMARY KEY,
   fullName VARCHAR2(45) NOT NULL,
   DOB DATE NOT NULL,
```

```
  currAddress VARCHAR2(60) NOT NULL,
  gender CHAR(1) NOT NULL,
  phoneNo INT UNIQUE NOT NULL,
  email VARCHAR2(40) UNIQUE NOT NULL,
  deptID INT NOT NULL,
  guardianID INT NOT NULL,
  FOREIGN KEY (deptID) references DEPARTMENT(deptID) ON DELETE CASCADE,
  FOREIGN KEY (guardianID) references GUARDIAN(guardianID) ON DELETE CASCADE
);
```

➢ TABLE COURSE :-

Name	Type	Size	Constraint
courseID	INT	DEFAULT	PRIMARY KEY
courseName	VARCHAR2	45	NOT NULL
credits	INT	DEFAULT	NOT NULL
hours	INT	DEFAULT	NOT NULL
numLectures	INT	DEFAULT	NOT NULL
deptID	INT	DEFAULT	NOT NULL,FOREIGN KEY

```
CREATE TABLE Course(
  courseID INT PRIMARY KEY,
  courseName VARCHAR2(45) NOT NULL,
  credits INT NOT NULL,
  hours INT NOT NULL,
  numLectures INT NOT NULL,
  deptID INT NOT NULL,
  FOREIGN KEY (deptID) references DEPARTMENT(DEPTID) ON DELETE CASCADE
);
```

➢ TABLE STUD_VISADETAILS :-

Name	Type	Size	Constraint
studentID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY
visaNo	INT	DEFAULT	UNIQUE,NOT NULL
visaStatus	VARCHAR2	40	NOT NULL

```
CREATE TABLE stud_visaDetails(
  studentID INT PRIMARY KEY,
  visaNo INT UNIQUE NOT NULL,
  visaStatus VARCHAR2(40) NOT NULL,
  FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE CASCADE
);
```

➢ TABLE COOP :-

Name	Type	Size	Constraint
coopID	INT	DEFAULT	PRIMARY KEY
company_Name	VARCHAR2	45	NOT NULL
position	VARCHAR2	45	NOT NULL

```
CREATE TABLE coop(
   coopID INT PRIMARY KEY,
   company_Name VARCHAR2(45) NOT NULL,
   position VARCHAR2(45) NOT NULL
);
```

➢ TABLE STUDENT_HAS_COOP :-

Name	Type	Size	Constraint
studentID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY
duration	INT	DEFAULT	NOT NULL
salary	INT	DEFAULT	NOT NULL
coopID	INT	DEFAULT	NOT NULL,FOREIGN KEY

```
CREATE TABLE student_has_coop(
   studentID INT PRIMARY KEY,
   duration INT NOT NULL,
   salary INT NOT NULL,
   coopID INT NOT NULL,
   FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE CASCADE,
   FOREIGN KEY (coopID) REFERENCES coop(coopID) ON DELETE CASCADE
);
```

➢ TABLE studAcademicRecord :-

Name	Type	Size	Constraint
studentID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY
lastCGPA	FLOAT	DEFAULT	NOT NULL
currCGPA	FLOAT	DEFAULT	NOT NULL

```
CREATE TABLE studAcademicRecord(
   studentID INT PRIMARY KEY,
   lastCGPA FLOAT NOT NULL,
   currCGPA FLOAT NOT NULL,
   FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE CASCADE
);
```

➢ TABLE ATTENDANCE :-

Name	Type	Size	Constraint
studentID	INT	DEFAULT	NOT NULL
curAttendance	INT	DEFAULT	NOT NULL
totaleAttendance	INT	DEFAULT	NOT NULL
courseID	INT	DEFAULT	NOT NULL
studentID , courseID			PRIMARY KEY,FOREIGN KEY

```
CREATE TABLE attendance(
   studentID INT NOT NULL,
   curAttendance INT NOT NULL,
   totalAttendance INT NOT NULL,
   courseID INT NOT NULL,
   PRIMARY KEY(studentID,courseID),
   FOREIGN KEY (studentID,courseID) REFERENCES student_registers_courses(studentID,courseID)
   ON DELETE CASCADE
);
```

➢ TABLE GradeReport :-

Name	Type	Size	Constraint
reportID	INT	DEFAULT	PRIMARY KEY
courseID	INT	DEFAULT	NOT NULL,FOREIGN KEY
studentID	INT	DEFAULT	NOT NULL,FOREIGN KEY
numericGrade	FLOAT	DEFAULT	NOT NULL
semester	INT	DEFAULT	NOT NULL

```
CREATE TABLE GradeReport(
   reportID INT PRIMARY KEY,
   courseID INT NOT NULL,
   studentID INT NOT NULL,
   numericGrade FLOAT NOT NULL,
   semester INT NOT NULL,
   FOREIGN KEY (courseID) REFERENCES Course(courseID) ON DELETE CASCADE,
   FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE CASCADE
);
```

➢ TABLE FEE_PAYMENT :-

Name	Type	Size	Constraint
billID	INT	DEFAULT	PRIMARY KEY
studentID	INT	DEFAULT	NOT NULL,FOREIGN KEY
date_of_payment	DATE	DEFAULT	NOT NULL
totalAmnt	INT	DEFAULT	NOT NULL
Amount_Paid	INT	DEFAULT	NOT NULL
semester	INT	DEFAULT	NOT NULL

