American International University-Bangladesh

Advanced Database Management System (SECTION:B)

Project: *UniversityRegistration Management System*

GROUP NO:

Student Name	Student Id
Arindam Shaha Niloy	**19-40661-1**
Saiful Islam	**17-35218-2**
MD. GOLAM RABBANI RAFI	**19-41123-2**
MD. RAHANUR NISHAN	**20-43093-1**
AZRAF FAHEEM	**20-42152-1**

INTRODUCTION:

The objective of this project is to develop a course registration management system that will simplify the registration process for educational institutions and enable students to register for courses efficiently. The system will also include features for managing student records, generating reports, and communicating with students.

PROJECT PROPOSAL:

Scope:

The course registration management system will include the following features:

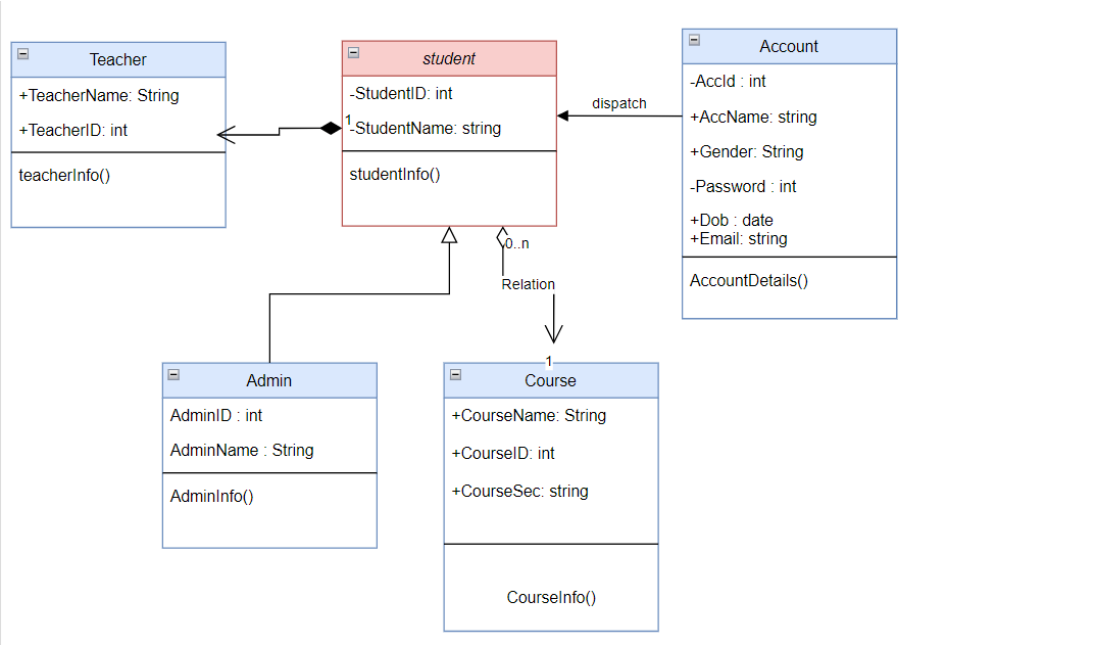
Feature Management:

1. **Student:**
   Login/Registration request: Student have to an account to Login. If he/she have no account, he/she have to register first. For the registration student need to give all personal information (Name, Date of Birth, Email, Phone Number, Phone Number, Father's name, Mother's name, Gender, Password, Confirm Password). Student have to make a request for registration. Admin have the access to approve/reject the registration request. While log in to the account student can visit his/her profile info.
   Select course: In the registration page student can select course. He/she can select course by suitable time. Every course has some section and every section has a time schedule. Every course has some credits also.
   Capacity of section: If the maximum capacity reached of section, students can't select that section until the respectful faculty/admin offered seat to that section.

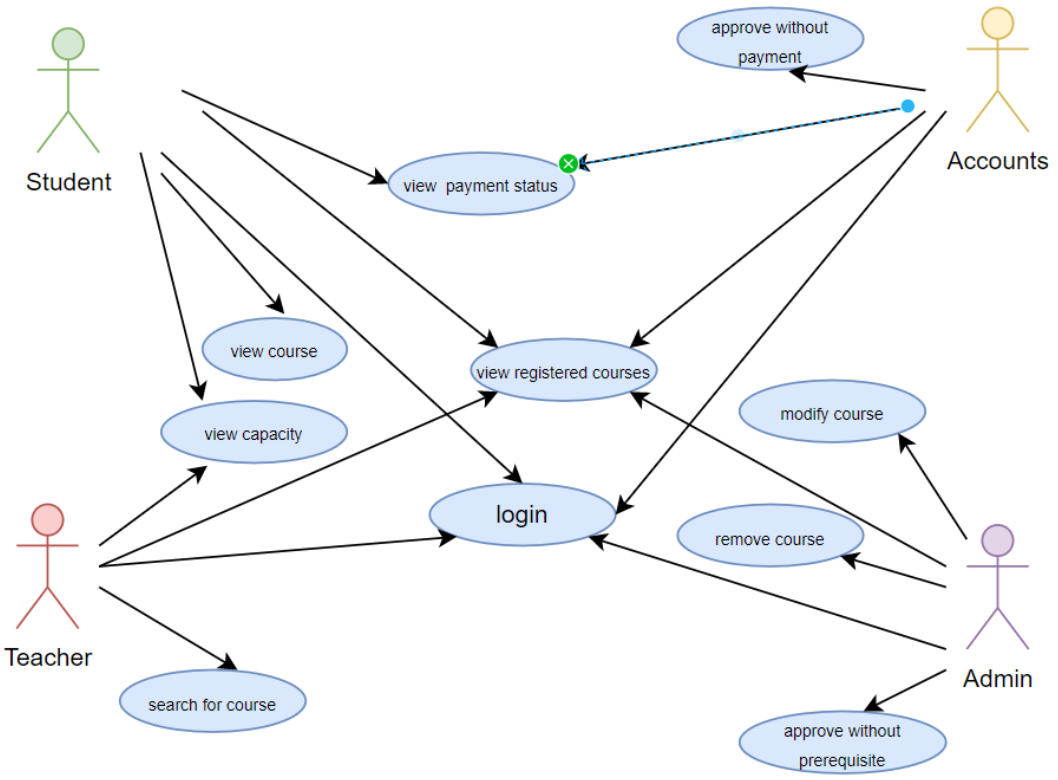
There are some limitations for student. Student can't select a course until the prerequisite course is completed. Student can't select section if he/she already select maximum credit's course for that semester. After taking courses student have to confirm his registration.

2. **Faculty**:
   Log in/registration request: Like students' faculty have a log in page. If his information hasn't register yet then he has to make registration request like students.
   Searching: He/she can visit his own profile and also can visit a student account by the student's user id. He/she can visit a course information also (How many sections available, how many seat available of a section etc.).
   Offer course: He/she can offer section to the students. He can increase seat of any course section. He/she will able to create section, cancel section, increase capacity.
3. **Admin**: Admin have access to change anything. Admin can approve/reject student/faculties registration request. Admin can add any student/faculty. He/she can visit any student or faculties profile. He can change course information.

