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

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

I would also like to thank my family and my team members for their continuous support and guidance .

Table Of Contents

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.

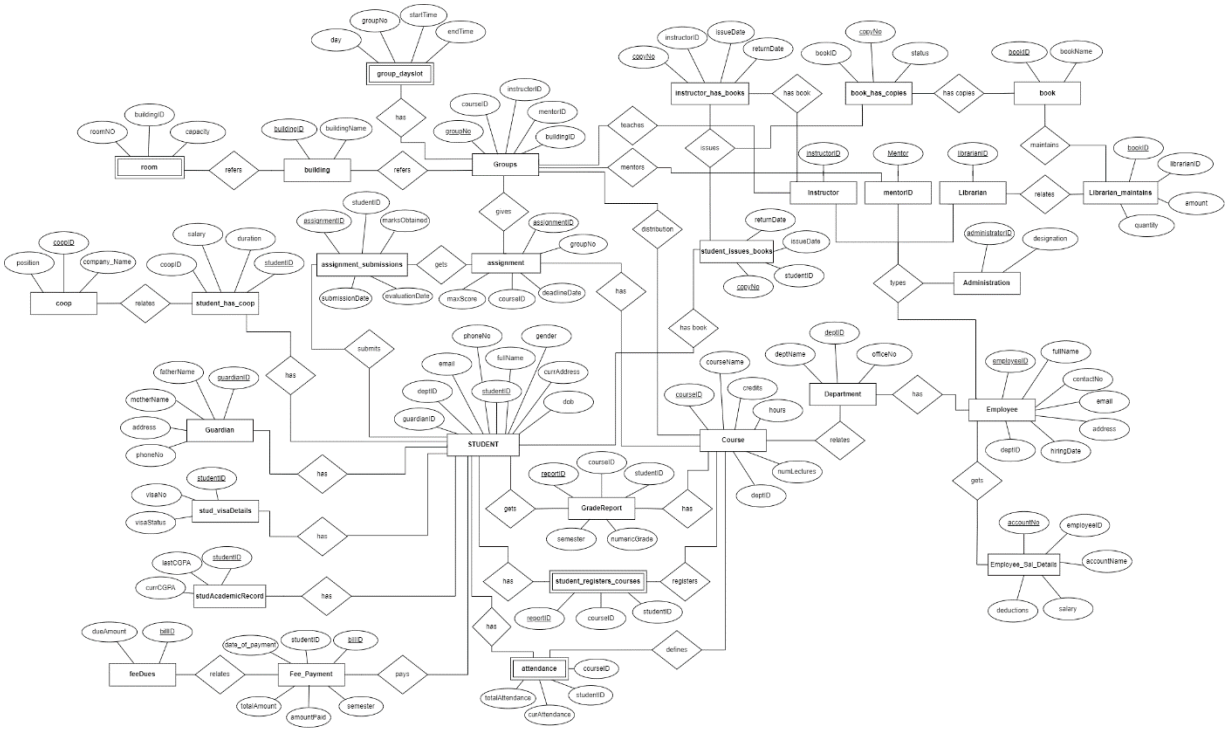
  DBMS manage the data, the database engine, and the database schema, allowing for data to be manipulated or extracted by users and other programs. This helps provide data security, data integrity, concurrency, and uniform data administration procedures.

  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.

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.