```
CREATE TABLE Fee_Payment(
   billID INT PRIMARY KEY,
   studentID INT NOT NULL,
   date_of_payment DATE NOT NULL,
   totalAmnt INT NOT NULL,
   Amount_Paid INT NOT NULL,
   semester INT NOT NULL,
   FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE CASCADE
);
```

- TABLE FeeDues :-

Name	Type	Size	Constraint
billID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY
dueAmnt	INT	DEFAULT	NOT NULL

```
CREATE TABLE feeDues(
   billID INT PRIMARY KEY,
   dueAmnt INT NOT NULL,
   FOREIGN KEY (billID) REFERENCES Fee_Payment(billID) ON DELETE CASCADE
);
```

- TABLE STUDENT_REGISTERS_COURSES :-

Name	Type	Size	Constraint
studentID	INT	DEFAULT	NOT NULL,FOREIGN KEY
courseID	INT	DEFAULT	NOT NULL,FOREIGN KEY
registrationDate	DATE	DEFAULT	
(studentID,courseID)			PRIMARY KEY

```
CREATE TABLE student_registers_courses(
   studentID INT NOT NULL,
   courseID INT NOT NULL,
   registrationDate DATE,
   PRIMARY KEY(studentID,courseID),
   FOREIGN KEY (courseID) REFERENCES Course(courseID) ON DELETE CASCADE,
   FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE CASCADE
);
```

- TABLE BUILDING :-

Name	Type	Size	Constraint
buildingID	INT	DEFAULT	PRIMARY KEY
buildingName	CHAR	1	NOT NULL

```
CREATE TABLE BUILDING(
   buildingID INT PRIMARY KEY,
   buildingName VARCHAR(45) NOT NULL
);
```

➢ TABLE ROOM :-

Name	Type	Size	Constraint
roomNO	INT	DEFAULT	PRIMARY KEY
buildingID	INT	DEFAULT	NOT NULL,FOREIGN KEY
capacity	INT	DEFAULT	NOT NULL

```
CREATE TABLE ROOM(
   roomNO INT PRIMARY KEY,
   buildingID INT NOT NULL,
   capacity INT NOT NULL,
   FOREIGN KEY (buildingID) references building(buildingID) ON DELETE CASCADE
);
```

➢ TABLE GROUPS :-

Name	Type	Size	Constraint
groupNo	INT	DEFAULT	PRIMARY KEY
courseID	INT	DEFAULT	NOT NULL,FOREIGN KEY
instructorID	INT	DEFAULT	NOT NULL,FOREIGN KEY
mentorID	INT	DEFAULT	NOT NULL,FOREIGN KEY
buildingID	INT	DEFAULT	NOT NULL,FOREIGN KEY

```
CREATE TABLE GROUPS(
   groupNo INT PRIMARY KEY,
   courseID INT NOT NULL,
   instructorID INT NOT NULL,
   mentorID INT NOT NULL,
   buildingID INT NOT NULL,
   FOREIGN KEY (courseID) REFERENCES Course(courseID) ON DELETE CASCADE,
   FOREIGN KEY (instructorID) REFERENCES INSTRUCTOR(instructorID) ON DELETE CASCADE,
   FOREIGN KEY (mentorID) REFERENCES MENTOR(mentorID) ON DELETE CASCADE,
   FOREIGN KEY (buildingID) REFERENCES BUILDING(buildingID) ON DELETE CASCADE
);
```

➢ TABLE GROUP_DAYSLOT :-

Name	Type	Size	Constraint
day	CHAR	1	NOT NULL
groupNo	INT	DEFAULT	NOT NULL,FOREIGN KEY
startTime	INT	DEFAULT	NOT NULL
endTime	INT	DEFAULT	NOT NULL

```
CREATE TABLE Group_dayslot(
   day VARCHAR(10) NOT NULL,
   groupNo INT NOT NULL,
   startTime INTERVAL DAY TO SECOND(2) NOT NULL,
   endTime INTERVAL DAY TO SECOND(2) NOT NULL,
   FOREIGN KEY (groupNo) REFERENCES GROUPS(groupNo) ON DELETE CASCADE
);
```

➢ TABLE ASSIGNMENT :-

Name	Type	Size	Constraint
assignmentID	INT	DEFAULT	PRIMARY KEY
groupNo	INT	DEFAULT	NOT NULL,FOREIGN KEY
courseID	INT	DEFAULT	NOT NULL,FOREIGN KEY
deadlineDate	DATE	DEFAULT	NOT NULL
maxScore	INT	DEFAULT	NOT NULL

