University Management System

DATABASE DESIGN

Shelley Bhatnagar
[BHATNAGAR.S@HUSKY.NEU.EDU]| [NUID: 001907934]

ABSTRACT:

UNIVERSITY MANAGEMENT SYSTEM (UMS) deals with the maintenance of university, college, faculty, student information within the university. UMS has a relational database, which is used to store the college, faculty, student, courses and information of a college.

Starting from registration of a new student in the college, it maintains all the details regarding the attendance and marks of the students. The project deals with retrieval of information through an intranet based campus wide portal. It collects related information from all the departments of an organization and maintains files, which are used to generate reports in various forms to measure individual and overall performance of the students.

The university management system is store and retrieve the information through web based application. So it collects the information of individual and overall performance of students in various departments. UMS focuses on the basic need of accomplishing the task of maintaining the large stock of information in a university by creating a database. The interface is a very efficient application for the management of a university which not only benefits the user of the university but also plays a major role in enabling the management of the university to work in a proficient manner. This system will be a platform where users will have access to the facilities of the university including blackboard from anywhere using the Internet. This project report will provide a detailed account of the functionalities of the user interface which is taken as a reference to manage a university. 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.

PURPOSE OF THE SYSTEM:

UNIVERSITY MANAGEMENT SYSTEM [UMS] deals with the maintenance of university, college, faculty, student information within the university. This project of UMS involved the automation of student information that can be implemented in different college managements.

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

PROBLEMS IN THE EXISTING SYSTEM:

Storing and accessing the data in the form of Excel sheets and account books is a tedious work. It requires a lot of laborious work. It may often yield undesired results. Maintaining these records as piles may turn out to be a costlier task than any other of the colleges and institutions.

RISK INVOLVED IN THE EXISTING SYSTEM:

Present System is time-consuming and also results in lack of getting inefficient results. Some of the risks involved in the present system can be as follows: During the entrance of marks and attendance, if any mistake is done at a point, then this becomes cumulative and leads to adverse consequences. If there is any need to retrieve results it may seem to be difficult to search.

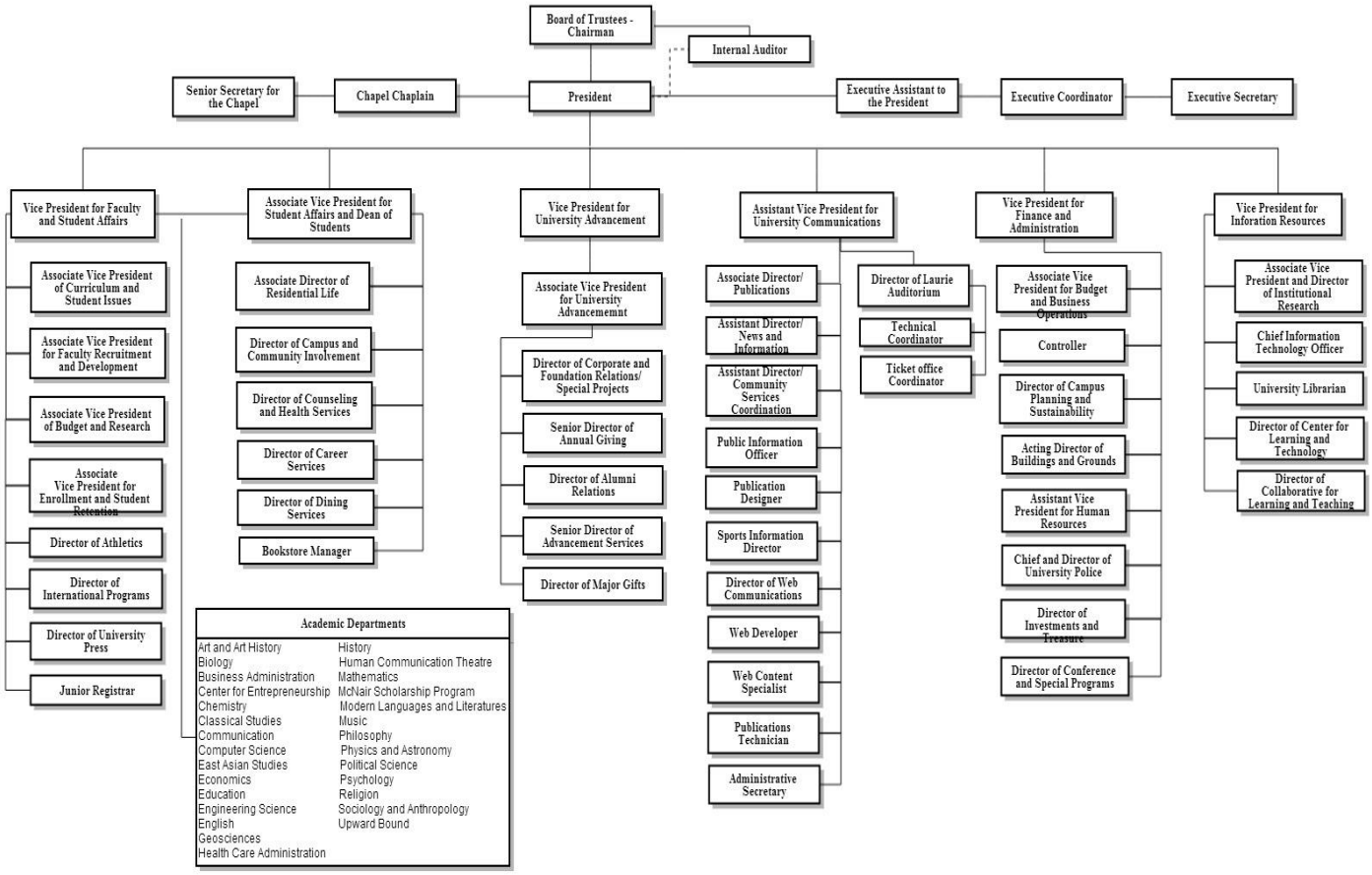
PROPOSED SYSTEM:

UMS (UNIVERSITY MANAGEMENT SYSTEM) 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.