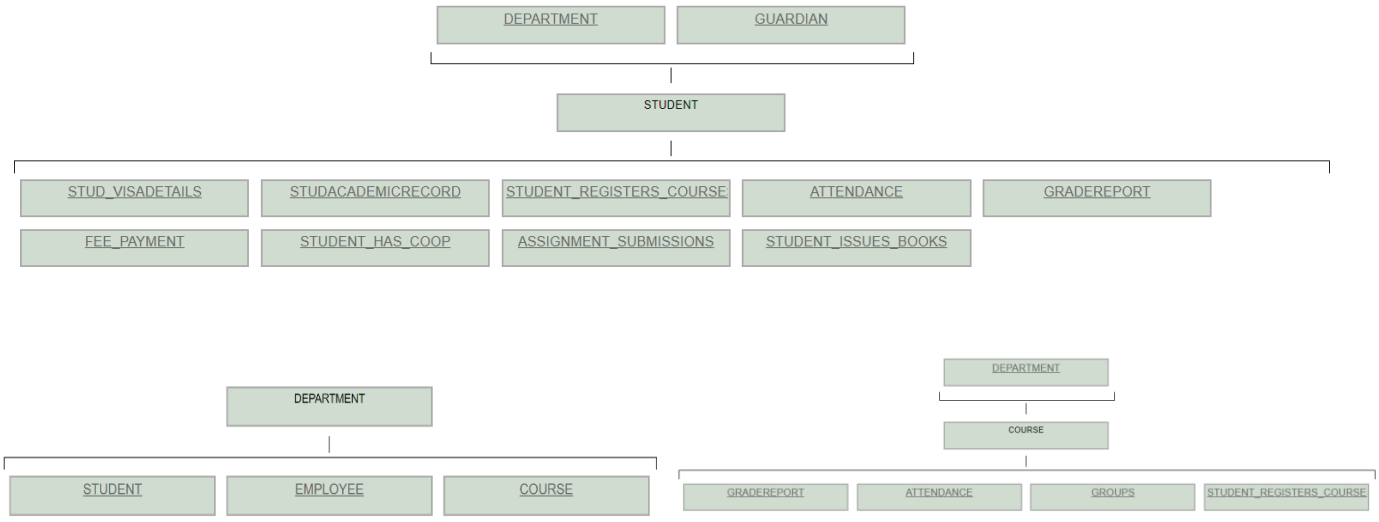
  The project deals with retrieval of information using database. It collects related information from all the departments of an organization and maintains database, which are used to generate reports in various forms to measure individual and overall performance of the students.

- 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

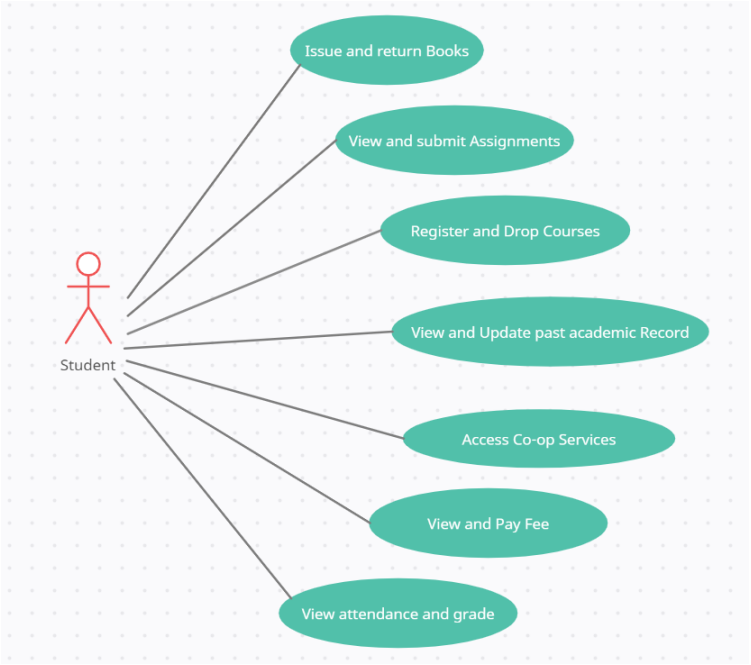
the performance of the students. These reports should be viewed by the faculty and user.