```
CREATE TABLE assignment(
   assignmentID INT PRIMARY KEY,
   groupNo INT NOT NULL,
   courseID INT NOT NULL,
   deadlineDate DATE NOT NULL,
   maxScore INT NOT NULL,
   FOREIGN KEY (groupNo) REFERENCES GROUPS(groupNo) ON DELETE CASCADE,
   FOREIGN KEY (courseID) REFERENCES COURSE(COURSEID) ON DELETE CASCADE
);
```

➢ TABLE ASSIGNMENT_SUBMISSIONS :-

Name	Type	Size	Constraint
assignmentID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY
studentID	INT	DEFAULT	NOT NULL,FOREIGN KEY
submissionDate	DATE	DEFAULT	NOT NULL
evaluationDate	DATE	DEFAULT	NOT NULL
marksObtained	INT	DEFAULT	NOT NULL

```
CREATE TABLE assignment_submissions(
   assignmentID INT PRIMARY KEY,
   studentID INT NOT NULL,
   submissionDate DATE NOT NULL,
   evaluationDate DATE NOT NULL,
   marksObtained INT NOT NULL,
   FOREIGN KEY (assignmentID) REFERENCES assignment(assignmentID) ON DELETE CASCADE,
   FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE CASCADE
);
```

➢ TABLE LIBRARIAN :

Name	Type	Size	Constraint
librarianID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY

```
CREATE TABLE Librarian(
   librarianID INT PRIMARY KEY,
   FOREIGN KEY (librarianID) REFERENCES EMPLOYEE(employeeID) ON DELETE CASCADE
);
```

➢ TABLE BOOK :-

Name	Type	Size	Constraint
bookID	INT	DEFAULT	PRIMARY KEY
bookName	VARCHAR2	45	

```
CREATE TABLE BOOK(
   bookID INT PRIMARY KEY,
   bookName VARCHAR2(45)
);
```

➢ TABLE LIBRARIAN_MAINTAINS :-

Name	Type	Size	Constraint
bookID	INT	DEFAULT	PRIMARY KEY,FOREIGN KEY
librarianID	INT	DEFAULT	NOT NULL,FOREIGN KEY
amount	FLOAT	DEFAULT	NOT NULL
quantity	INT	DEFAULT	NOT NULL

```
CREATE TABLE Librarian_maintains(
   bookID INT PRIMARY KEY,
   librarianID INT NOT NULL,
   amount FLOAT NOT NULL,
   quantity INT NOT NULL,
   FOREIGN KEY (bookID) references BOOK(bookID) ON DELETE CASCADE,
   FOREIGN KEY (librarianID) references Librarian(librarianID) ON DELETE CASCADE
);
```

➢ TABLE BOOK_HAS_COPIES :-

Name	Type	Size	Constraint
copyNo	INT	DEFAULT	NOT NULL
bookID	INT	DEFAULT	NOT NULL , FOREIGN KEY
status	CHAR	1	NOT NULL
copyNo,bookID			PRIMARY KEY

```
CREATE TABLE Book_has_copies(
   copyNo INT NOT NULL,
   bookID INT NOT NULL,
   status CHAR NOT NULL,
   PRIMARY KEY(copyNo,bookID),
   FOREIGN KEY (bookID) references BOOK(bookID) ON DELETE CASCADE
);
```

➢ TABLE INSTRUCTOR_HAS_BOOKS :-

Name	Type	Size	Constraint
copyNo	INT	DEFAULT	NOT NULL
bookID	INT	DEFAULT	NOT NULL
instructorID	INT	DEFAULT	NOT NULL,FOREIGN KEY
issueDate	DATE	DEFAULT	NOT NULL
returnDate	DATE	DEFAULT	-
copyNo,bookID			PRIMARY KEY , FOREIGN KEY

```
CREATE TABLE instructor_has_books(
   copyNo INT PRIMARY KEY,
   instructorID INT NOT NULL,
   issueDate DATE NOT NULL,
   returnDate DATE,
   FOREIGN KEY (copyNo) references Book_has_copies(copyNo) ON DELETE CASCADE,
   FOREIGN KEY (instructorID) references INSTRUCTOR(instructorID) ON DELETE CASCADE
);
```

➢ TABLE STUDENT_ISSUES_BOOKS :-

Name	Type	Size	Constraint
copyNo	INT	DEFAULT	NOT NULL
bookID	INT	DEFAULT	NOT NULL
studentID	INT	DEFAULT	NOT NULL,FOREIGN KEY
issueDate	DATE	DEFAULT	NOT NULL
returnDate	DATE	DEFAULT	-
copyNo,bookID			PRIMARY KEY , FOREIGN KEY

```
CREATE TABLE student_issues_books(
   copyNo INT PRIMARY KEY,
   studentID INT NOT NULL,
   issueDate DATE NOT NULL,
   returnDate DATE,
   FOREIGN KEY (copyNo) references Book_has_copies(copyNo) ON DELETE CASCADE,
   FOREIGN KEY (studentID) references Student(studentID) ON DELETE CASCADE
);
```

➢ Sequences:-

These 7 sequences are used to make it easy for insertion of Primary Key values , IDs and to get rid of checking the previous value .