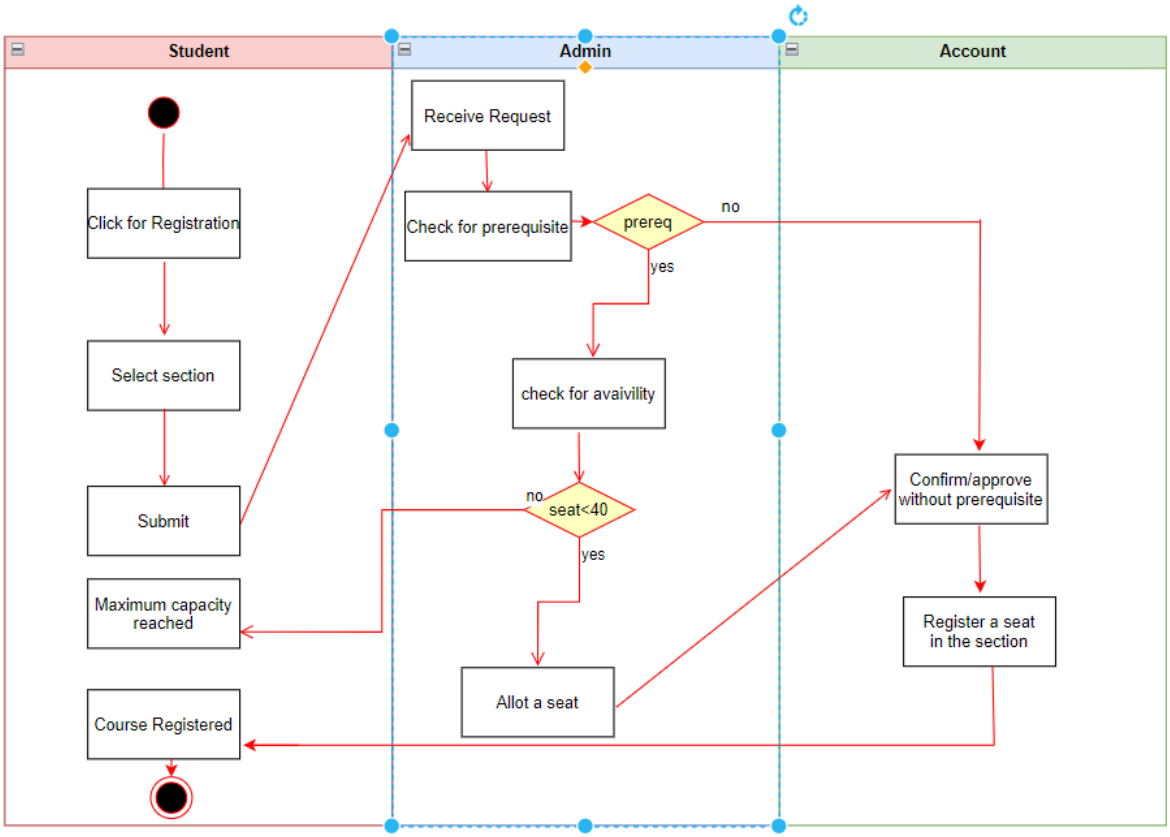
CLASS DIAGRAM



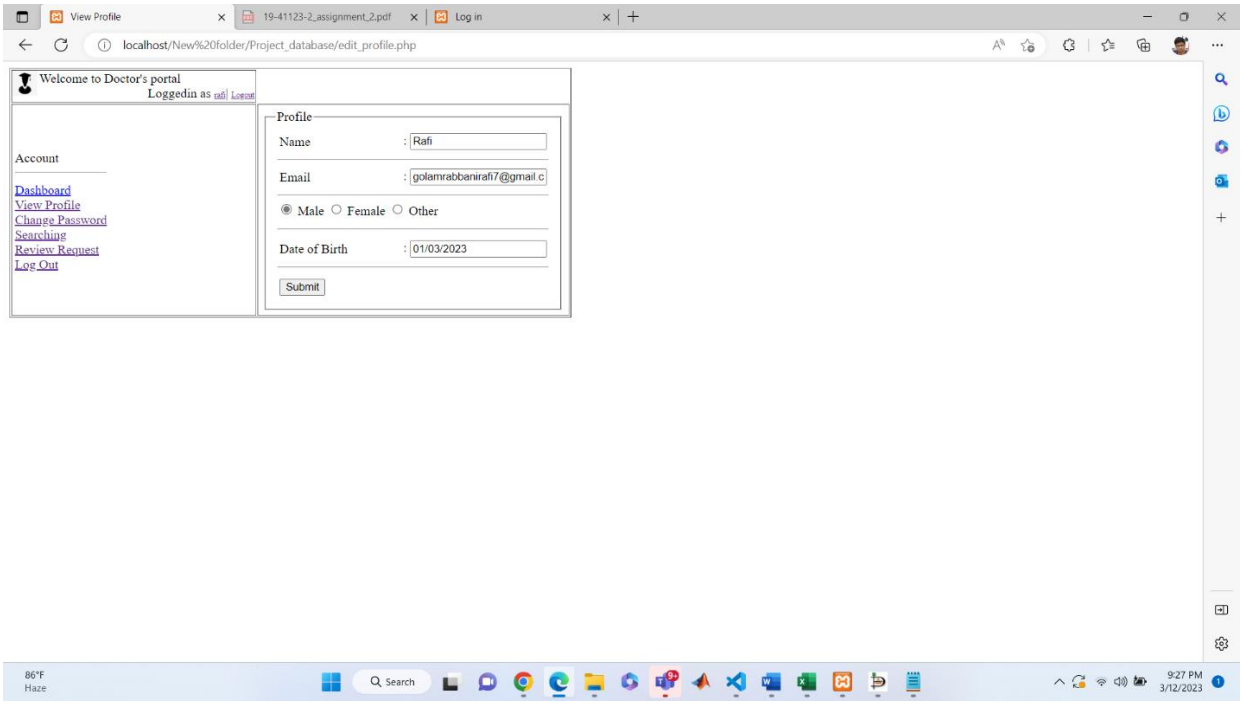
USE CASE DIAGRAM

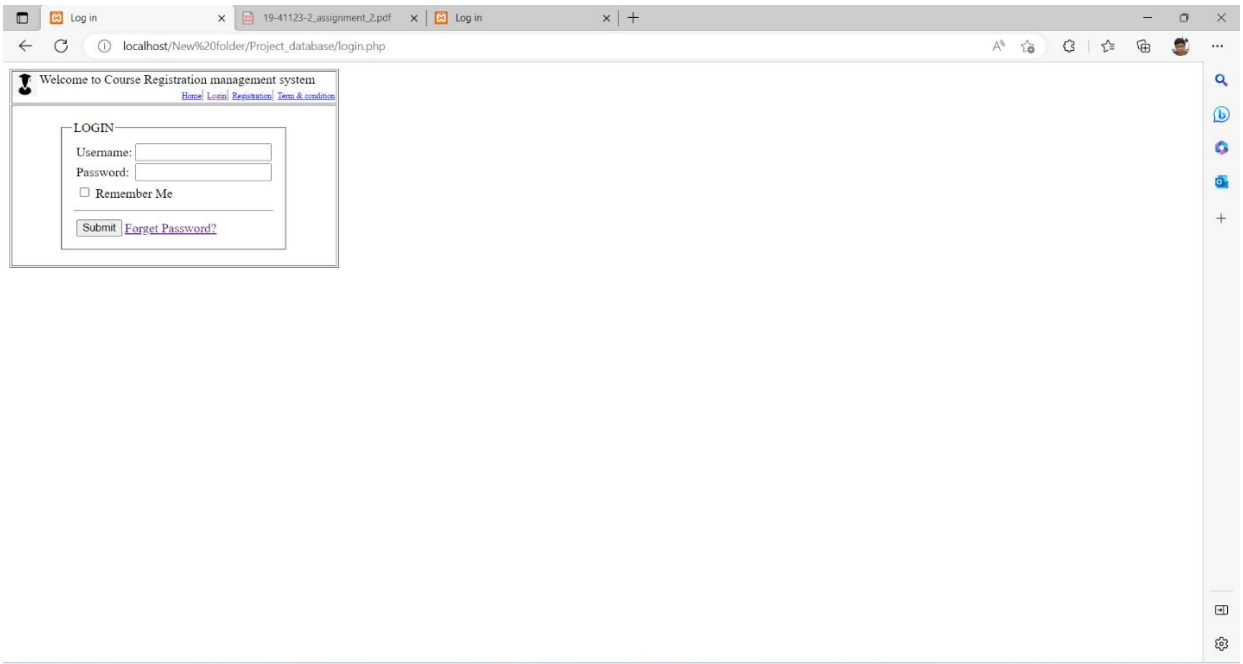


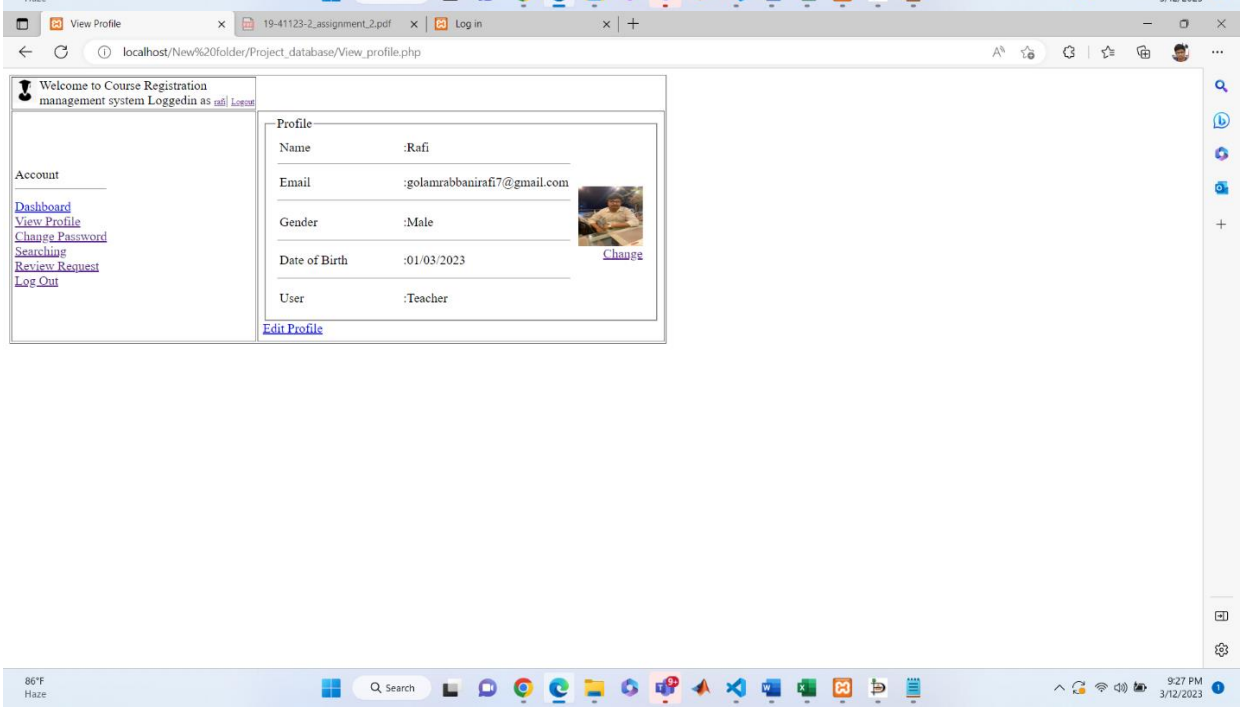
ACTIVITY DIAGRAM

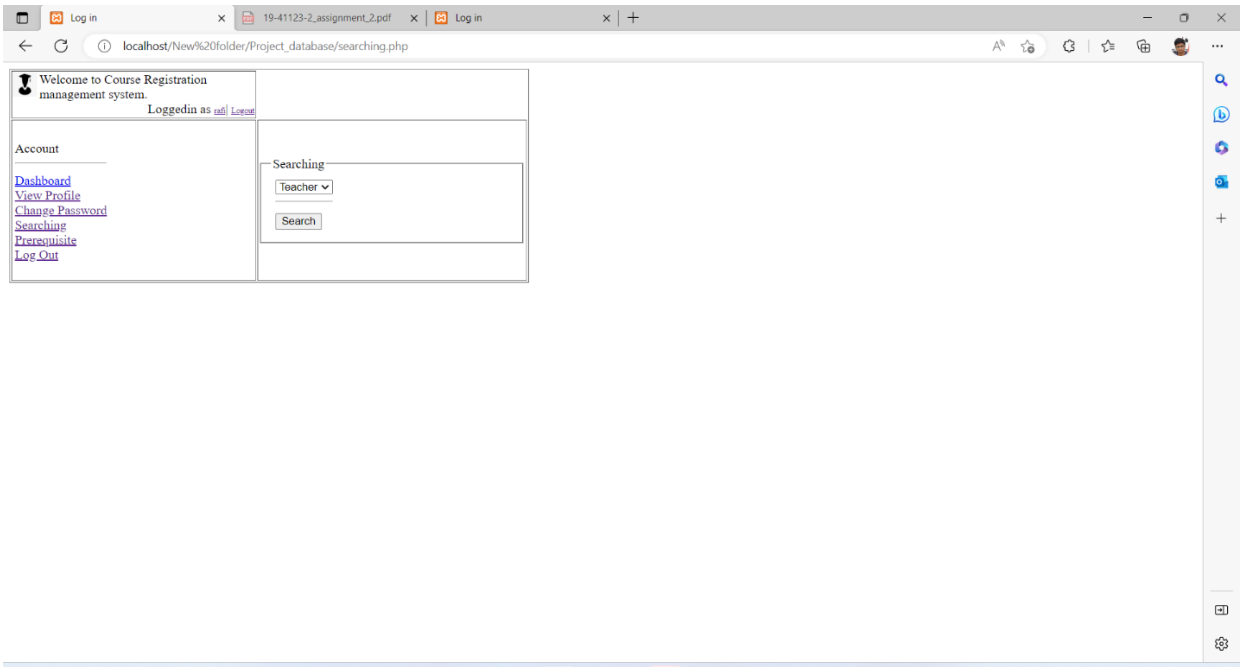


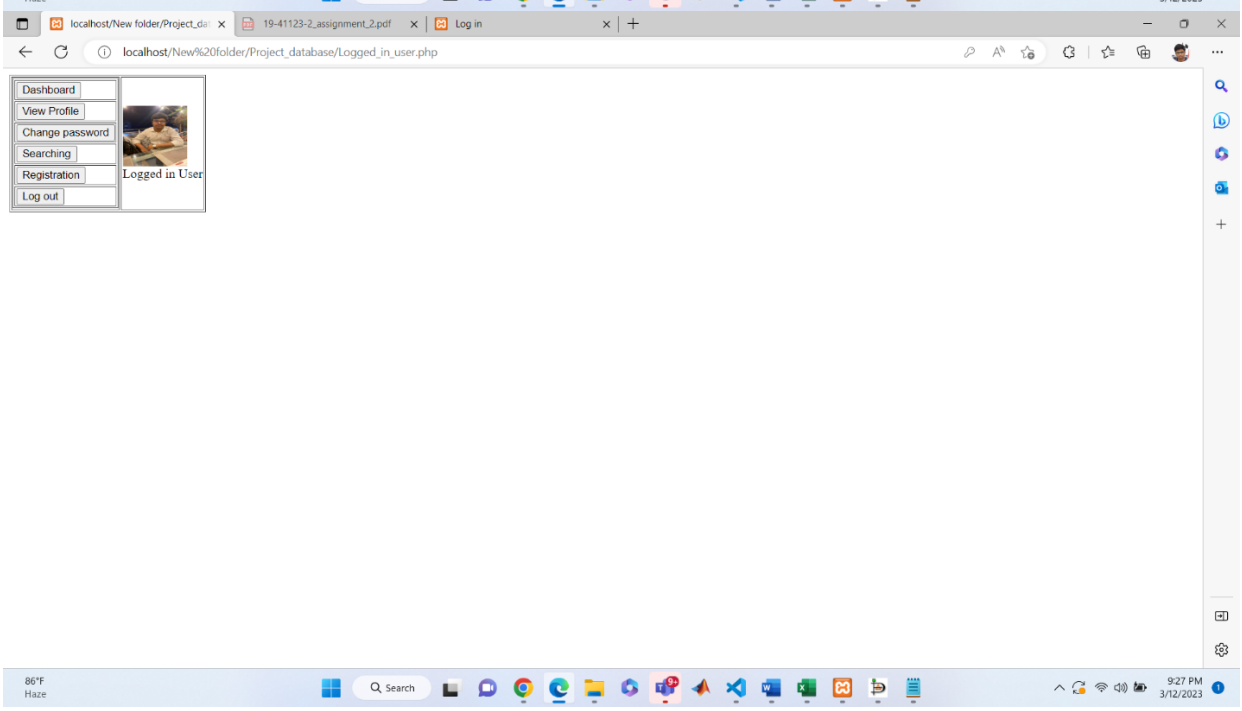
USER INTERFACE