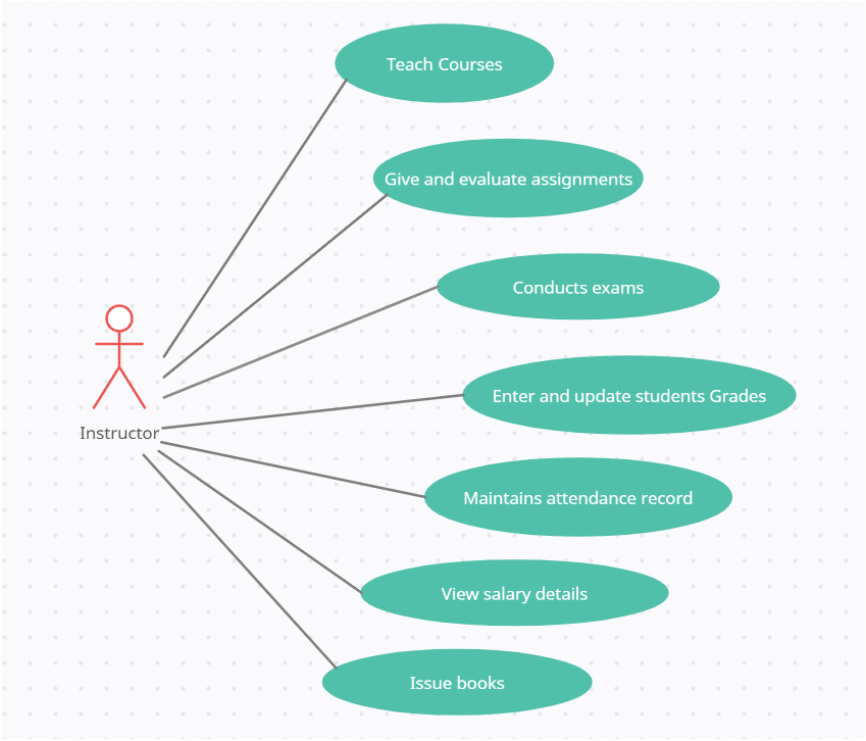
➢ ROLES AND RESPONSIBILITIES:

There are various roles played by different people:-

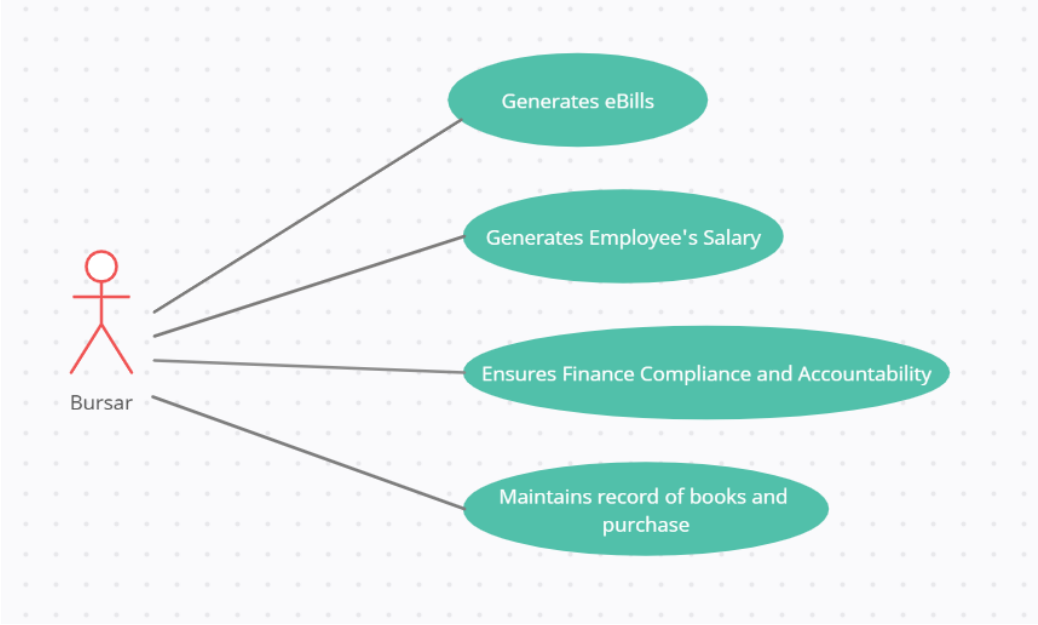
a) **Student:** The person who uses the application to interact with institute and associated employees, access his/her career opportunities & also get professional counselling from experienced mentors. It is instructed by instructors by taking section under various courses offered by the institution.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