```
create sequence ForStudents start with 1;
create sequence ForGuardians start with 1;
create sequence ForDept start with 10 increment by 10;
create sequence ForEmp start with 100;
create sequence ForGroups start with 1;
create sequence ForBookCopies start with 1;
create sequence ForCourses start with 10 increment by 10;
```

➢ Triggers:-

- Trigger to update the status of the book when any student issues/returns the book.

```
CREATE OR REPLACE TRIGGER update_Stud_issue_Status
AFTER INSERT OR UPDATE OR DELETE ON STUDENT_ISSUES_BOOKS
FOR EACH ROW
DECLARE
   currStatus CHAR;
BEGIN
   IF inserting THEN
      select status into currStatus from BOOK_HAS_COPIES where (copyNo =
      :New.copyNo
      AND bookID = :New.bookID);
      IF currStatus='N' THEN raise_application_error( -20001, 'BOOK
      ALREADY ISSUED!!!');
      ELSE update book_has_copies set status = 'N' where (copyNo =
      :New.copyNo AND bookID = :New.bookID);
       END IF;
   ELSIF (updating AND (:NEW.returnDate IS NOT NULL)) OR deleting THEN update
   book_has_copies set status = 'Y' where (copyNo = :OLD.copyNo AND bookID =
   :OLD.bookID);
  END IF;
END;
```

- Trigger to update the status of the book when any instructor issues/returns the book.

```
CREATE OR REPLACE TRIGGER update_instructor_issue_Status
AFTER INSERT OR UPDATE OR DELETE ON INSTRUCTOR_HAS_BOOKS
FOR EACH ROW
DECLARE
  currStatus CHAR;
BEGIN
  IF inserting THEN
    select status into currStatus from BOOK_HAS_COPIES where (copyNo
    = :New.copyNo AND bookID = :New.bookID);
    IF currStatus='N' THEN raise_application_error( -20001, 'BOOK
    ALREADY ISSUED!!!');
    ELSE update book_has_copies set status = 'N' where (copyNo =
    :New.copyNo AND bookID = :New.bookID);
    END IF;
  ELSIF (updating AND (:NEW.returnDate IS NOT NULL)) OR deleting
  THEN update book_has_copies set status = 'Y' where (copyNo =
  :OLD.copyNo AND bookID = :OLD.bookID);
  END IF;
END;
```

- Trigger to insert the billID of fee_payment table into dues table if there is some due.

```
CREATE TRIGGER FEE_PAYMENT_TRIGGER
AFTER INSERT ON FEE_PAYMENT
FOR EACH ROW
DECLARE
due NUMBER:=:NEW.totalAmnt-:NEW.AMOUNT_PAID;
BEGIN
IF due!=0 THEN
insert into FEEDUES values(:NEW.billID,due);
END IF;
END;
```

- Trigger to insert default attendance values in attendance table whenever student registers a new course.

```
CREATE OR REPLACE TRIGGER on_stud_courses_registration
AFTER INSERT ON student_registers_courses
FOR EACH ROW
BEGIN
INSERT INTO ATTENDANCE VALUES(:NEW.STUDENTID,0,0,:NEW.COURSEID);
END;
```

2.4 Interactive Queries

Query 1: List all the male student's IDs from the Student Table.

```
select STUDENTID from student where GENDER='M';
```

STUDENTID
3
4
1
2

Query 2: List the studentIDs along with their registered course names in sorted order.

```
select studentID,courseName from STUDENT_REGISTERS_COURSES join
course ON STUDENT_REGISTERS_COURSES.COURSEID=course.COURSEID
ORDER BY studentID;
```

STUDENTID	COURSENAME
1	DISCRETE STRUCTURES
2	DSA
2	DISCRETE STRUCTURES
3	AWD

Query 3: List total count of students who are studying Discrete Structures currently.

```
select COUNT(*) from STUDENT_REGISTERS_COURSES t1,COURSE t2 where
t1.courseID=t2.courseID AND courseName='DISCRETE STRUCTURES';
```

COUNT(*)
2

Query 4: List all the bookNames that are available to issue.

```
select DISTINCT bookName from book t1,book_has_copies t2
where status='Y' AND t1.bookID=t2.bookID;
```

BOOKNAME
DISCOVERY OF INDIA
PASSAGE OF INDIA

Query 5: Display salary details of the instructor with instructorID 103.

- `select * from EMPLOYEE_SAL_DETAILS where employeeID=103;`

ACCOUNTNO	EMPLOYEEID	ACCOUNTNAME	SALARY	DEDUCTIONS
211635548739	103	KAPIL SAINI	30000	0

Query 6: Display information of all the employees who are mentor for any group.

- `select * from employee where employeeID IN (select mentorID from mentor);`

EMPLOYEEID	FULLNAME	CONTACTNO	EMAIL	ADDRESS	HIRINGDATE	DEPTID
100	RAHUL SHARMA	7648436616	rahul@gmail.com	Sector 27 ,Chandigarh	11/22/2017	10
101	AKSHAY MALIK	8726520295	akshay@gmail.com	Sector 21 ,Chandigarh	08/01/2020	20
102	VINIT GOYAL	2809359551	vinit@gmail.com	Sector 11 ,Chandigarh	09/05/2021	30

Query 7: Display studentID and Name of the student who is having some fee due.

- `select t3.STUDENTID,t3.fullNAME from FEEDUES t1 join FEE_PAYMENT t2 on t1.billID=t2.billID join STUDENT t3 on 2.STUDENTID=t3.STUDENTID;`

STUDENTID	FULLNAME
1	SAHIL MEHTA

INSERTION QUERIES

DEPARTMENT :