A university has a structural hierarchy wherein there is administration at each level and there are employees under each administration. Each employee will belong to one administration level. Starting with Chairman, President all the way down to Managers and teaching staff, has a role of providing services to the respective department. The academic department has various courses under the Faculty and student affairs tier. There are people playing various roles like Bookstore Manager maintaining bookstore, librarian maintaining library, treasurer maintaining finance, IT desk providing technical support. Each of these employee has a certain role in the university and hence in the database. The database build by me is based by this hierarchy and classification of employees and each tier into their relevant departments.

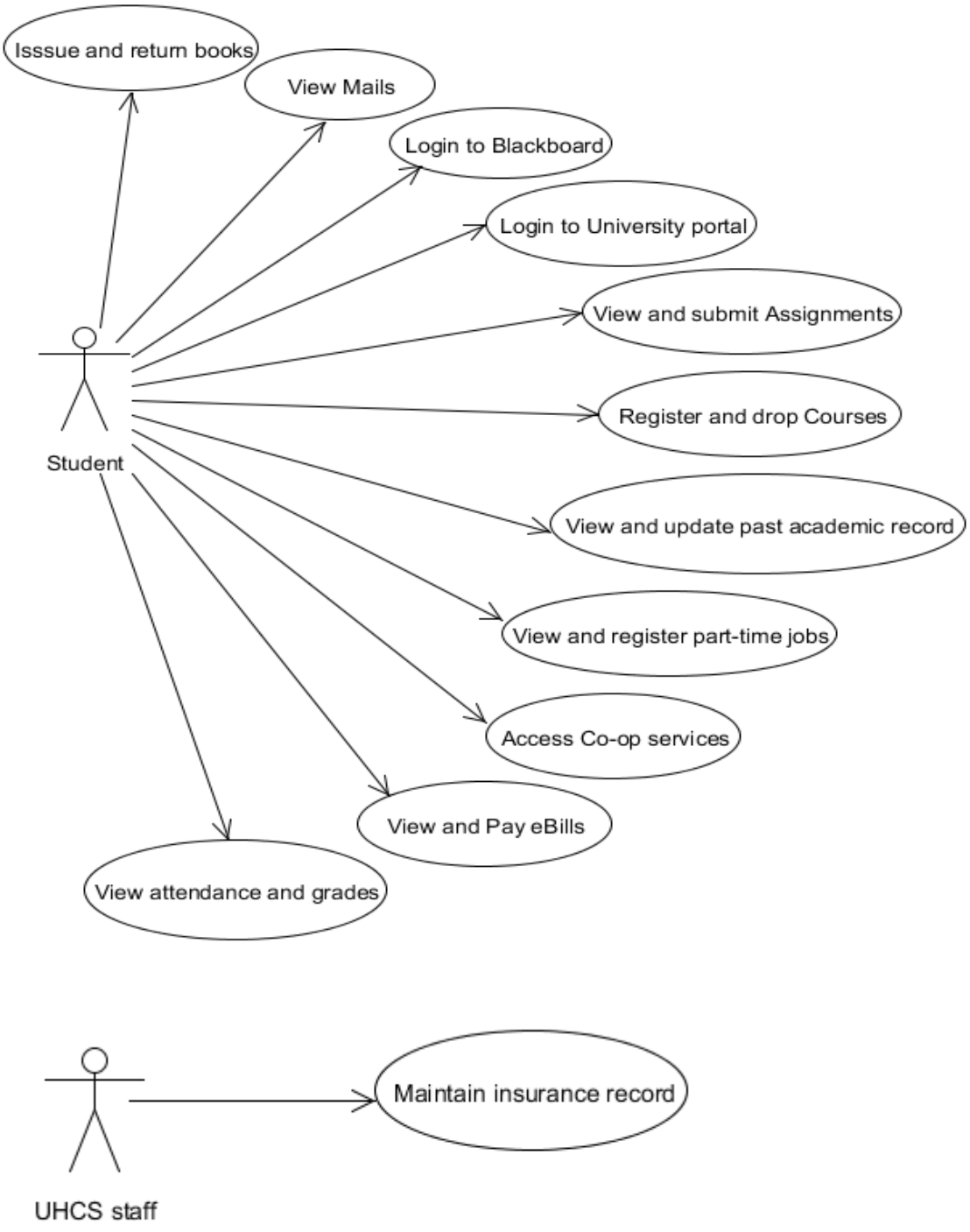
ROLES AND RESPONSIBILITIES:

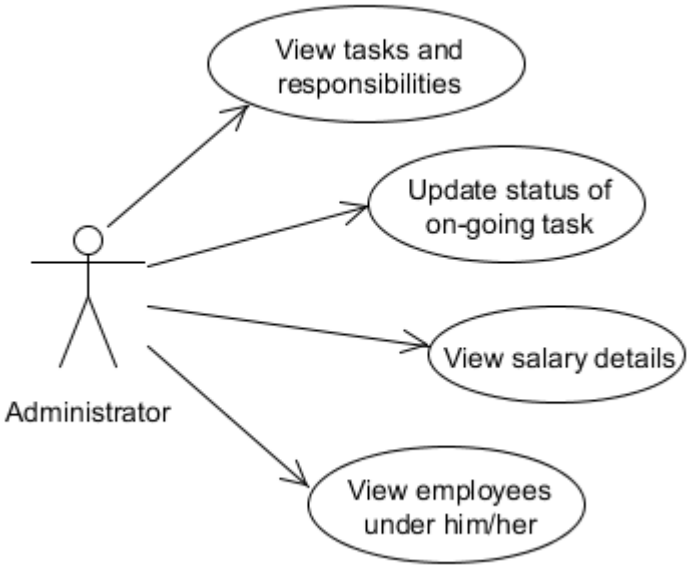
ACTORS:

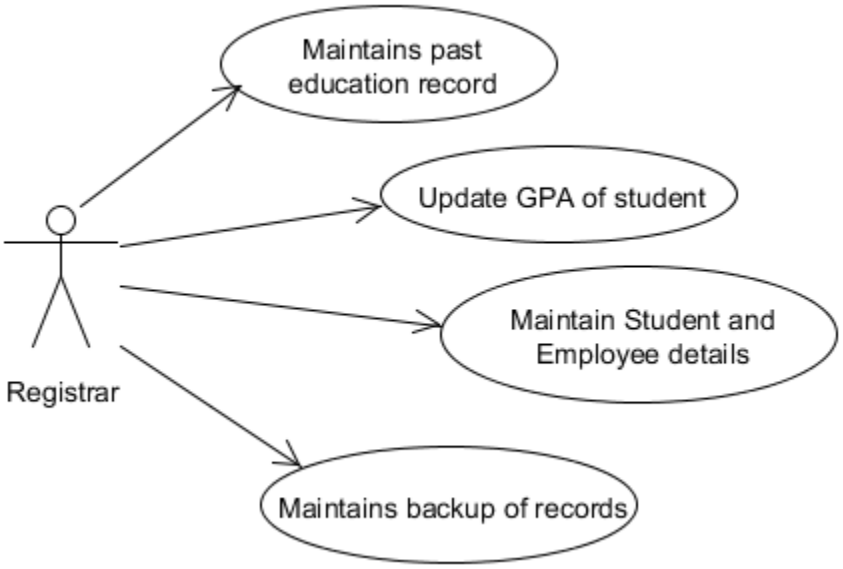
There are various roles played by different people:-

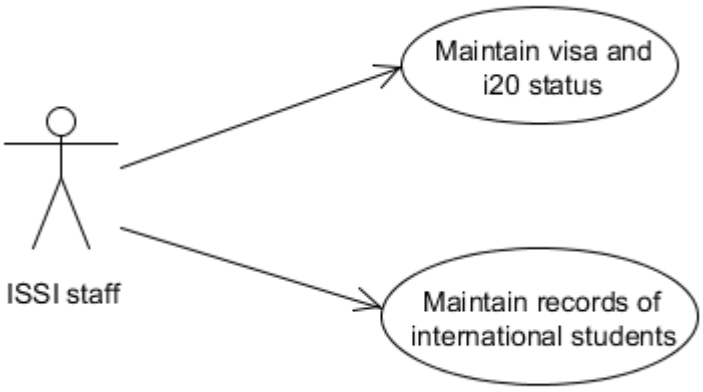
a) Student: The person who uses the application to interact with university and associated employees, assess his/her career opportunities through application portal search & also get professional counseling from experienced advisors. It is instructed by instructors by taking section under various courses offered by the university. 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) Treasurer: The person who supervises the fiscal matter of the university 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 maintain student details be it demographic or academic 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 which can be accessed through online portal.
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) ISSI staff: Maintains international status of students coming abroad for studies and record of their required documents.
h) University Health and Counsel Services (UHCS) staff: An employee who maintain the record of insurance taken by students in case of availing health services.
i) Advisor: A person who provide guidance and monitors the student's term at the university.

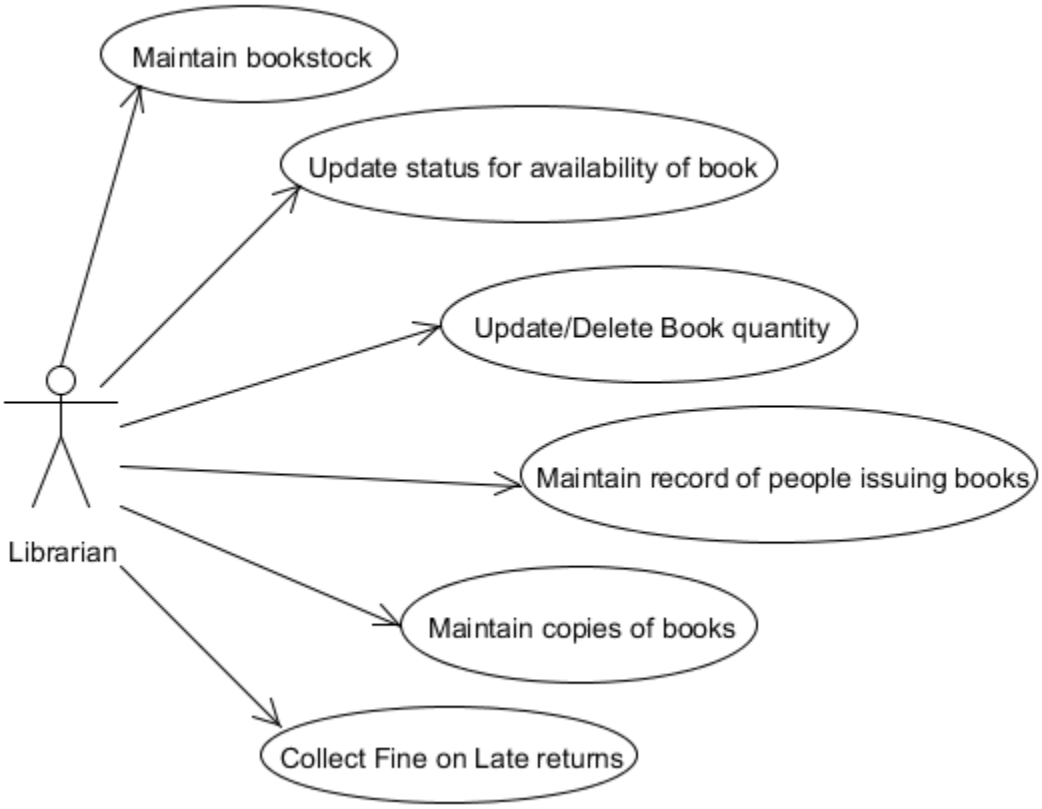
USE CASES:

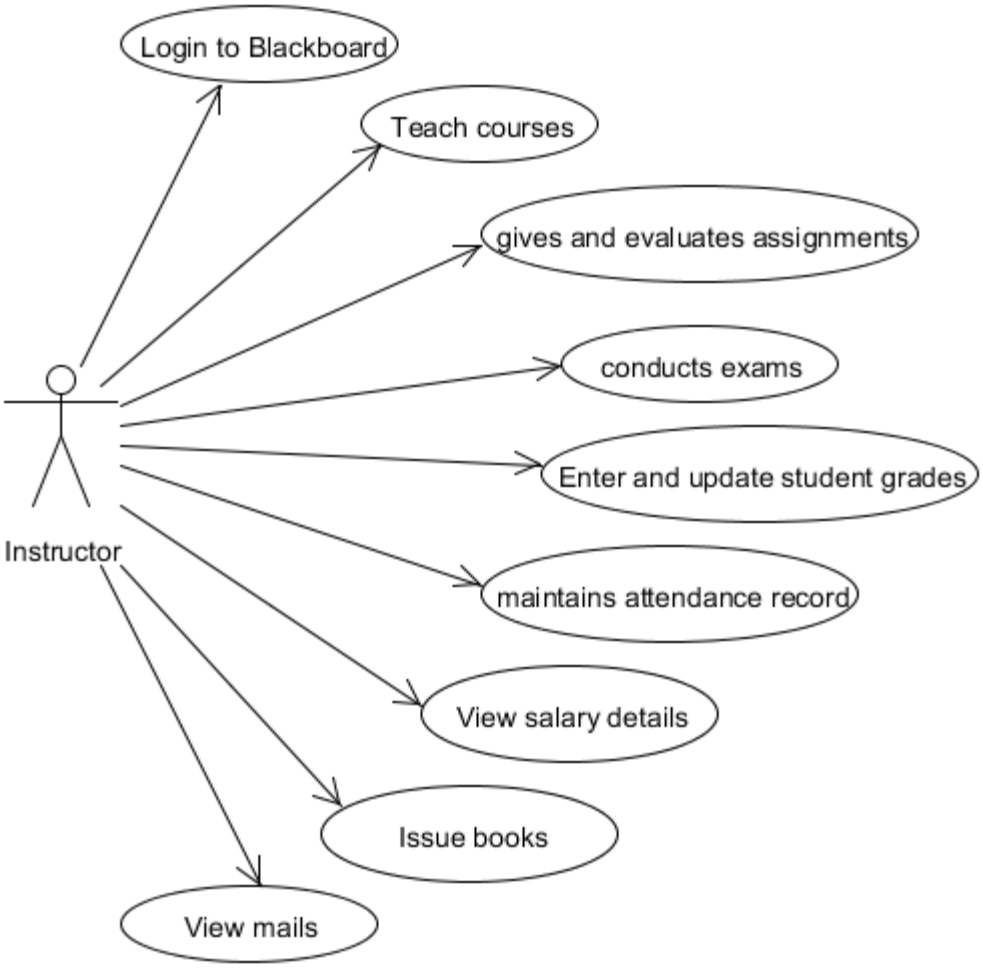

Isssue and return books
View Mails
Login to Blackboard
Login to University portal
View and submit Assignments
Register and drop Courses
View and update past academic record
View and register part-time jobs
Access Co-op services
View and Pay eBills
View attendance and grades
Student
Maintain insurance record
UHCS staff

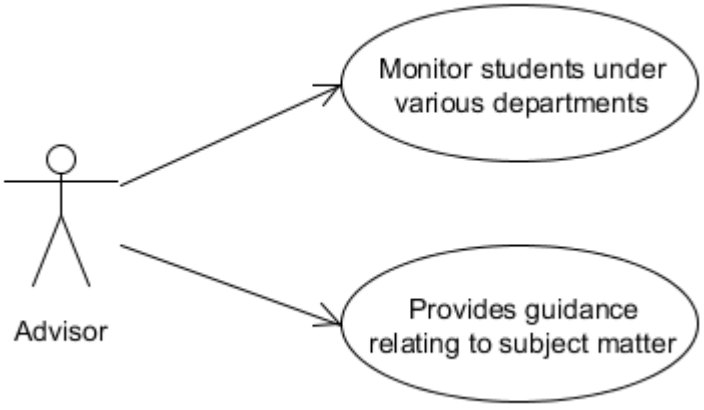

View tasks and responsibilities
Update status of on-going task
View salary details
View employees under him/her
Administrator