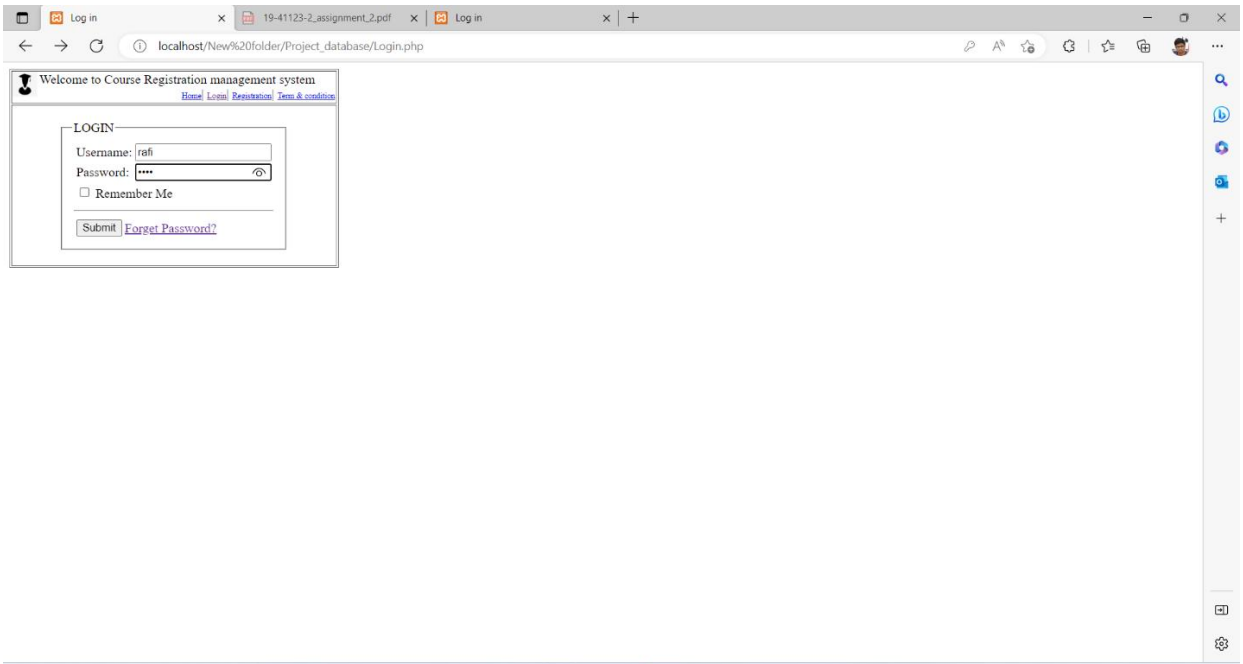


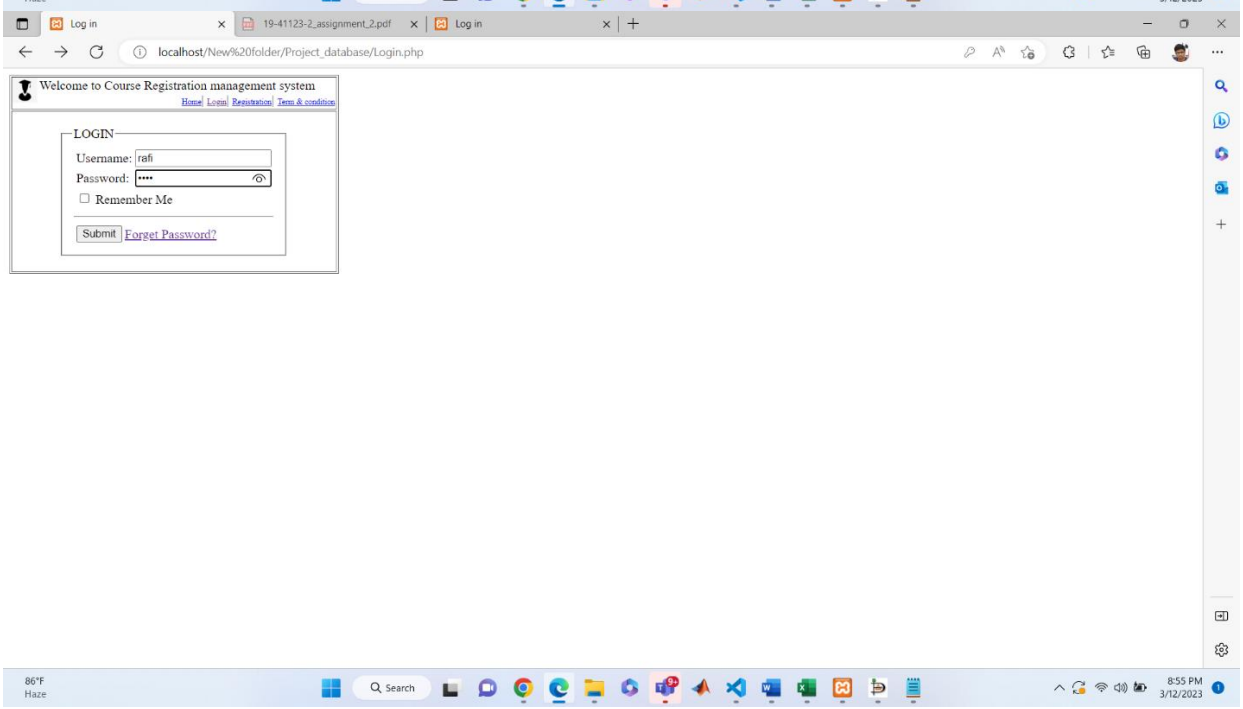

Log in
19-41123-2_assignment_2.pdf
Log in
localhost/New%20folder/Project_database/login.php
Welcome to Course Registration management system
Home| Login| Registration| Term & condition
LOGIN
Username:
Password:
Remember Me
Submit
Forget Password?