```
INSERT INTO DEPARTMENT VALUES(FORDEPT.NEXTVAL , 'ECE' , 11);
INSERT INTO DEPARTMENT VALUES(FORDEPT.NEXTVAL , 'IT' , 9);
INSERT INTO DEPARTMENT VALUES(FORDEPT.NEXTVAL , 'ME' , 7);
INSERT INTO DEPARTMENT VALUES(FORDEPT.NEXTVAL , 'EEE' , 5);
INSERT INTO DEPARTMENT VALUES(FORDEPT.NEXTVAL , 'CSE' , 3);
INSERT INTO DEPARTMENT VALUES(FORDEPT.NEXTVAL , 'MANAGEMENT' , 17);
```

BUILDING :

```
INSERT INTO BUILDING VALUES(201 , 'NEWTON');
INSERT INTO BUILDING VALUES(202 , 'TESLA');
INSERT INTO BUILDING VALUES(203 , 'EINSTEIN');
INSERT INTO BUILDING VALUES(204 , 'GALILEO');
```

```
INSERT INTO BUILDING VALUES(205 , 'STEPHEN');
```

ROOM :

```
INSERT INTO ROOM VALUES(34 , 201 , 50);

INSERT INTO ROOM VALUES(55 , 202 , 45);
```

EMPLOYEE:

```
INSERT INTO EMPLOYEE VALUES(ForEmp.nextval,'RAHUL
SHARMA',7648436616,'rahul@gmail.com','Sector 27
,Chandigarh',TO_DATE('22/11/2017','DD/MM/YYYY'),10);

INSERT INTO EMPLOYEE VALUES(ForEmp.nextval,'AKSHAY
MALIK',8726520295,'akshay@gmail.com','Sector 21
,Chandigarh',TO_DATE('01/08/2020','DD/MM/YYYY'),20);

INSERT INTO EMPLOYEE VALUES(ForEmp.nextval,'VINIT
GOYAL',2809359551,'vinit@gmail.com','Sector 11
,Chandigarh',TO_DATE('05/09/2021','DD/MM/YYYY'),30);

INSERT INTO EMPLOYEE VALUES(ForEmp.nextval,'KAPIL
SAINI',2428287834,'kapil@gmail.com','Sector 2
,Chandigarh',TO_DATE('08/10/2020','DD/MM/YYYY'),40);

INSERT INTO EMPLOYEE VALUES(ForEmp.nextval,'RAJBIR
SINGH',9201000991,'rajbir@gmail.com','Sector 17
,Chandigarh',TO_DATE('17/05/2019','DD/MM/YYYY'),50);

INSERT INTO EMPLOYEE VALUES(ForEmp.nextval,'URVASHI
CHOUBEY',7044553351,'urvashi@gmail.com','Sector 01
,Chandigarh',TO_DATE('01/09/2020','DD/MM/YYYY'),50);

INSERT INTO EMPLOYEE
VALUES(ForEmp.nextval,'RAKESH',7452201446,'rakesh@gmail.com','Sector
01 ,Chandigarh',TO_DATE('01/09/2020','DD/MM/YYYY'),50);

INSERT INTO EMPLOYEE VALUES(ForEmp.nextval,'RANVEER
SINGH',3532291709,'ranveer@gmail.com','Sector 01
,Chandigarh',TO_DATE('01/09/2020','DD/MM/YYYY'),50);

INSERT INTO EMPLOYEE VALUES(ForEmp.nextval,'AMAN
SHARMA',6442413873,'aman@gmail.com','Sector 01
,Chandigarh',TO_DATE('01/09/2020','DD/MM/YYYY'),50);
```

EMPLOYEE SAL DETAILS:

```
INSERT INTO EMPLOYEE_SAL_DETAILS VALUES(332329020345,100,'RAHUL
SHARMA',15000,0);

INSERT INTO EMPLOYEE_SAL_DETAILS VALUES(366855344342,101,'AKSHAY
MALIK',20000,0);
```

```
INSERT INTO EMPLOYEE_SAL_DETAILS VALUES(939631189422,102,'VINIT
GOYAL',25000,0);

INSERT INTO EMPLOYEE_SAL_DETAILS VALUES(211635548739,103,'KAPIL
SAINI',30000,0);

INSERT INTO EMPLOYEE_SAL_DETAILS VALUES(177323139284,104,'RAJBIR
SINGH',12000,0);
```

INSTRUCTOR:

```
INSERT INTO INSTRUCTOR VALUES(100);