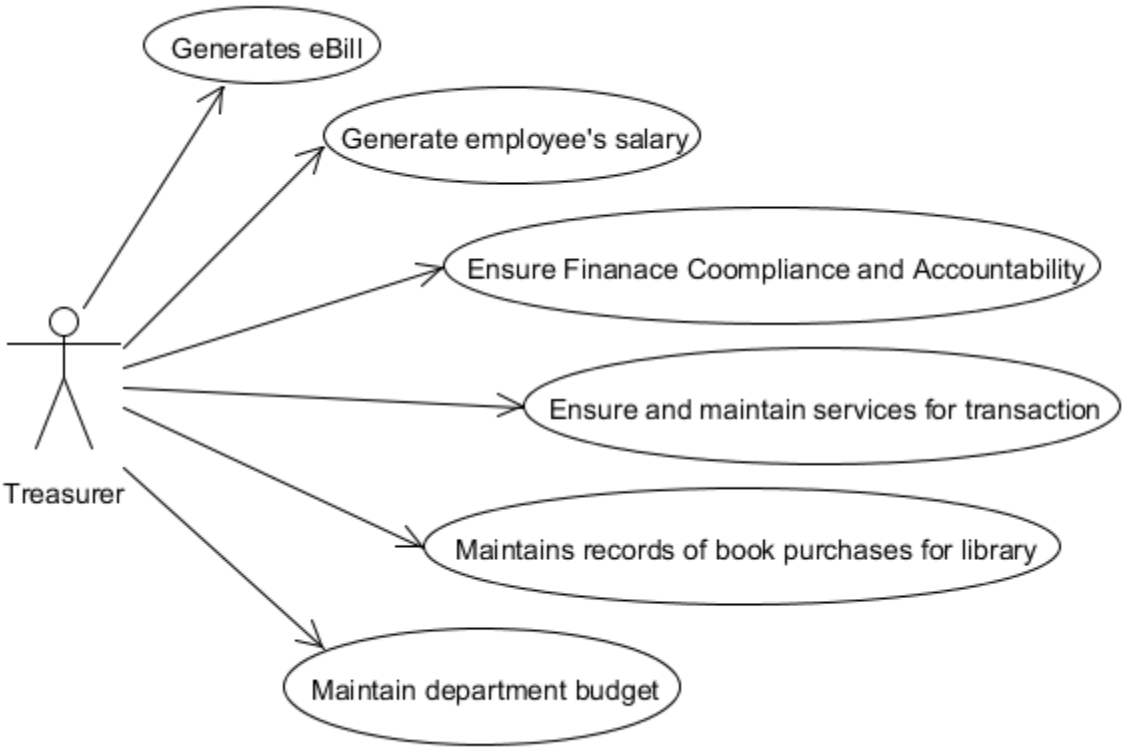

Maintains past education record
Update GPA of student
Maintain Student and Employee details
Maintains backup of records
Registrar


Maintain visa and i20 status
ISSI staff
Maintain records of international students

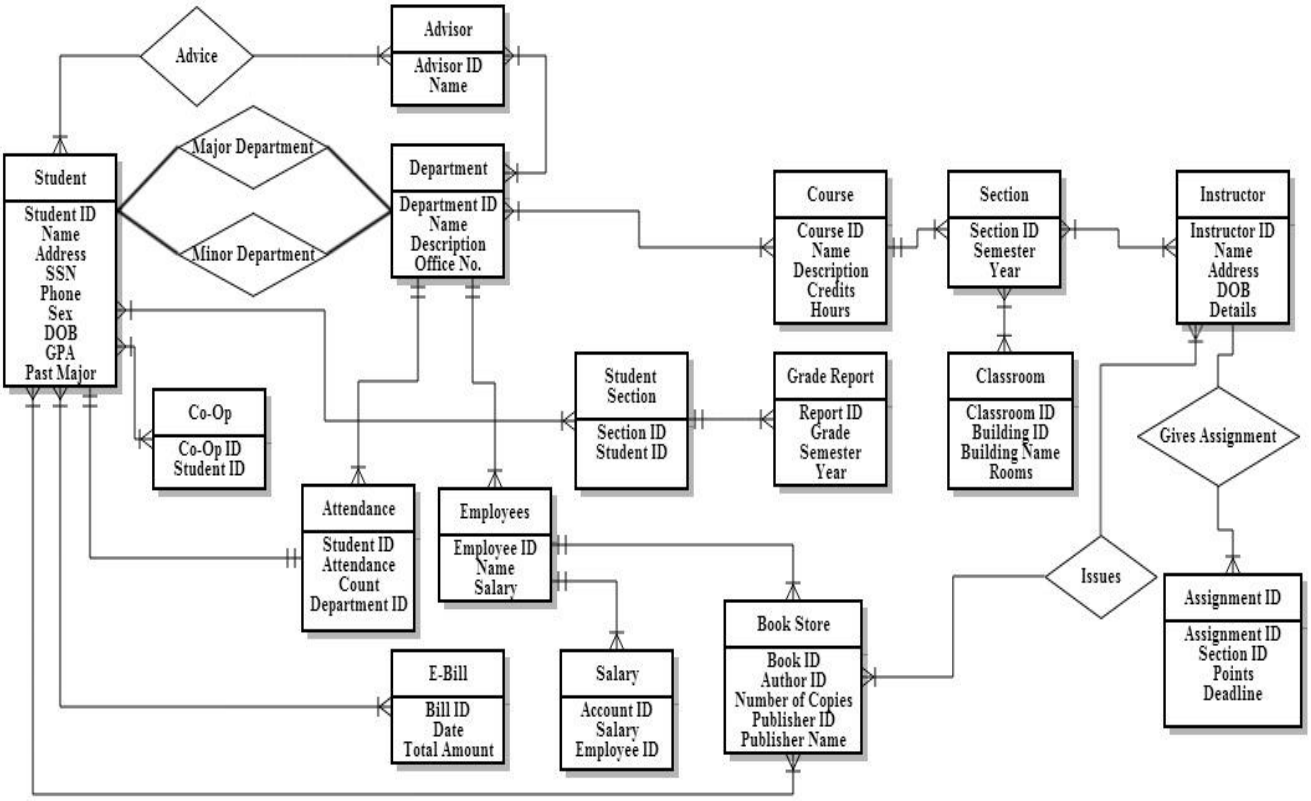

Maintain bookstock
Update status for availability of book
Update/Delete Book quantity
Librarian
Maintain record of people issuing books
Maintain copies of books
Collect Fine on Late returns


Login to Blackboard
Teach courses
gives and evaluates assignments
conducts exams
Enter and update student grades
maintains attendance record
View salary details
Issue books
View mails
Instructor


Monitor students under various departments
Provides guidance relating to subject matter
Advisor


Generates eBill
Generate employee's salary
Ensure Finanace Coompliance and Accountability
Ensure and maintain services for transaction
Maintains records of book purchases for library
Maintain department budget
Treasurer

INTIAL ENTITY-RELATIONSHIP DIAGRAM:



The above diagram shows an initial Entity-relationship diagram which was developed at the beginning of the project to initially understand the relationship between the entities which were initially created.

The following can be understood from ER Diagram:

- The Student is the main actor in the database and most of the table are involved in relationships with the Entity student.
- A student can either belong to a Major or Minor Department. Thus, there exists a one to many relationship between Department and student where a department can have many students whereas a student can belong to one Department.
- There are employees in database, which are basically playing the roles of librarian, advisor, instructor, ISSI staff, Registrar Staff, Health advisory staff etc.
- Both the employees and student have bills where finance staff has to maintain a record and details of both and generate a monthly salary of each employee and a bill semester-wise for students registering for courses.