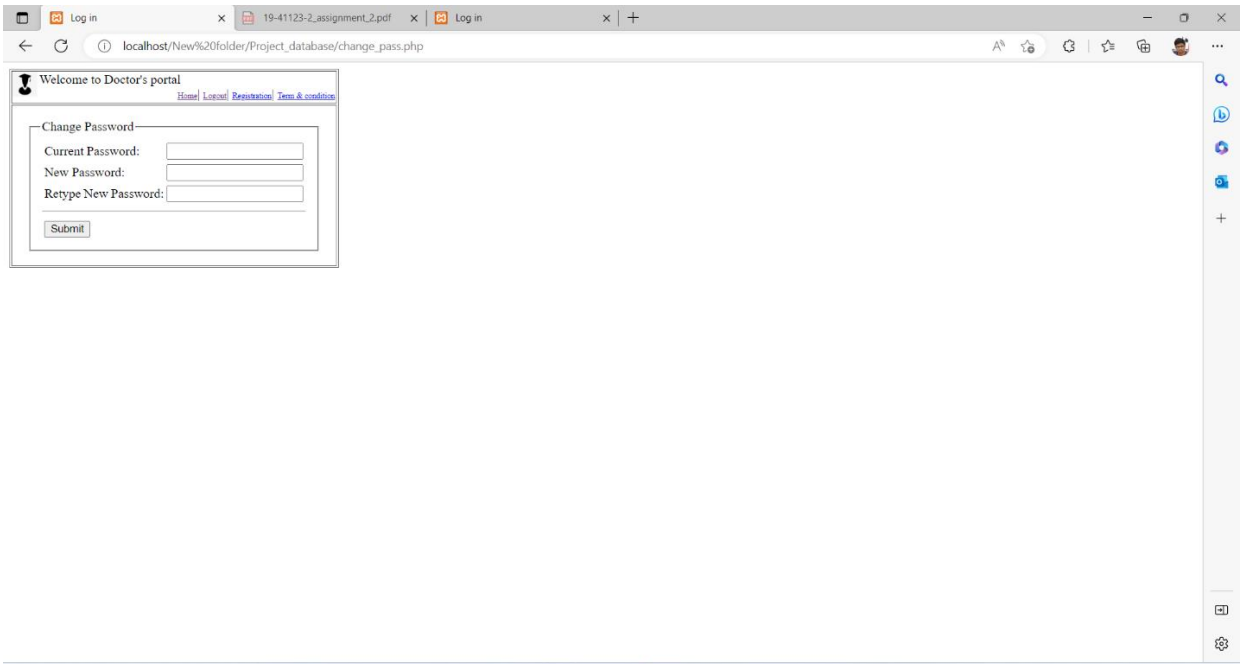

View Profile
19-41123-2_assignment_2.pdf
Log in
localhost/New%20folder/Project_database/View_profile.php
Welcome to Course Registration management system Loggedin as rafi| Logout
Account
Dashboard
View Profile
Change Password
Searching
Review Request
Log Out
Profile
Name :Rafi
Email :golamrabbanirafi7@gmail.com
Gender :Male
Date of Birth :01/03/2023
User :Teacher
Change
Edit Profile

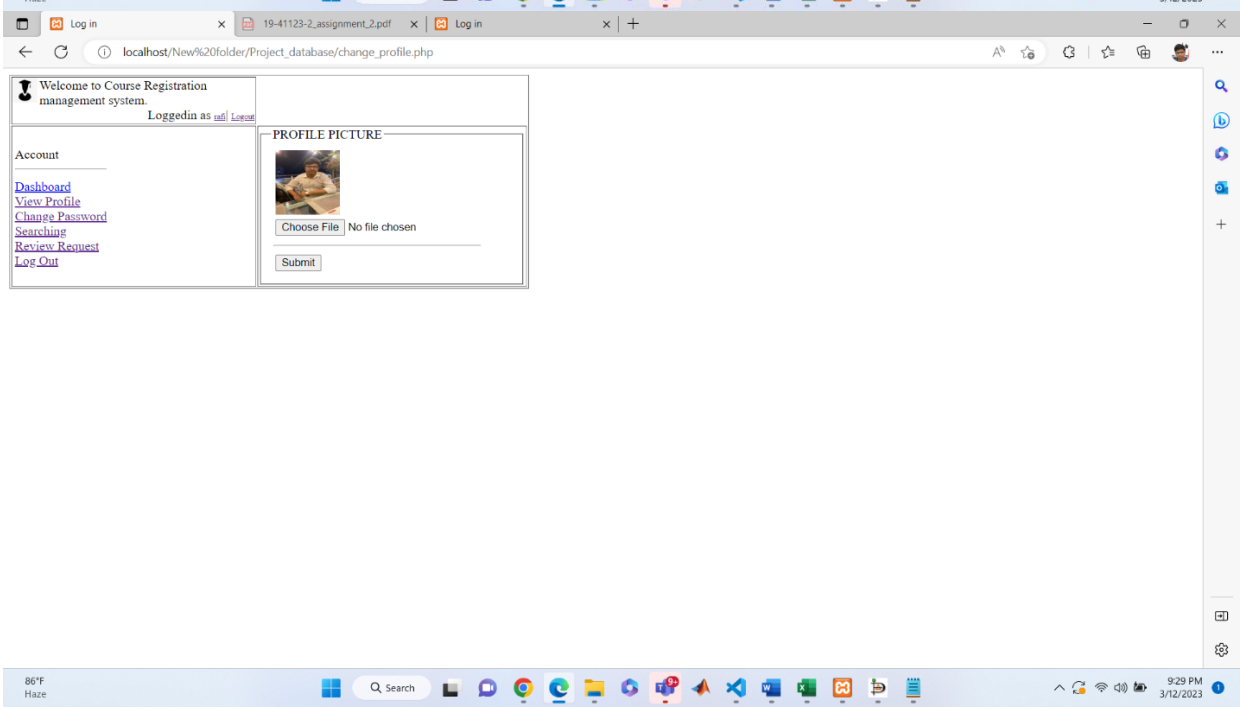

Log in
19-41123-2_assignment_2.pdf
Log in
localhost/New%20folder/Project_database/searching.php
Welcome to Course Registration management system.
Loggedin as rafi Logout
Account
Dashboard
View Profile
Change Password
Searching
Prerequisite
Log Out
Searching
Teacher
Search


localhost/New folder/Project_da
19-41123-2_assignment_2.pdf
Log in
localhost/New%20folder/Project_database/Logged_in_user.php
Dashboard
View Profile
Change password
Searching
Registration
Log out
Logged in User


Log in
19-41123-2_assignment_2.pdf
Log in
localhost/New%20folder/Project_database/Login.php
Welcome to Course Registration management system
Home Login Registration Term & condition
LOGIN
Username: rafi
Password:
Remember Me
Submit
Forget Password?
86°F
Haze
Search
8:55 PM
3/12/2023


Log in
19-41123-2_assignment_2.pdf
Log in
localhost/New%20folder/Project_database/Login.php
Welcome to Course Registration management system
Home Login Registration Term & condition
LOGIN
Username: rafi
Password:
Remember Me
Submit
Forget Password?
86°F
Haze
Search
8:55 PM
3/12/2023

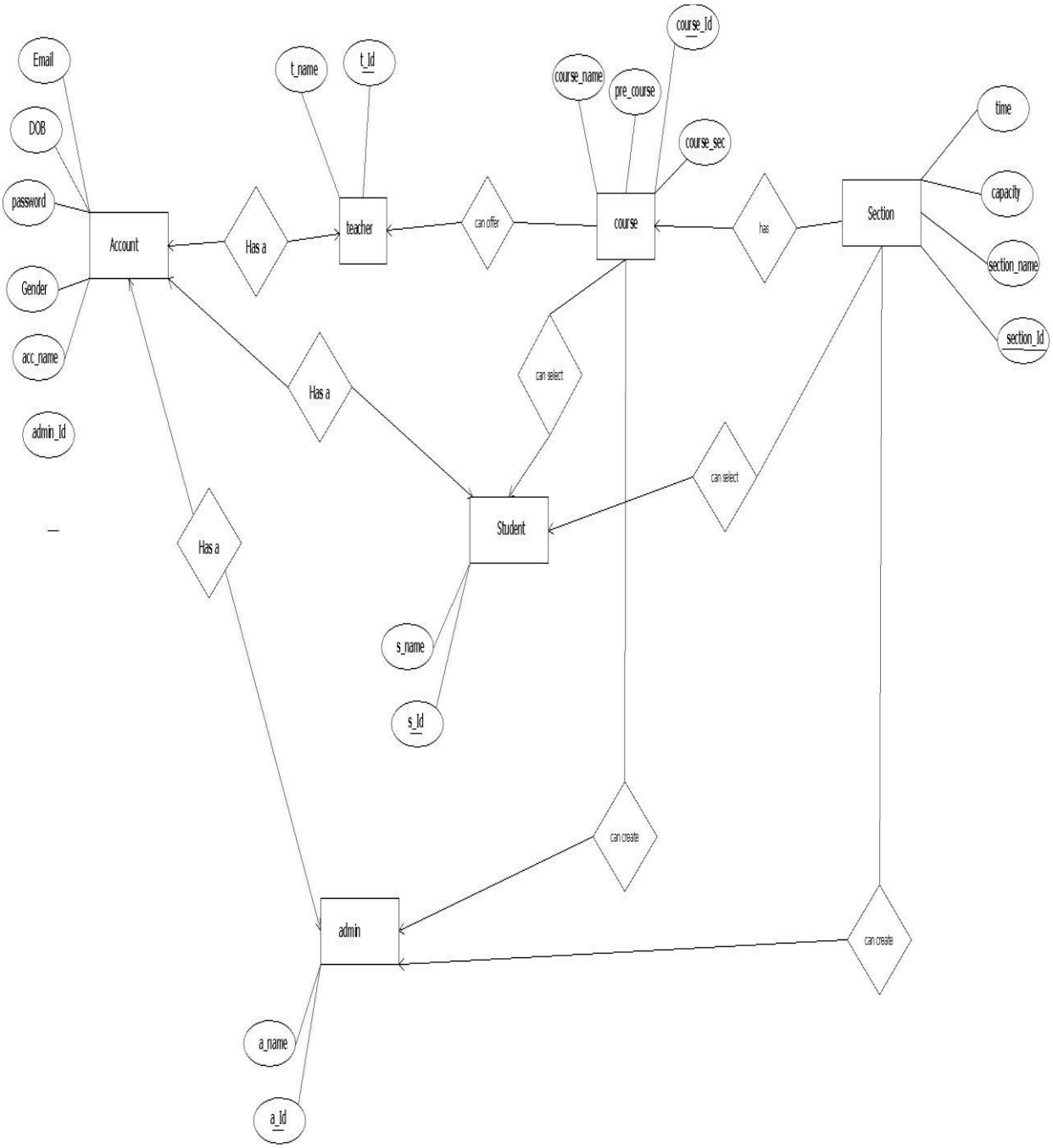

Log in
19-41123-2_assignment_2.pdf
Log in
localhost/New%20folder/Project_database/change_pass.php
Welcome to Doctor's portal
Home| Logout| Registration| Term & condition
Change Password
Current Password:
New Password:
Retype New Password:
Submit


Log in
19-41123-2_assignment_2.pdf
Log in
localhost/New%20folder/Project_database/change_profile.php
Welcome to Course Registration management system.
Loggedin as rafi| Logout
Account
Dashboard
View Profile
Change Password
Searching
Review Request
Log Out
PROFILE PICTURE
Choose File
No file chosen
Submit

SCENARIO DESCRIPTION

This is a course registration management system of our university. Admin is the main power of our university. All admin has a account identified by (a_name,a_id,email,dob,password,gender). Admin can create by lots of courses(course_name,course_id,course_sec,pre_course). He(admin) also created by many section (section_name,section_id,time,capacity).Every teacher (t_id,t_name) has an account. Teacher can offer many course of a student.All of students(s_id,s_name) has a account. A students can select many sections(section_name, section_id,time,capacity). A course has a lots of sections.