INSERT INTO INSTRUCTOR VALUES(101);

INSERT INTO INSTRUCTOR VALUES(102);

INSERT INTO INSTRUCTOR VALUES(103);

INSERT INTO INSTRUCTOR VALUES(104);
```

MENTOR:

```
INSERT INTO MENTOR VALUES(100);

INSERT INTO MENTOR VALUES(101);

INSERT INTO MENTOR VALUES(102);
```

ADMINISTRATION:

```
INSERT INTO ADMINISTRATION VALUES(105,'REGISTRAR');
```

LIBRARIAN:

```
INSERT INTO LIBRARIAN VALUES(106);

INSERT INTO LIBRARIAN VALUES(107);

INSERT INTO LIBRARIAN VALUES(108);
```

BOOK :

```
INSERT INTO BOOK VALUES(011 , 'WINGS OF FIRE');

INSERT INTO BOOK VALUES(012 , 'DISCOVERY OF INDIA');

INSERT INTO BOOK VALUES(013 , 'JUNGLE BOOK');

INSERT INTO BOOK VALUES(014 , 'PASSAGE OF INDIA');

INSERT INTO BOOK VALUES(015 , 'INVISIBLE MAN');
```

BOOK_HAS_COPIES :

```
INSERT INTO BOOK_HAS_COPIES VALUES(1,11,'N');
INSERT INTO BOOK_HAS_COPIES VALUES(2,11,'N');
INSERT INTO BOOK_HAS_COPIES VALUES(1,12,'N');
INSERT INTO BOOK_HAS_COPIES VALUES(2,12,'N');
INSERT INTO BOOK_HAS_COPIES VALUES(1,13,'N');
INSERT INTO BOOK_HAS_COPIES VALUES(2,13,'N');
INSERT INTO BOOK_HAS_COPIES VALUES(1,14,'N');
INSERT INTO BOOK_HAS_COPIES VALUES(2,14,'N');
INSERT INTO BOOK_HAS_COPIES VALUES(3,14,'N');
```

INSTRUCTOR_HAS_BOOKS :

```
INSERT INTO INSTRUCTOR_HAS_BOOKS VALUES(1,13,100,SYSDATE,NULL);
INSERT INTO INSTRUCTOR_HAS_BOOKS VALUES(3,14,101,SYSDATE,NULL);
INSERT INTO INSTRUCTOR_HAS_BOOKS VALUES(2,12,102,SYSDATE,NULL);
```

STUDENT_ISSUES_BOOKS :

```
INSERT INTO STUDENT_ISSUES_BOOKS VALUES(1,11,3,SYSDATE,NULL);
INSERT INTO STUDENT_ISSUES_BOOKS VALUES(2,11,5,SYSDATE,NULL);
INSERT INTO STUDENT_ISSUES_BOOKS VALUES(2,13,1,SYSDATE,NULL);
```

LIBRARIAN_MAINTAINS :

```
INSERT INTO LIBRARIAN_MAINTAINS VALUES(11,106,150,50);
INSERT INTO LIBRARIAN_MAINTAINS VALUES(12,106,250,50);
INSERT INTO LIBRARIAN_MAINTAINS VALUES(13,107,80,50);
INSERT INTO LIBRARIAN_MAINTAINS VALUES(14,108,150,50);
INSERT INTO LIBRARIAN_MAINTAINS VALUES(15,107,70,50);
```

GUARDIAN :

```
INSERT INTO GUARDIAN
VALUES(ForGuardians.nextval,'Abhijit','Nandita','Mehrauli Gurgaon
Road, Gurgaon',6018929463);
```

```
INSERT INTO GUARDIAN
VALUES(ForGuardians.nextval,'Akshat','Neetu','16, Shyam Bazar,
Chandni Chowk,Faridabad',4862110941);

INSERT INTO GUARDIAN
VALUES(ForGuardians.nextval,'Sanjeet','Meenakshi','33, Shree
Society, R V Desai Road, Bhiwani',4970577610);

INSERT INTO GUARDIAN
VALUES(ForGuardians.nextval,'Sumit','Mallika','37, Gyan Society,Near
Rose Garden,Hisar',6982051066);

INSERT INTO GUARDIAN
VALUES(ForGuardians.nextval,'Aniket','Deepa','32, Shree Society, R V
Desai Road, Bhiwani',566586461);
```

STUDENT :

```
INSERT INTO STUDENT VALUES(ForStudents.NEXTVAL , 'SAHIL MEHTA' ,
TO_DATE('22/01/2002','DD/MM/YYYY') , 'Mehrauli Gurgaon Road,
Gurgaon' , 'M' , 456732 , 'sahil@gmail.com' , 10 , 1);

INSERT INTO STUDENT VALUES(ForStudents.NEXTVAL , 'SUMIT' ,
TO_DATE('29/08/2002','DD/MM/YYYY') , '16, Shyam Bazar, Chandni
Chowk,Faridabad' , 'M' , 657589 , 'sumit@gmail.com' , 20 , 2);