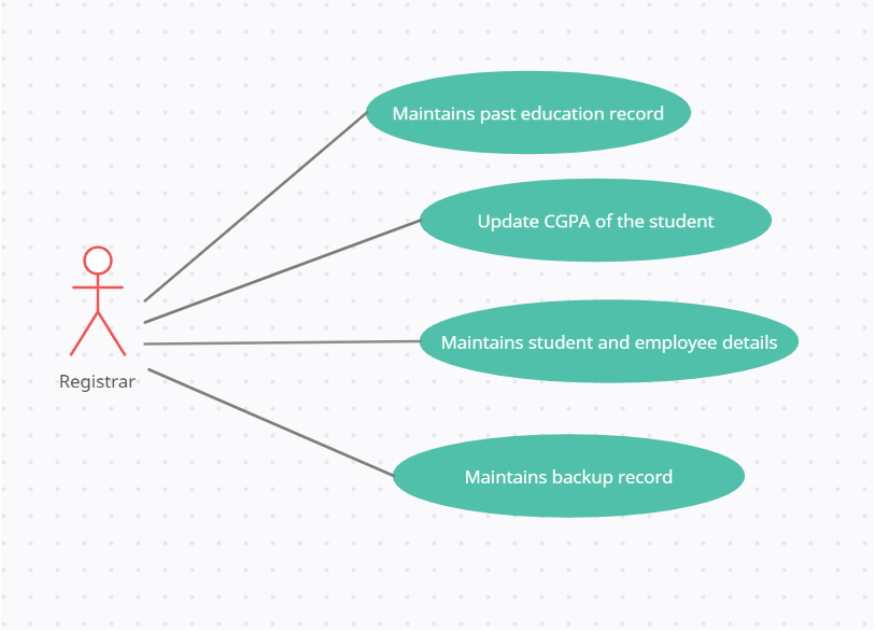
b) **Instructor:** The person who teaches students and is employed under a department, works under administration supervision and works for student welfare.



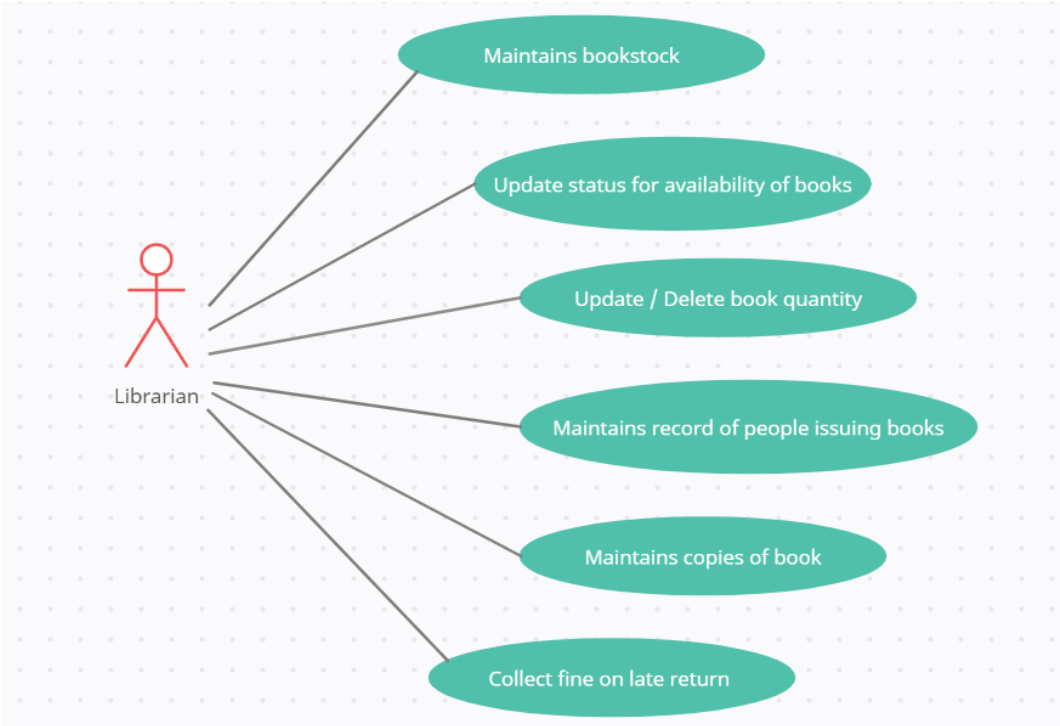
c) **Bursar:** The person who supervises the fiscal matter of the institution employees and student, responsible for generation of bills and remitting salary on monthly basis. It also keeps an eye on the dues of students.