- A department can have multiple courses which can further have multiple sections taught by instructors.
- A student has to submit assignment which is given by the instructor teaching a section. There exists a many-to-many relationship where a student has to submit multiple assignments and an instructor can give multiple assignments to the students.
- Each student has a grade report which records the courses registered by the student in each semester and the grades obtained by the students in those semester. The students will also have their running GPA's stored in the table.
- A student can go on avail opportunities like part-time jobs and co-op wherein a record has to be maintained about the duration for the same and the respective salaries obtained by the student in the jobs.
- A department can have advisors who can advise multiple students. There exists a many-to-many relationship between advisor and department and department and student.
- A student and a faculty can avail the services offered by the library where they can issue multiple books.
- A student has an attendance record which is maintained by the department he is enrolled into.
- The sections are held in a building which has a one-to-many relationship with the classrooms. The sections are also held in a multiple day slots and have a class timing.
- The courses registered by the students have certain pre-requisite courses which must be undertaken first in order to take courses which have a pre-requisite course. Thus, a mandatory-many cardinality exists between a course and its pre-requisites.
- The student also have a record with ISSI where their International details, arrival and departure in country, passport and visa details are recorded.
- The student also availing dining services have a record and a bill generated accordingly.
- A student also has a health and insurance record with university of health and counsel services wherein insurance details of each student are recorded.

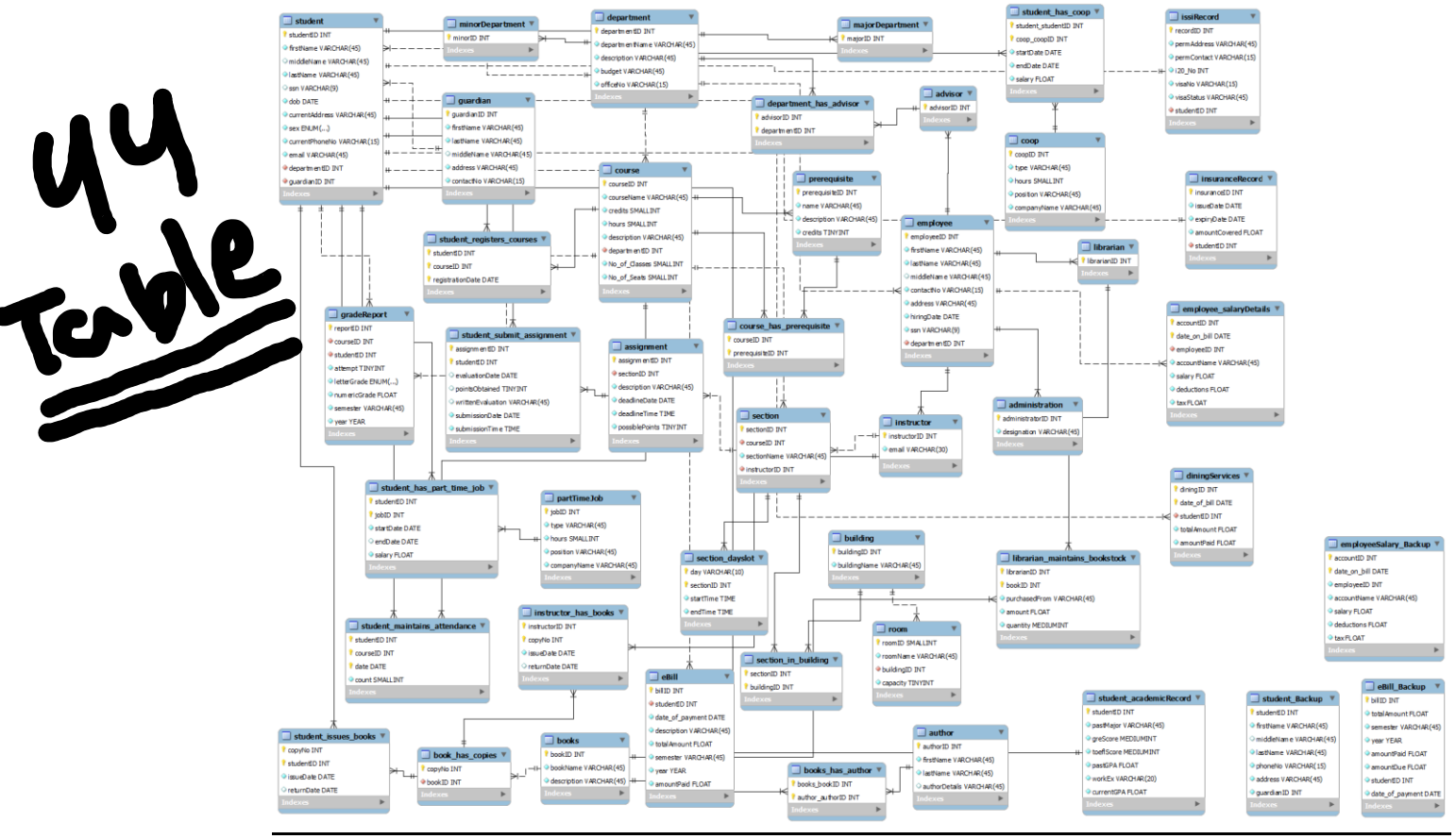
NORMALIZATION:

To confirm that data is consistent across the various entities, following are the steps taken:

- Since a student can either belong to a major or minor department, the major and minor departments are created separately from department which shows total specialization in the ER diagram. The majorID and minorID are separated to show that a student can belong to either of two.
- There are many employees in the database playing different roles. To show the relationship of each role with the entities, a supertype entity named employee is taken

into consideration, which has subtypes named librarian, advisor, instructor etc. Each subtype has different relationships with other entities. Thus, advisor shares many-to-many relationship with department and is separated into another table. Another table, department_has_advisor is taken into consideration to bring the advisor and department table in 3NF.