INSERT INTO STUDENT VALUES(ForStudents.NEXTVAL , 'UJJWAL' ,
TO_DATE('08/02/2004','DD/MM/YYYY') , '33, Shree Society, R V Desai
Road, Bhiwani' , 'M' , 657582 , 'ujjwal@gmail.com' , 30 , 3);

INSERT INTO STUDENT VALUES(ForStudents.NEXTVAL , 'VIPUL KUMAR' ,
TO_DATE('02/09/2002','DD/MM/YYYY') , '37, Gyan Society,Near Rose
Garden,Hisar' , 'M' , 12334 , 'vipul@gmail.com' , 40 , 4);

INSERT INTO STUDENT VALUES(ForStudents.NEXTVAL , 'URVASHI' ,
TO_DATE('01/09/2002','DD/MM/YYYY') , '32, Shree Society, R V Desai
Road, Bhiwani' , 'F' , 974655 , 'urvashi@gmail.com' , 50 , 5);
```

COOP :

```
INSERT INTO COOP VALUES(1111 , 'GeeksForGeeks' , 'Campus
Ambassador');

INSERT INTO COOP VALUES(1901 , 'TwoWaits' , 'DSA SCHOLAR');
```

STUDENT_HAS_COOP :

```
INSERT INTO STUDENT_HAS_COOP VALUES(3, 6 , 0,1111);

INSERT INTO STUDENT_HAS_COOP VALUES(5, 6 , 0,1901);
```

STUD_VISADETAILS :

```
INSERT INTO STUD_VISADETAILS VALUES( 2 , 999112313771 , 'ACTIVE');

INSERT INTO STUD_VISADETAILS VALUES( 4 , 111787363918 , 'ACTIVE');
```

FEE_PAYMENT :

```
INSERT INTO FEE_PAYMENT VALUES(12101871 , 1 ,SYSDATE , 30000, 15000,
1);
```

COURSE :

```
INSERT INTO COURSE VALUES(ForCourses.nextval,'DBMS',05,130,65,50);

INSERT INTO COURSE VALUES(ForCourses.nextval,'DSA',07,150,75,50);

INSERT INTO COURSE VALUES(ForCourses.nextval,'DISCRETE
STRUCTURES',04,120,60,10);

INSERT INTO COURSE VALUES(ForCourses.nextval,'DELD',05,130,65,40);

INSERT INTO COURSE VALUES(ForCourses.nextval,'AWD',05,130,65,50);
```

STUDENT_REGISTERS_COURSES :

```
INSERT INTO STUDENT_REGISTERS_COURSES VALUES(1,30,SYSDATE);

INSERT INTO STUDENT_REGISTERS_COURSES VALUES(2,20,SYSDATE);

INSERT INTO STUDENT_REGISTERS_COURSES VALUES(2,30,SYSDATE);