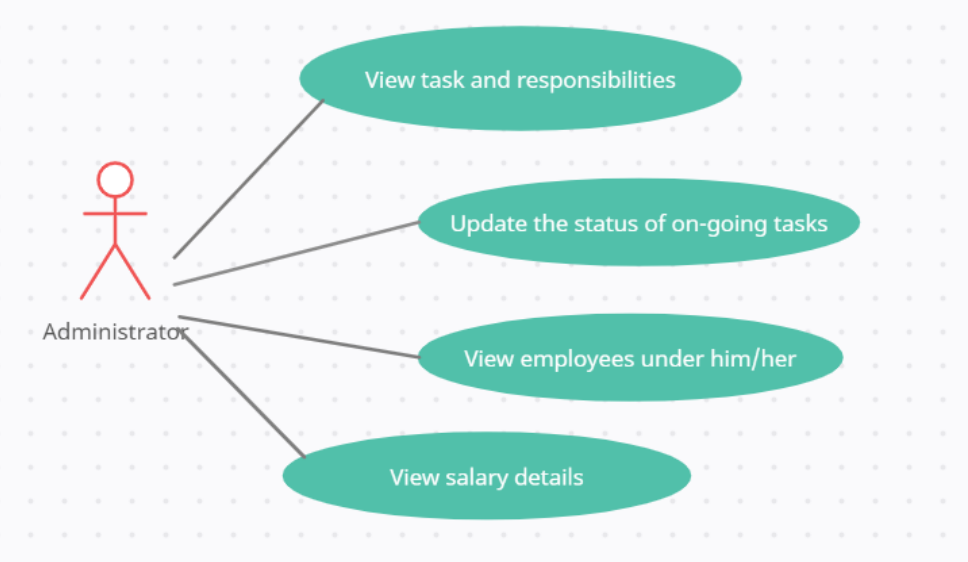
d) **Registrar:** The person who maintains student details. It also keeps a record of employee and administration staff. It also maintains a backup of the data in case it is lost. It is also responsible for updating the grades and marks of students in accordance with his registered courses.



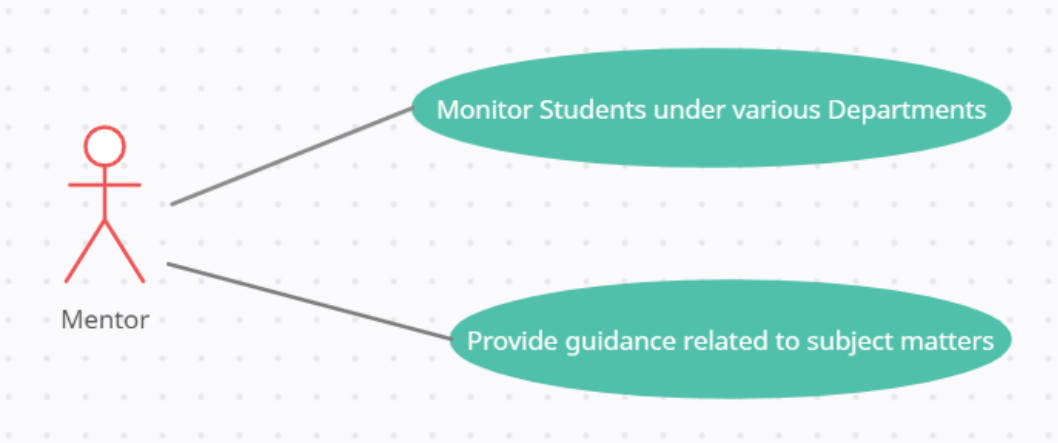
e) **Librarian:** The person who maintains the library, purchases books to serve as reference material for the students as well as instructors, maintains the quantity of books as per the need and demand. It maintains a record of people who issues books, and update the status of availability for the convenience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