ER DIAGRAM


Email
DOB
password
Gender
acc_name
admin_Id
Account
Has a
t_name
t_Id
teacher
can offer
course_Id
course_name
pre_course
course_sec
course
has
Section
time
capacity
section_name
section_Id
Has a
can select
Has a
can select
Student
s_name
s_Id
can create
admin
can create
a_name
a_Id

NORMALIZATION

Has a (Stu~~←Acc~~) (s_id ,sname, acc_id , acc_name, gender, password, DOB,email)

1NF

There is no multivalued attribute.

2NF

s_id , sname

acc_id , acc_name, gender, password, DOB,email, s_id

3NF

There is no transative dependency

s_id (PK),sname

acc_id (PK), acc_name, gender, password, DOB,email, s_id(FK).

TABLE

1. s_id(PK),sname
2. acc_id(PK), acc_name, gender, password, DOB,email, s_id(FK).

Has a (Tea~~←Acc~~) **(**t_id , t_name, acc_id , acc_name, gender, password, DOB,email)

1NF

There is no multivalued attribute.

2NF

t_id , t_name

acc_id , acc_name, gender, password, DOB,email , t_id

3NF

There is no transative dependency

t_id (PK),t_name

acc_id (PK),acc_name, gender, password, DOB,email , t_id(FK).

TABLE

1. t_id(PK),t_name
2. acc_id(PK),acc_name, gender, password, DOB,email , t_id(FK).

Has a (Ad ~~Acc~~) (a_id , a_name, acc_id , acc_name, gender, password, DOB,email)

1NF

There is no multivalued attribute.

2NF

a_id , a_name

acc_id , acc_name, gender, password, DOB,email , a_id

3NF

There is no transative dependency

a_id(PK), a_name

acc_id (PK),acc_name, gender, password, DOB,email , a_id(FK)

TABLE

1. a_id(PK), a_name
2. acc_id (PK),acc_name, gender, password, DOB,email , a_id(FK)

Can Select (s_id , s_name, course_id, course_name, course_sec, pre_course)

1NF

There is no multivalued attribute.

2NF

s_id , s_name

course_id, course_name, course_sec, pre_course, s_id

3NF

There is no transative dependency

s_id (PK), s_name

course_id(PK),course_name, course_sec, pre_course, s_id(FK).

TABLE

1. s_id (PK), s_name
2. course_id(PK),course_name, course_sec, pre_course, s_id(FK).

Can Offer (t_id , t_name , course_id, course_name, course_sec, pre_course)

1NF

There is no multivalued attribute.

2NF

t_id , t_name

course_id, course_name, course_sec, pre_course ,t_id

3NF

There is no transative dependency

t_id(PK),t_name

course_id(PK),course_name, course_sec, pre_course ,t_id(FK).

TABLE

1.t_id(PK),t_name

2.course_id(PK),course_name, course_sec, pre_course ,t_id(FK).

Has (course_id, course_name, course_sec, pre_course, section_id , section_name, time, capacity)

1NF

There is no multivalued attribute.

2NF

course_id, course_name, course_sec, pre_course

section_id , section_name, time, capacity, course_id

3NF

There is no transative dependency

course_id(PK),course_name, course_sec, pre_course(FK).

section_id(PK), section_name, time, capacity, course_id(FK).

TABLE

1. course_id(PK),course_name, course_sec, pre_course(FK).
2. section_id(PK), section_name, time, capacity, course_id(FK).

Can Select (s_id , s_name, section_id , section_name, time, capacity)

1NF

There is no multivalued attribute.

2NF

s_id , s_name

section_id , section_name, time, capacity, s_id

3NF

There is no transative dependency

s_id(PK), s_name

section_id(PK), section_name, time, capacity, s_id(FK).

TABLE

1.s_id (PK),s_name

2.section_id (PK), section_name, time, capacity, s_id(FK).

Can Create (a_id , a_name, course_id, course_name, course_sec, pre_course)

1NF

There is no multivalued attribute.

2NF

a_id , a_name

course_id, course_name, course_sec, pre_course, a_id

3NF

There is no transative dependency

a_id (PK),a_name

course_id(PK),course_name, course_sec, pre_course, a_id(FK).

TABLE

1.a_id (PK), a_name

2.course_id(PK), course_name, course_sec, pre_course, a_id(FK).

Can Create (a_id , a_name , section_id , section_name, time, capacity)

1NF

There is no multivalued attribute.

2NF

a_id , a_name

section_id , section_name, time, capacity, a_id

3NF

There is no transative dependency

a_id(PK),a_name

section_id (PK),section_name, time, capacity, a_id(FK).

TABLE

1.a_id(PK),a_name

2.section_(PK), section_name, time, capacity, a_id(FK).

ALL TABLE

1. s_id(PK),sname.
2. acc_id(PK),acc_name, gender, password, DOB,email, s_id(FK).
3. t_id(PK), t_name.

4. acc_id(PK),acc_name, gender, password, DOB,email , t_id(FK).
5. a_id (PK),a_name.
6. acc_id(PK), acc_name, gender, password, DOB,email , a_id(FK).
7. s_id(PK), s_name.
8. course_id(PK),course_name, course_sec, pre_course, s_id(FK).
9. t_id(PK),t_name.
10. course_id(PK),course_name, course_sec, pre_course ,t_id(FK).
11. course_id(PK), course_name, course_sec, pre_course.
12. section_id (PK),section_name, time, capacity, course_id(FK).
13. s_id (PK), s_name
14. section_id (PK),section_name, time, capacity, s_id(FK).
15. a_id (PK),s_name.
16. course_id(PK),course_name, course_sec, pre_course, a_id(FK).
17. a_id(PK),a_name
18. section_id (PK),section_name, time, capacity, a_id(FK).

FINAL TABLE

1.STUDENT :s_id(PK) , sname

2. STUDENT _ ACCOUNT: acc_id(PK), acc_name, gender, password, DOB,email, s_id(FK)

3.TEACHER : t_id(PK) , t_name

4.TEACHER _ACCOUNT: acc_id(PK) ,acc_name,gender,password,DOB,email , t_id(FK)

5.ADMIN: a_id(PK) , a_name

6.ADMIN_ ACCOUNT:acc_id (PK), acc_name, gender, password, DOB,email,a_id(FK)

7.STUDENT_COURSE:course_id(PK), course_name,course_sec, pre_course,s_id(FK)

8.TEACHER_COURSE:course_id(PK), course_name, course_sec, pre_course,t_id(FK)

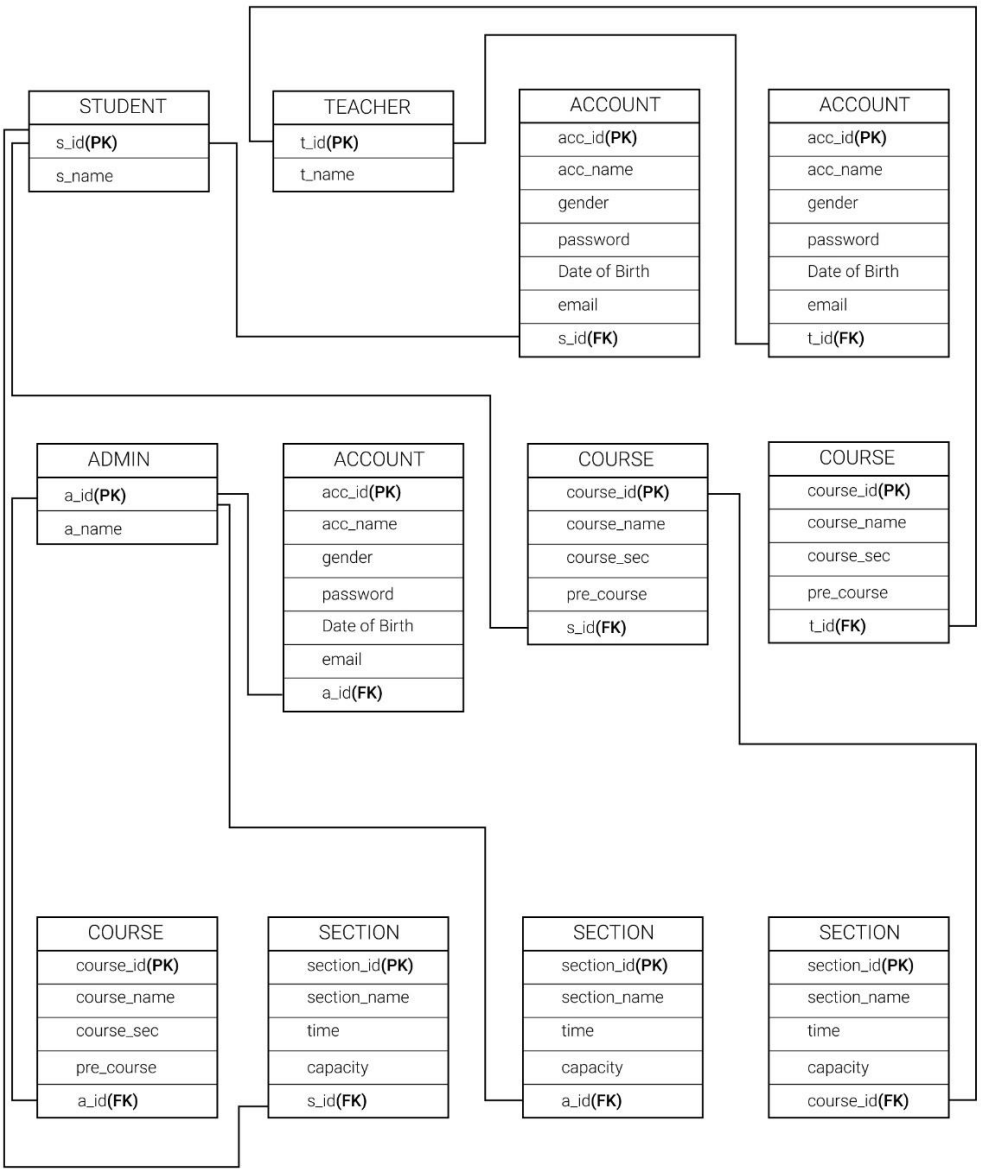
9.COURSE_SECTION:section_id(PK) , section_name, time, capacity, course_id(FK)