- 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_salaryDetails where the details of salary slip of each employee will be recorded.
- The past details and academic record as well as current GPA is removed from the student table as there existed partial dependencies of attributes on academic records. The table is in 3NF.
- The author and Copies of Book are removed from the table Books and are taken into new tables namely books_has_author and book_has_copies as there exists many-to-many relationship between authors and books. Also, since a book can have multiple copies, it is separated from the table.
- A course can have many sections with its own instructor. Hence, it is removed from course table.
- A section is held in different building and at different timing, each taught by some instructor. Hence, a new table named section_dayslot is taken into consideration and removed from section table. Similarly, to remove dependencies, buildingID, and roomID is taken into new tables.
- The prerequisite courses are removed from course table, and defined in a new table to normalize the table course.



KEY ENTITIES AND RELATIONSHIP:

- **Student**: The majority of the database deals with storing details of student involved with complex relationship with various entities. A student has various details (attributes) whose values are stored in database under student table. A student can be cared by only one guardian, who can act as a caretaker of one or more students. Thus the table contains foreign key constraint as guardianID. The Address could be taken into different table and shown under details corresponding to zipcode, country, street address, state, country. It is not taken here to reduce the complexity as there is always a trade-off between normalization and performance. A student also have a single record with ISSI deparment, single insurance record with the University health services as well as multiple dining bills if it avails the facility.
- **Department**: This consists of supertype entity in hierarchy with minor and major department as subtype entities. A student may either belong to a major or minor department. Thus a student table also has a foreign key for departmentID which references to this table. Major and minor ID's are thus foreign keys representing major and minor department respectively referencing department table. A department can

have multiple courses, employees which can belong to a single department only. A department can have multiple advisors, and an advisor can provide guidance (advice) multiple departments, showing many-to-many relationship.

- **Course**: Each course under a department has multiple sections, which are taught by different faculties at different days, timings and locations. A course can have multiple pre-requisite courses, who can be a requirement to many courses. This many-to-many relationship is represented by table course_has_prerequisite which has foreign keys referencing to prerequisite course and courseID under course table. Also, a prerequisiteID corresponds to a courseID in course table and is a foreign key.
- **Section**: Each section must belong to any one course. The section can be taught by only one instructor. Hence the table contains foreign keys as instructorID referencing to instructor who teaches it and courseID referring to the course to which it belongs. The timing and location can be described as:

  Section_dayslot: The day and sectionID uniquely determines when are the days where the classes are scheduled

  Section_in_building: This table consists of composite primary key which also references to the building where it will be held. A building can have many rooms with different capacities where these classes are held.
- **Employee**: An employee represent many user roles which play their importance in the database. Hence a supertype entity employee with instructor, advisor, librarian as subtype entities are taken into consideration. Each of this subtype has a primary key ID which is also a foreign key referencing to the employee table. The administration who manages the university also has employees, where the administration staff are themselves the employees working in the university. Thus administratorID acts as a foreign key referencing to this table. Each employee can belong to only one department and thus has a foreign key referencing to department. Each employer has a salary whose record is maintained in a separate table and the table can maintain many records of an employee.
- **Grade report**: Each student registers for multiple courses, whose grade obtained is recorded in this table. This table maintains the grade of each course taken by the student. If the student wish to retakes the course, then another attribute attempt is taken wherein the number of attempts of passing the course are recorded.
- **Assignment:** Each section has to be passed by completing the assignments which are submitted by students. This mentions the section to which it belongs as well as the deadlines of submission. The table student_submit_assignments tells about the students taking that course to which that section belongs, while giving details about the submission time as well as describing the evaluation of the assignment.

- **Jobs and coop:** The student can take multiple part-time jobs as well as co-ops which in turn can be completed by many students. There exists many-to-many cardinality between them.
- **Student maintains attendance**: This relationship gives the count of attendance maintained by the student. The studentID, courseID and date of recording attendance uniquely determines the attendance of each student who have registered multiple courses. Thus even if a student retakes a course, the record date will uniquely determine the count of each registered course.
- **eBill:** The student have their separate financial record maintained by finance department. A student having multiple e-bills for each semester in which he/she takes courses, has a foreign key referencing to student table. It also gives the amount paid and can be used to calculate dues.
- **Student_issues_books & Instructor_has_books:** A student as well as instructor can issue multiple books thus having a composite primary key determined by book's ID and student and instructor ID's respectively
- **Book:** A book can have its own description, details of author as well as multiple copies. Each of these attributes are taken uniquely into their own tables. Book_has_copies determines the copies of the book in the library uniquely determined by its copyID. Thus the copies are actually the books issues by instructor and student. A book can have multiple authors, each of one can be a writer of several books.

SECURITY ISSUES:

This section of the document deals with the security issues of the system. The system is made secure using all possible secure coding methods (SSL, Cache Clearance, Anti CSRF) to avoid breach of data integration. The system deals with the concept of views which are used to control the data that is accessed by the application user. There are two views in the procedure involving the admin view and student view.

The application users have no access to the database directly, they can access the data only the web secure online enterprise application.

Application Security

The application is made secure by assigning unique usernames to all users and password is not made visible to any of the system users or administrators and is customer specific. The application makes use of https, ssl and verisign secure to make secure all kind of financial and business critical data transactions.

Views used in the application: (STUDENT VIEW)

```
USE `universityManagementSystem`;

create  OR REPLACE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER view student_view as (select s.firstName, c.courseName, se.sectionID, a.description, sma.count

from student s

inner join student_registers_courses

inner join course c

inner join section se

inner join assignment a

inner join student_maintains_attendance sma

where(s.studentID = student_registers_courses.studentID and student_registers_courses.courseID = c.courseID and c.courseID = se.courseID and se.sectionID = a.sectionID and (sma.studentID = s.studentID and sma.courseID = c.courseID))

);
```

PROCEDURES AND TRIGGERS:

PROCEDURES:

1. The following procedure gives the details of the student who have co-ops, their salary as well as duration and give their academic detail such as current gpa.

```
DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `routine1` (studentID INT)

BEGIN

select student.firstName, coop.companyName, student_has_coop.startDate,
student_has_coop.endDate, student_has_coop.salary,
student_academicRecord.currentGPA from student_has_coop

inner join student

on student_has_coop.Student_studentID = student.studentID

inner join coop

on student_has_coop.coop_coopID = coop.coopID

inner join student_academicRecord

on student_academicRecord.studentID=student.studentID

where student_has_coop.student_studentID = studentID;

END
```