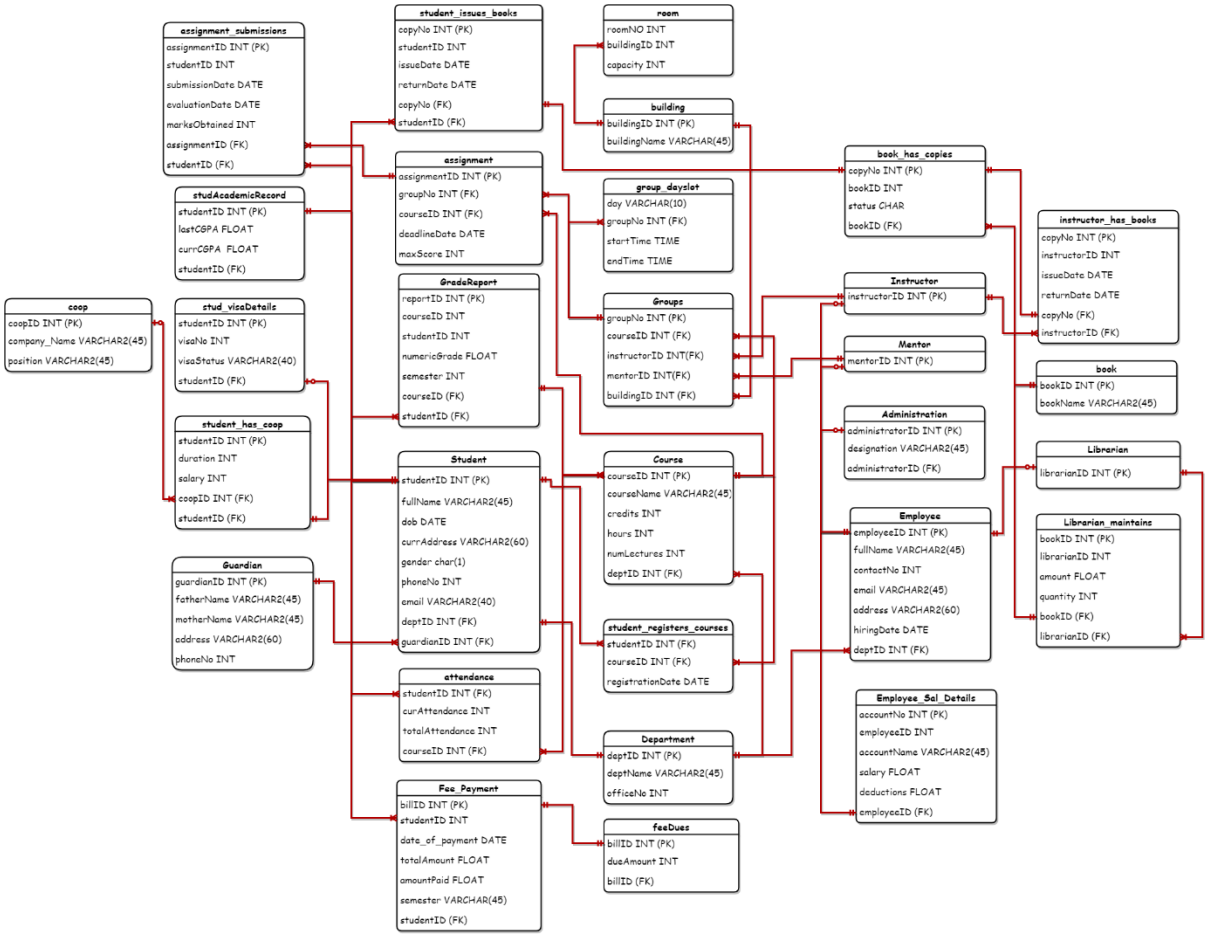
f) **Administrator:** An employee at a senior level and position, experienced and responsible for running the department for which he is accounted for. He can view the employee details of his department working under him .



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 :-

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

- TABLE EMPLOYEE :-

```
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 :-

```
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 :-

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

- TABLE MENTOR :-

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

- TABLE INSTRUCTOR :-

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

- TABLE GUARDIAN :-

```
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 :-

```
CREATE TABLE Student(
   studentID INT PRIMARY KEY,
   fullName varchar2(45) NOT NULL,
   DOB DATE NOT NULL,
   currAddress INT NOT NULL,
```

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

- TABLE Course :-

```
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 :-

```
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 :-

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

- TABLE student_has_coop :-

```
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 :-

```
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 :-

```
CREATE TABLE attendance(
   studentID INT NOT NULL,
   curAttendance INT NOT NULL,
   totaleAttendance INT NOT NULL,
   courseID INT NOT NULL,
   FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE
   CASCADE,
   FOREIGN KEY (courseID) REFERENCES Course(courseID) ON DELETE CASCADE
);
```

- TABLE GradeReport :-