10.STUDENT_SECTION:section_id(PK) , section_name, time, capacity, s_id(FK)

11.ADMIN_COURSE:course_id(PK),course_name,course_sec,pre_course,a_id(FK)

12._ADMIN_SECTION: section_id(PK) , section_name, time, capacity, a_id(FK)

Schema Diagram



USER CREATE

CREATE USER PROJECT IDENTIFIED BY Password;


```
User created.

0.02 seconds
```

GRANT PRIVILEGES

GRANT CREATE TABLE, CREATE SEQUENCE, CREATE VIEW TO PROJECT;

```
Statement processed.

0.02 seconds
```

GRANT UNLIMITED TABLESPACE TO PROJECT;

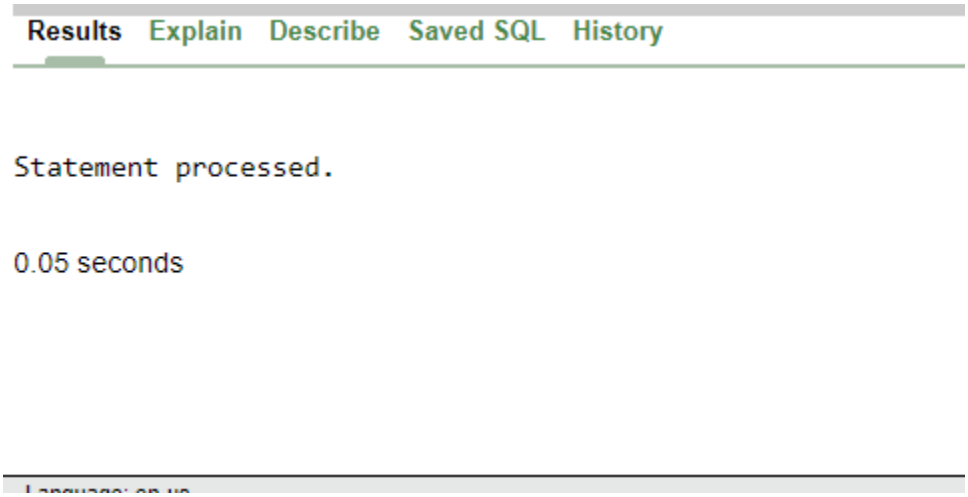


TABLE CREATE

```
CREATE TABLE students (
  s_id NUMBER PRIMARY KEY,
  sname VARCHAR2(50) NOT NULL
);
INSERT INTO students (s_id, sname) VALUES (1, 'John');
INSERT INTO students (s_id, sname) VALUES (2, 'Jane');
INSERT INTO students (s_id, sname) VALUES (3, 'Adam');
INSERT INTO students (s_id, sname) VALUES (4, 'Emily');
INSERT INTO students (s_id, sname) VALUES (5, 'David');



CREATE TABLE student_account (
  acc_id NUMBER PRIMARY KEY,
  acc_name VARCHAR2(50) NOT NULL,
  gender VARCHAR2(10) NOT NULL,
  password VARCHAR2(50) NOT NULL,
  DOB DATE NOT NULL,
  email VARCHAR2(50) NOT NULL,
  s_id NUMBER,
  CONSTRAINT fk_student_account_students
    FOREIGN KEY (s_id)
    REFERENCES students(s_id)
);
INSERT INTO student_account (acc_id, acc_name, gender, password, DOB, email, s_id)
```

```
VALUES (1, 'Alice', 'Female', 'password1', TO_DATE('1990-01-01', 'YYYY-MM-DD'), 'alice@example.com', 1);

INSERT INTO student_account (acc_id, acc_name, gender, password, DOB, email, s_id)
VALUES (2, 'Bob', 'Male', 'password2', TO_DATE('1992-05-15', 'YYYY-MM-DD'), 'bob@example.com', 2);

INSERT INTO student_account (acc_id, acc_name, gender, password, DOB, email, s_id)
VALUES (3, 'Charlie', 'Male', 'password3', TO_DATE('1995-12-31', 'YYYY-MM-DD'), 'charlie@example.com', 3);

INSERT INTO student_account (acc_id, acc_name, gender, password, DOB, email, s_id)
VALUES (4, 'Danielle', 'Female', 'password4', TO_DATE('1998-08-08', 'YYYY-MM-DD'), 'danielle@example.com', 4);

INSERT INTO student_account (acc_id, acc_name, gender, password, DOB, email, s_id)
VALUES (5, 'Edward', 'Male', 'password5', TO_DATE('2000-03-25', 'YYYY-MM-DD'), 'edward@example.com', 5);

CREATE TABLE teacher (
  t_id NUMBER PRIMARY KEY,
  t_name VARCHAR2(50) NOT NULL
);

INSERT INTO teacher (t_id, t_name)
VALUES (1, 'John Smith');

INSERT INTO teacher (t_id, t_name)
VALUES (2, 'Jane Doe');

INSERT INTO teacher (t_id, t_name)
```

```
VALUES (3, 'Michael Brown');

INSERT INTO teacher (t_id, t_name)
VALUES (4, 'Emily Green');

INSERT INTO teacher (t_id, t_name)
VALUES (5, 'David Lee');

CREATE TABLE teacher_account (
  acc_id NUMBER PRIMARY KEY,
  acc_name VARCHAR2(50) NOT NULL,
  gender VARCHAR2(10) NOT NULL,
  password VARCHAR2(50) NOT NULL,
  dob DATE NOT NULL,
  email VARCHAR2(100) NOT NULL,
  t_id NUMBER NOT NULL,
  CONSTRAINT fk_teacher_account_teacher FOREIGN KEY (t_id) REFERENCES teacher(t_id) );

5. CREATE TABLE ADMIN(
A_ID number(12) primary key,
A_NAME varchar2(20) not null
);
6.CREATE TABLE ACCNP(
ACC_ID number(15) not null,
ACC_NAME varchar2(20) not null,
GENDER VARCHAR2(15) not null,
PASSWORD VARCHAR2(40) not null,
DOB DATE,
EMAIL VARCHAR2(35),
```

```
A_ID NUMBER(15),
PRIMARY KEY (ACC_ID),
FOREIGN KEY (A_ID) REFERENCES ADMIN(A_ID)
);

7.CREATE TABLE COURSE(
COURSE_ID number(15) not null,
COURSE_NAME varchar2(20) not null,
PRE_COURSE varchar2(20) not null,
S_ID NUMBER(15),
PRIMARY KEY (COURSE_ID),
FOREIGN KEY (S_ID) REFERENCES STUDENT(S_ID)
);

8.CREATE TABLE CRSE(
COURSE_ID number(15) not null,
COURSE_NAME varchar2(20) not null,
COURSE_SEC varchar2(20) not null,
PRE_COURSE varchar2(20) not null,
T_ID NUMBER(15),
PRIMARY KEY (COURSE_ID),
FOREIGN KEY (T_ID) REFERENCES TEACHER(T_ID)
);

11.CREATE TABLE COSEE(
COURSE_ID number(15) not null,
COURSE_NAME varchar2(20) not null,
COURSE_SEC varchar2(20) not null,
PRE_COURSE varchar2(20) not null,
```

```
A_ID NUMBER(15),
PRIMARY KEY (COURSE_ID),
FOREIGN KEY (A_ID) REFERENCES ADMIN(A_ID)
);
9.CREATE TABLE SECTION(
SECTION_ID number(15) not null,
SECTION_NAME varchar2(20) not null,
TIME DATE,
CAPACITY number(20) not null,
COURSE_ID NUMBER(15),
PRIMARY KEY (SECTION_ID),
FOREIGN KEY (COURSE_ID) REFERENCES COURSE(COURSE_ID)
);

10.CREATE TABLE SEC(
SECTION_ID number(15) not null,
SECTION_NAME varchar2(20) not null,
TIME DATE,
CAPACITY number(20) not null,
S_ID NUMBER(15),
PRIMARY KEY (SECTION_ID),
FOREIGN KEY (S_ID) REFERENCES STUDENT(S_ID)
);

12.CREATE TABLE STION(
SECTION_ID number(15) not null,
SECTION_NAME varchar2(20) not null,
TIME DATE,
CAPACITY number(20) not null,
```

A_ID NUMBER(15),

PRIMARY KEY (SECTION_ID),

FOREIGN KEY (A_ID) REFERENCES ADMIN(A_ID)

);

Data insertion :

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **STUDENTS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
STUDENTS	S_ID	Number	-	-	-	1	-	-	-
	SNAME	Varchar2	50	-	-	-	-	-	-

1 - 2

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **TEACHER**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
TEACHER	T_ID	Number	-	-	-	1	-	-	-
	T_NAME	Varchar2	50	-	-	-	-	-	-

1 - 2

.