2. The following routine tells about the students who secured A grade in the courses registered by them:

```
DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `routine2` ()

BEGIN

select student_academicRecord.currentGPA, student_academicrecord.studentID from
student_academicRecord where

student_academicRecord.studentID IN (select gradeReport.studentID from gradeReport
where gradeReport.letterGrade = 'A');

END
```

3. The following routine tells us about the grades obtained by the students in the courses registered by them who have part-time jobs:

```
DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `routine3` (studentID int)
BEGIN
select student.firstName, course.courseName,
gradeReport.numericGrade,student_has_part_time_job.jobID from
course inner join gradeReport
on gradeReport.courseId = course.courseId
inner join student
on gradeReport.studentID=student.studentID
inner join student_registers_courses
on student_registers_courses.courseID= gradeReport.courseID and
student_registers_courses.studentID=gradeReport.studentID
inner join student_has_part_time_job
on student_has_part_time_job.studentID=gradeReport.studentID
where student_registers_courses.studentID = studentID;
END
```

4. The following procedure tells us about the remaining seats in the course which are currently taken by the student

```
DELIMITER $$

CREATE PROCEDURE `routine4` ()
BEGIN
DECLARE a INT;DECLARE b INT;DECLARE c INT;
```

```
SELECT No_of_Seats INTO a FROM course WHERE courseID=2000;

SELECT count(courseID) INTO b FROM student_registers_courses WHERE
courseID=2000;

SET c = a-b;
SELECT c as remainingSeats;
END
```

INDEX:

Four indexes were created to deliver quicker access of data. The indexes were applied on key columns often used during data transactions.

```
create index student_Idx on student(studentID);

create index course_Idx on course(courseID);

create index section_Idx on section(sectionID);

create index book_Idx on books(bookID);
```

TRIGGERS:

1. This trigger runs before delete on guardian backup table where the data if deleted from the guardian table can be entered safely into backup table.

```
Delimiter //
CREATE TRIGGER backup_guardian BEFORE DELETE ON guardian
FOR EACH ROW
BEGIN
   INSERT INTO
 GuardianBackUp(guardianID, firstName, middleName, lastName, address, contactNo)
values(OLD.guardianID, OLD.firstName, OLD.middleName, OLD.lastName, OLD.address,
OLD.contactNo);
```

```
END//
```

2. This trigger runs when the student records are updated and is entered into student backup table.

```
Delimiter //
CREATE TRIGGER backup_student BEFORE UPDATE ON student
FOR EACH ROW
BEGIN
   INSERT INTO
   student_Backup(studentID, firstName, middleName, lastName, address, phoneNo,
guardianID)
         values(Old.studentID,Old.firstName,Old.middleName,Old.lastName,Old.currentA
ddress,Old.currentPhoneNo,Old.guadianID);
END//
```

3. This trigger runs when a student returns a book and is now available for others. The status of the book is changed to available.

```
Delimiter //

CREATE TRIGGER statusReturn_books AFTER UPDATE ON student_issues_books
FOR EACH ROW
BEGIN
  if(NEW.returnDate > OLD.returnDate) THEN
update book_has_copies
set status = 'Available'
where copyNo = New.copyNo;
END IF;
END//
```

4. This trigger sets the status of the book to unavailable if it is issued by somebody.

```
Delimiter //
CREATE TRIGGER statusUpdate_books AFTER INSERT ON student_issues_books
FOR EACH ROW
BEGIN
  if(NEW.returnDate = 0000-00-00) THEN
update book_has_copies
set status = 'Unavailable'
where copyNo = New.copyNo;
```

```
END IF;
END//
```

PRIVILEDGES:

```
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'%'
GRANT SELECT ON student_registers_courses.* TO 'student'@'%'
GRANT UPDATE ON student TO 'student'@'%'
GRANT SELECT ON gradeReport TO 'student'@'%'
GRANT SELECT ON student_maintains_attendance TO 'student'@'%'
GRANT SELECT ON eBill TO 'student'@'%'
GRANT SELECT ON diningservices TO 'student'@'%'
GRANT SELECT ON student_academicrecord TO 'student'@'%'
GRANT SELECT ON insurancerecord TO 'student'@'%'
GRANT SELECT ON student_issues_books TO 'student'@'%'
GRANT SELECT ON assignment TO 'student'@'%'
GRANT SELECTON student_submit_assignment TO 'student'@'%'


GRANT SELECT,UPDATE ON instructor_has_books TO 'instructor'@'%'
GRANT SELECT,UPDATE,INSERT ON assignment TO 'instructor'@'%'
GRANT INSERT,UPDATE ON student_maintains_attendance TO 'instructor'@'%'
GRANT SELECT, UPDATE, INSERT, DELETE ON student_registers_courses TO 'instructor'@'%'
GRANT SELECT ON employee_salarydetails TO 'instructor'@'%'
GRANT SELECT ON section TO 'instructor'@'%'

GRANT SELECT, UPDATE, INSERT ON librarian_maintains_bookstock TO 'librarian'@'%'
GRANT SELECT,UPDATE,INSERT,DELETE ON book_has_copies TO 'librarian'@'%'GRANT
SELECT,UPDATE,INSERT,DELETE ON books TO 'librarian'@'%'
GRANT SELECT,UPDATE,INSERT,DELETE ON student_issues_books TO 'librarian'@'%'
GRANT SELECT,UPDATE,INSERT,DELETE ON instructor_has_books TO 'librarian'@'%'
GRANT SELECT,UPDATE,INSERT,DELETE ON books_has_author TO 'librarian'@'%'
GRANT SELECT,UPDATE,INSERT,DELETE ON author TO 'librarian'@'%'

GRANT SELECT ON department_has_advisor TO 'advisor'@'%'
GRANT SELECT ON employee_salarydetails TO 'advisor'@'%'
GRANT SELECT ON employee TO 'advisor'@'%'
```