```
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 :-

```
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 :-

```
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 :-

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

```
    FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE
    CASCADE
);
```

- TABLE BUILDING :-

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

- TABLE ROOM :-

```
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 :-

```
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 :-

```
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 :-

```
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 :-

```
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 :-

```
CREATE TABLE Librarian(
    librarianID INT PRIMARY KEY,
```

```
    FOREIGN KEY (librarianID) REFERENCES EMPLOYEE(employeeID) ON DELETE
    CASCADE
);
```

- TABLE BOOK :-

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

- TABLE Librarian_maintains :-

```
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 :-

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

- TABLE instructor_has_books :-

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

```
    FOREIGN KEY (instructorID) references INSTRUCTOR(instructorID) ON DELETE
    CASCADE
);
```

- TABLE student_issues_books :-

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

- Sequences:-

```
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 TRIGGER update_Stud_issue_Status
AFTER INSERT OR UPDATE OR DELETE ON STUDENT_ISSUES_BOOKS
FOR EACH ROW
BEGIN
IF inserting THEN update book_has_copies set status = 'N' where copyNo =
:New.copyNo;
ELSIF (updating AND (:NEW.returnDate IS NOT NULL)) OR deleting THEN
update book_has_copies set status = 'Y' where copyNo = :New.copyNo;
END IF;
END;
```

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

```
CREATE TRIGGER update_instructor_issue_Status
AFTER INSERT OR UPDATE OR DELETE ON INSTRUCTOR_HAS_BOOKS
FOR EACH ROW
BEGIN
IF inserting THEN update book_has_copies set status = 'N' where copyNo =
:New.copyNo;
ELSIF (updating AND (:NEW.returnDate IS NOT NULL)) OR deleting THEN
update book_has_copies set status = 'Y' where copyNo = :New.copyNo;
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;
```

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
1
3
5
6
7

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	DBMS
1	MATHS
2	MATHS
2	CPP
3	CPP
3	DBMS

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

```
select COUNT(*) from temp1,temp2 where
temp1.courseID=temp2.courseID AND courseName='DBMS';
```

COUNT(*)
2

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

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

BOOKNAME
CSS
HTML

Query 5: Display salary details of the instructor with instructorID 12.

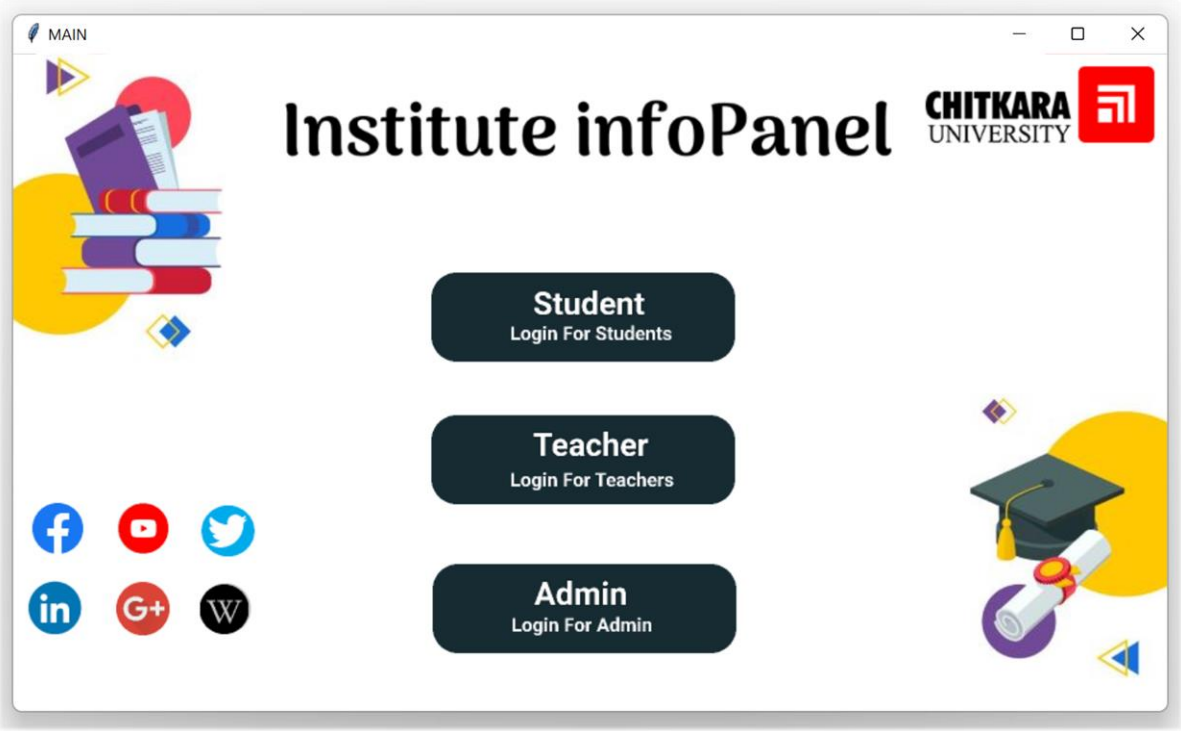
```
select * from EMPLOYEE_SAL_DETAILS where employeeID=12;
```