ORACLE Database Express Edition

Home Logout Help

User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
  course_id INT PRIMARY KEY,
  course_name VARCHAR(50),
  department_id INT,
  instructor_name VARCHAR(50),
  CONSTRAINT instructor_name_constraint CHECK (instructor_name LIKE '%Asian%')
);

-- insert values into the Courses table
INSERT INTO Courses (course_id, course_name, department_id, instructor_name)
VALUES (1, 'Introduction to Computer Science', 1, 'John Kim (Asian Studies)'),
       (2, 'English Literature', 2, 'Jane Lee (Asian American)'),
       (3, 'Asian History', 3, 'Tom Nguyen (East Asian)'),
       (4, 'Biology of Asian Plants', 4, 'Sarah Chang (Asian Ecology)'),
       (5, 'Chemistry of Asian Foods', 5, 'Alex Wang (Asian Cuisine)');

select * from courses
```

Results Explain Describe Saved SQL History

COURSE_ID	COURSE_NAME	DEPARTMENT_ID	INSTRUCTOR_NAME
1	Introduction to Computer Science	1	John Kim (Asian Studies)
2	English Literature	2	Jane Lee (Asian American)
3	Asian History	3	Tom Nguyen (East Asian)
4	Biology of Asian Plants	4	Sarah Chang (Asian Ecology)
5	Chemistry of Asian Foods	5	Alex Wang (Asian Cuisine)

5 rows returned in 0.00 seconds CSV Export

Autocommit Display 10 Save Run

```
INSERT INTO Departments (department_id, department_name)
VALUES (5, 'Psychology');
select * from departments
```

Results Explain Describe Saved SQL History

DEPARTMENT_ID	DEPARTMENT_NAME
1	Computer Science
2	Engineering
3	Mathematics
4	Business
5	Psychology

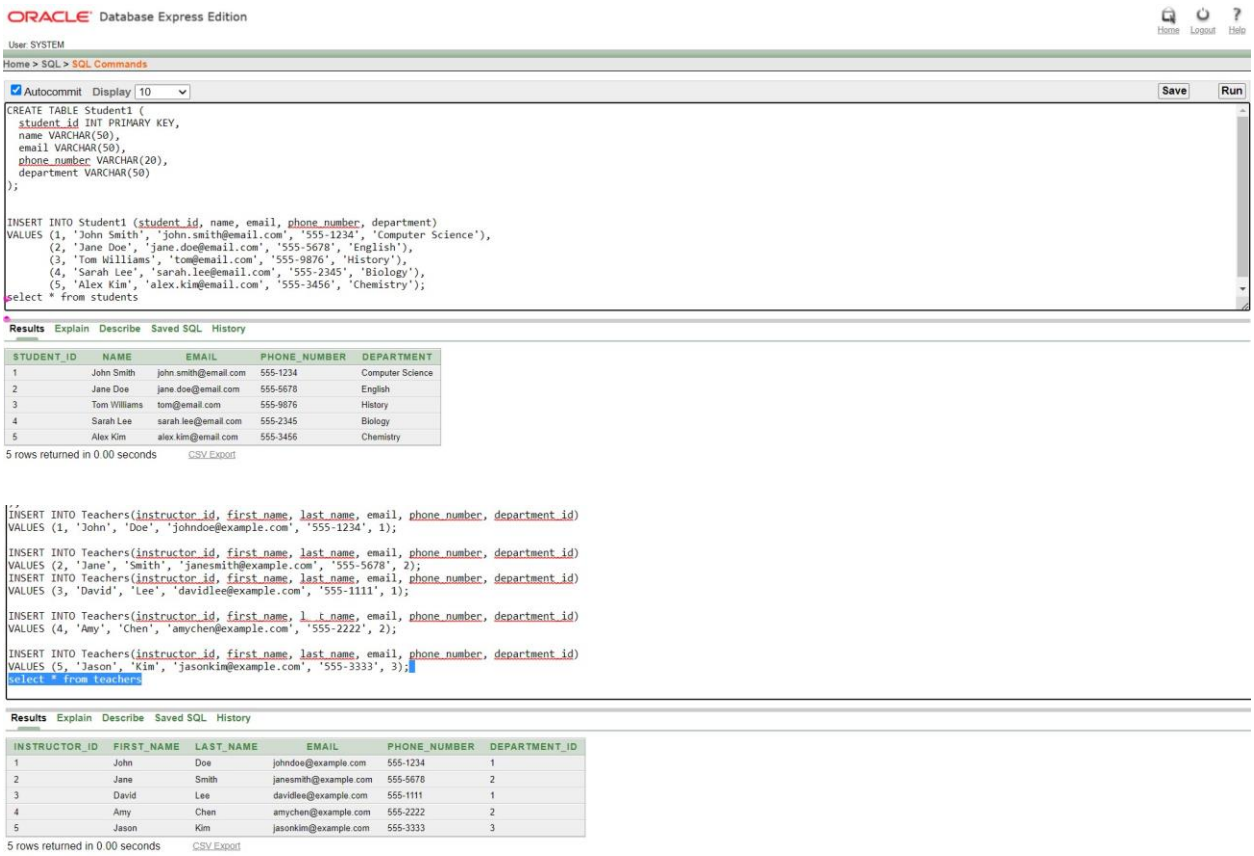
5 rows returned in 0.00 seconds CSV Export

```
select * from registration
```

Results Explain Describe Saved SQL History

REGISTRATION_ID	STUDENT_ID	COURSE_ID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

5 rows returned in 0.00 seconds CSV Export



Results Explain Describe Saved SQL History

ACC_ID	ACC_NAME	GENDER	PASSWORD	DOB	EMAIL	S_ID
1	Alice	Female	password1	01-JAN-90	alice@example.com	1
2	Bob	Male	password2	15-MAY-92	bob@example.com	2
3	Charlie	Male	password3	31-DEC-95	charlie@example.com	3
4	Danielle	Female	password4	08-AUG-98	danielle@example.com	4
5	Edward	Male	password5	25-MAR-00	edward@example.com	5

5 rows returned in 0.01 seconds CSV Export

Results Explain Describe Saved SQL History

T_ID	T_NAME
1	John Smith
2	Jane Doe
3	Michael Brown
4	Emily Green
5	David Lee

5 rows returned in 0.00 seconds CSV Export

Results Explain Describe Saved SQL History

S_ID	SNAME
1	John
2	Jane
3	Adam
4	Emily
5	David

5 rows returned in 0.00 seconds CSV Export

SINGLE ROW FUNCTION

1.What is the list of all student names in uppercase letters?

Ans:select upper(sname) from student;

2.What is the list of all teacher names in lowercase letters?

Ans:select lower(t_name) from teacher;

3.What is the password prefix for all the student accounts in the student_account table?

Ans: SELECT SUBSTR(password, 1, 3) AS password_prefix

FROM student_account;

Results Explain Describe Saved SQL History

LOWER(T_NAME)
john smith
jane doe
michael brown
emily green
david lee

5 rows returned in 0.00 seconds CSV Export

Results Explain Describe Saved SQL History

UPPER(SNAME)
JOHN
JANE
ADAM
EMILY
DAVID
RAFI

6 rows returned in 0.00 seconds CSV Export

Results Explain Describe Saved SQL History

PASSWORD_PREFIX
pas
pas
pas
pas
pas

5 rows returned in 0.00 seconds CSV Export

GROUP FUNCTION

1.What is the total number of records in the table "student_account"?

Ans:SELECT COUNT(*) FROM student_account;

2.What is the average length of passwords for all teacher accounts in the table "student_account"?

Ans:SELECT AVG(LENGTH(password)) AS avg_password_length

FROM student_account;

3.What is the number of teacher accounts grouped by gender in the student_account table?

Ans:SELECT gender, COUNT(*) AS num_accounts

FROM student_account

GROUP BY gender;

Results Explain Describe Saved SQL History

AVG_PASSWORD_LENGTH
9

1 rows returned in 0.00 seconds CSV Export

Results Explain Describe Saved SQL History

COUNT(*)
5

1 rows returned in 0.00 seconds CSV Export

Results Explain Describe Saved SQL History

GENDER	NUM_ACCOUNTS
Male	3
Female	2

2 rows returned in 0.00 seconds CSV Export

SUBQUERRY

1.Get the email addresses of students who have accounts?

Ans:SELECT email

FROM student_account

WHERE s_id IN (

  SELECT s_id

  FROM students

)

2.Get the average length of passwords for female students

Ans:SELECT AVG(LENGTH(password))

FROM student_account

WHERE gender = 'Female' AND s_id IN (

  SELECT s_id

  FROM students

)

3.Get the number of male students who have accounts?

Ans:SELECT COUNT(DISTINCT s_id)

FROM student_account

WHERE gender = 'Male' AND s_id IN (

  SELECT s_id

  FROM students

)

Results Explain Describe Saved SQL History

EMAIL
alice@example.com
bob@example.com
charlie@example.com
danielle@example.com
edward@example.com

5 rows returned in 0.00 seconds CSV Export

Results Explain Describe Saved SQL History

COUNT(DISTINCTS_ID)
3

1 rows returned in 0.00 seconds CSV Export

Results Explain Describe Saved SQL History

AVG(LENGTH(PASSWORD))
9

1 rows returned in 0.00 seconds CSV Export

JOINING

1.Write a query to retrieve the account information of all female students along with their names?

Ans:SELECT sa.acc_id, sa.acc_name, sa.gender, sa.password, sa.DOB, sa.email, s.sname

FROM student_account sa

JOIN students s ON sa.s_id = s.s_id

WHERE sa.gender = 'Female';

2.Write a query to retrieve the account information of all male students who were born after 1990?

Ans:SELECT sa.acc_id, sa.acc_name, sa.gender, sa.password, sa.DOB, sa.email, s.sname

FROM student_account sa

JOIN students s ON sa.s_id = s.s_id

WHERE sa.gender = 'Male' AND sa.DOB > TO_DATE('1990-01-01', 'YYYY-MM-DD');

3.Can you provide a list of all student accounts, along with their respective student IDs and names?

Ans:SELECT sa.acc_id, sa.acc_name, sa.s_id, s.sname

FROM student_account sa

JOIN students s ON sa.s_id = s.s_id;

Results Explain Describe Saved SQL History

ACC_ID	ACC_NAME	GENDER	PASSWORD	DOB	EMAIL	SNAME
2	Bob	Male	password2	15-MAY-92	bob@example.com	Jane
3	Charlie	Male	password3	31-DEC-95	charlie@example.com	Adam
5	Edward	Male	password5	25-MAR-00	edward@example.com	David

3 rows returned in 0.00 seconds CSV Export

Results Explain Describe Saved SQL History

ACC_ID	ACC_NAME	GENDER	PASSWORD	DOB	EMAIL	SNAME
1	Alice	Female	password1	01-JAN-90	alice@example.com	John
4	Danielle	Female	password4	08-AUG-98	danielle@example.com	Emily

2 rows returned in 0.00 seconds CSV Export

Results Explain Describe Saved SQL History

ACC_ID	ACC_NAME	S_ID	SNAME
1	Alice	1	John
2	Bob	2	Jane
3	Charlie	3	Adam
4	Danielle	4	Emily
5	Edward	5	David

5 rows returned in 0.00 seconds CSV Export

VIEW

1.Create a view that shows the account ID, name, and email of all female students?

Ans:CREATE VIEW female_student_accounts AS

SELECT acc_id, acc_name, email

FROM student_account

WHERE gender = 'Female';

2.Create a view that shows the account ID, name, and date of birth of all students born after January 1st, 1995?

Ans:CREATE VIEW young_student_accounts AS

SELECT acc_id, acc_name, DOB

FROM student_account

WHERE DOB > TO_DATE('1995-01-01', 'YYYY-MM-DD');

3.Create a view that shows the account ID, name, email, and name of the student's teacher for all students?

Ans:CREATE VIEW student_teacher_accounts AS

SELECT sa.acc_id, sa.acc_name, sa.email, s.sname, t.tname

FROM student_account sa

JOIN students s ON sa.s_id = s.s_id

JOIN teacher_students ts ON s.s_id = ts.s_id

JOIN teacher t ON ts.t_id = t.t_id;

Results Explain Describe Saved SQL History