INSERT INTO STUDENT_REGISTERS_COURSES VALUES(3,50,SYSDATE);
```

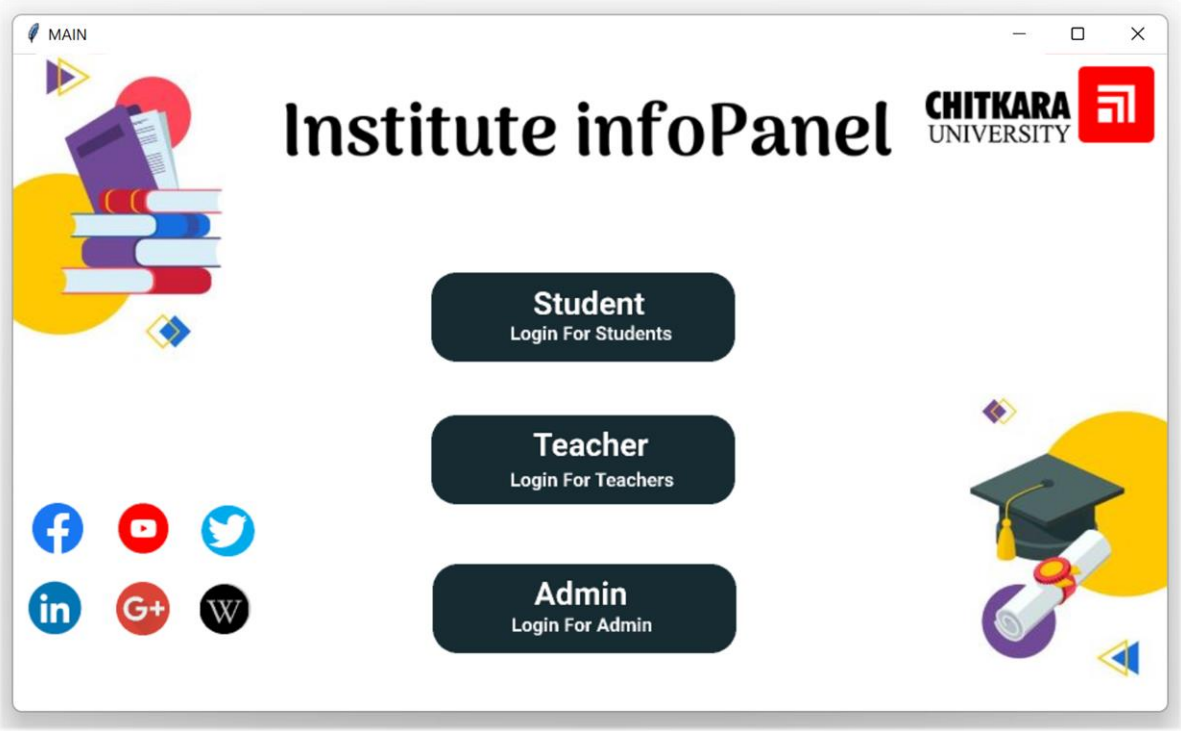
GROUPS :

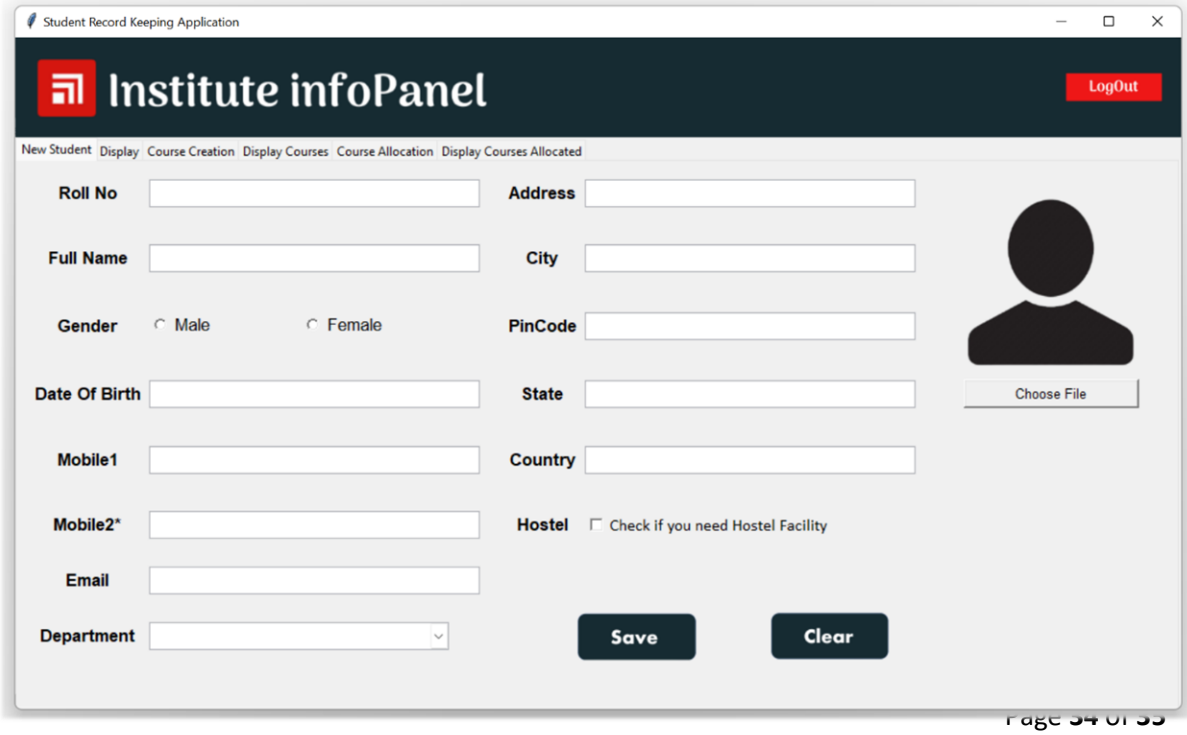
```
INSERT INTO GROUPS VALUES(ForGroups.NEXTVAL , 10 , 100 , 101 , 201);

INSERT INTO GROUPS VALUES(ForGroups.NEXTVAL , 20 , 101 , 100 , 202);

INSERT INTO GROUPS VALUES(ForGroups.NEXTVAL , 30 , 102 , 102 , 203);

INSERT INTO GROUPS VALUES(ForGroups.NEXTVAL , 40 , 103 , 100 , 205);

INSERT INTO GROUPS VALUES(ForGroups.NEXTVAL , 50 , 104 , 101 , 204);
```

Chapter 3: Conclusion & Future Work

Finally , conclusion of this project is to store and maintain an institute's all type of records efficiently. Without using this system, managing and maintaining the details is a tedious job for any organization.

In future, we can apply this normalized management system for effective and efficient institute database design to encourage different institute's faculties to store and manage data in a database without using papers, large files etc. efficiently and in a very flexible way.





We can build GUI application/software or some web-based portal by using which all members , students , staff can use this system in efficient manner. GUI development is also going on and will be completed soon !

BIBLIOGRAPHY

- ✓ GeeksForGeeks
- ✓ JavaTPoint
- ✓ W3Schools
- ✓ Oracle Docs
- ✓ Youtube
- ✓ Lecture Slides
- ✓ CHO Preffered Books