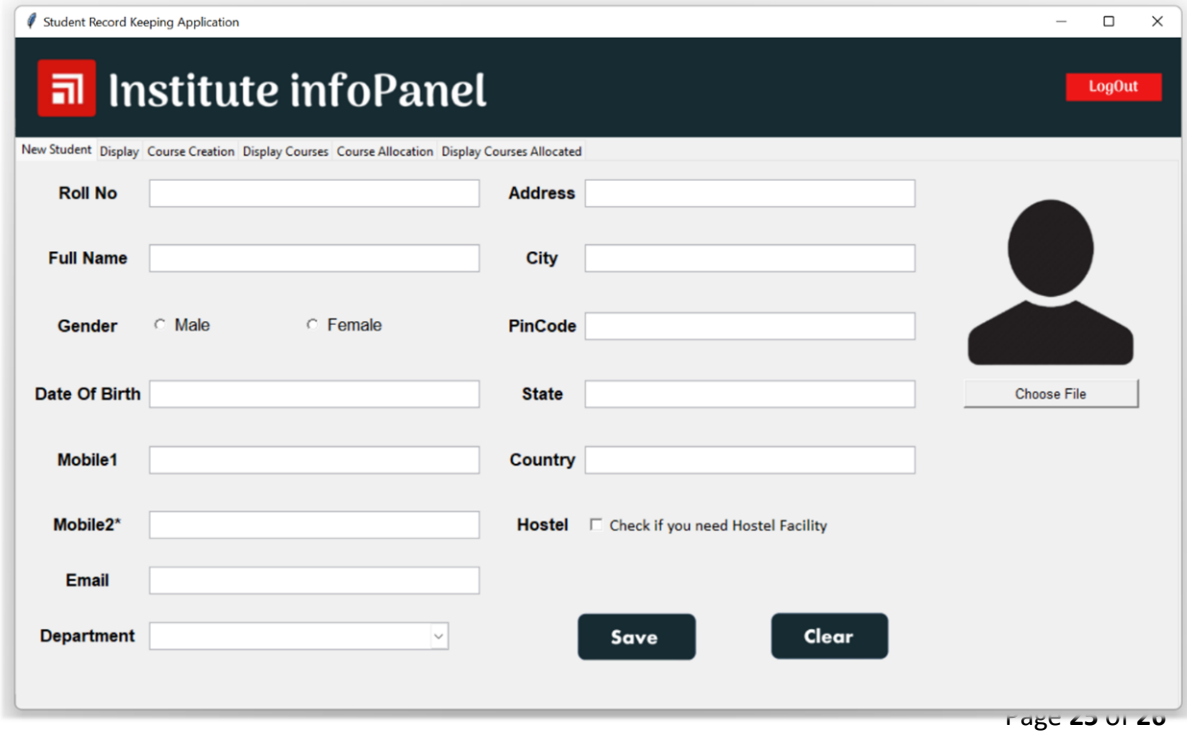
ACCOUNTNO	EMPLOYEEID	ACCOUNTNAME	SALARY	DEDUCTIONS
12091719211	12	John Miller	250000	0

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