```
View created.
```

0.05 seconds

Results Explain Describe Saved SQL History

```
View created.
```

0.05 seconds

Results Explain Describe Saved SQL History

```
View created.
```

0.05 seconds

Synonym

1.Create a synonym for the table?

CREATE SYNONYM sa FOR student_account;

3. Create a synonym for view called StudentInfo

PL/SQL:

Function:

1.To get Student Course count :

```
CREATE OR REPLACE FUNCTION get_department_course_count(department_id IN NUMBER) RETURN
NUMBER AS
  course_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO course_count
  FROM courses
  WHERE department_id = department_id;

  RETURN course_count;
END;
```

2.To get Course count :

```
CREATE OR REPLACE FUNCTION get_department_course_count(department_id IN NUMBER) RETURN
NUMBER AS
  course_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO course_count
  FROM courses
  WHERE department_id = department_id;

  RETURN course_count;
END;
```

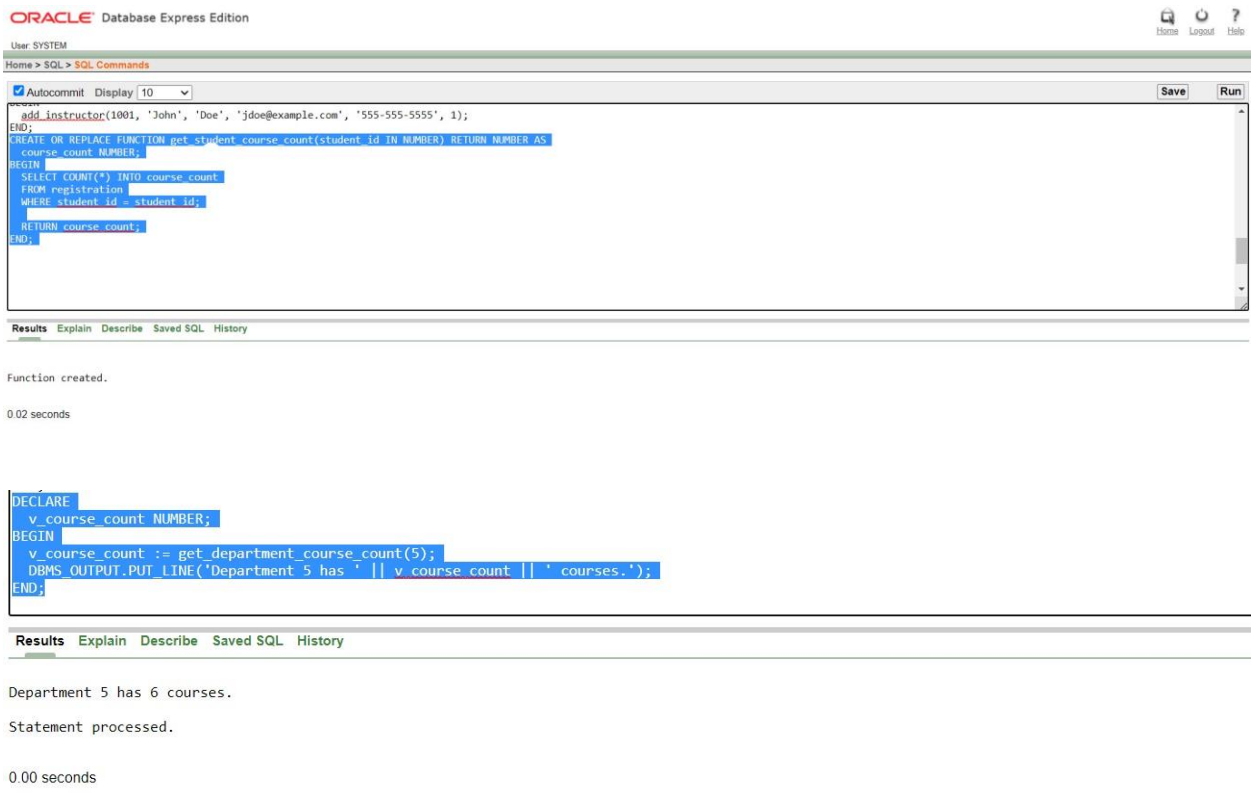
3.To get Course Teacher:

```
CREATE OR REPLACE FUNCTION get_course_instructor(course_id IN NUMBER) RETURN VARCHAR2 AS
  instructor_name VARCHAR2(100);
BEGIN
  SELECT first_name || ' ' || last_name INTO instructor_name
```

```
  FROM instructor
  WHERE instructor_id = (
    SELECT instructor_id
    FROM courses
    WHERE course_id = course_id
  );

  RETURN instructor_name;
END;
```

Screenshots:



Procedure:

1.Add students:

```
CREATE OR REPLACE PROCEDURE add_student (
  p_student_id IN NUMBER,
  p_name IN VARCHAR2,
  p_email IN VARCHAR2,
  p_phone_number IN VARCHAR2,
  p_department IN VARCHAR2
)
IS
BEGIN
  INSERT INTO Student1 (student_id, name,email, phone_number,department)
  VALUES (p_student_id, p_name,p_email, p_phone_number,p_department);
```

```
  COMMIT;

  DBMS_OUTPUT.PUT_LINE('New student added successfully.');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error adding student: ' || SQLERRM);
END;
BEGIN
  add_student(12345, 'Azraf','johndoe@example.com', '555-1234', 'Computer Science');
END;
```

2.Add Course:

```
CREATE OR REPLACE PROCEDURE add_course (
  p_course_id IN NUMBER,
  p_course_name IN VARCHAR2,
  p_department_id IN NUMBER,
  p_instructor_name IN VARCHAR2,
  p_credits IN NUMBER
)
IS
BEGIN
  INSERT INTO Courses (course_id, course_name, department_id, instructor_name, credits)
  VALUES (p_course_id, p_course_name, p_department_id, p_instructor_name, p_credits);

  COMMIT;

  DBMS_OUTPUT.PUT_LINE('New course added successfully.');
EXCEPTION
  WHEN OTHERS THEN
```

```
    DBMS_OUTPUT.PUT_LINE('Error adding course: ' || SQLERRM);
END;
BEGIN
  add_course(101, 'Introduction to Computer Science', 1, '', 3);
END;
```

3.Add Teacher:

```
CREATE OR REPLACE PROCEDURE add_instructor (
  p_instructor_id IN INT,
  p_first_name IN VARCHAR2,
  p_last_name IN VARCHAR2,
  p_email IN VARCHAR2,
  p_phone_number IN VARCHAR2,
  p_department_id IN INT
)
IS
BEGIN
  INSERT INTO Teachers(instructor_id, first_name, last_name, email, phone_number, department_id)
  VALUES (p_instructor_id, p_first_name, p_last_name, p_email, p_phone_number, p_department_id);
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('New instructor added successfully.');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error adding instructor: ' || SQLERRM);
END;
BEGIN
  add_instructor(1001, 'John', 'Doe', 'jdoe@example.com', '555-555-5555', 1);
END;
```

4.Same Department Finding:

```
CREATE OR REPLACE PROCEDURE find_students_in_department(
  dept_name IN VARCHAR2
)
IS
BEGIN
  FOR rec IN (SELECT name FROM Student1 WHERE department = dept_name)
  LOOP
    DBMS_OUTPUT.PUT_LINE(rec.name);
  END LOOP;
END;
BEGIN
  find_students_in_department('Computer Science');
END;
```

Screenshot:

Record: Student record ,course record , department record

```
DECLARE
  TYPE student_rec IS RECORD (
    student_id NUMBER,
    first_name VARCHAR2(50),
```

```
  last_name VARCHAR2(50),
  email VARCHAR2(100),
  phone_number VARCHAR2(20),
  department_id NUMBER
 );
 v_student student_rec;
BEGIN
 v_student.student_id := 1234;
 v_student.first_name := 'John';
 v_student.last_name := 'Doe';
 v_student.email := 'johndoe@email.com';
 v_student.phone_number := '555-1234';
 v_student.department_id := 5;
 -- use the v_student record as needed
END;
```

Course record:

```
DECLARE
 TYPE course_rec IS RECORD (
  course_id NUMBER,
  course_name VARCHAR2(100),
  instructor_name VARCHAR2(100),
  start_date DATE,
  end_date DATE,
  department_id NUMBER
 );
 v_course course_rec;
BEGIN
 v_course.course_id := 101;
 v_course.course_name := 'Intro to Database Systems';
```

```
  v_course.instructor_name := 'Jane Smith';
  v_course.start_date := TO_DATE('2023-09-01', 'YYYY-MM-DD');
  v_course.end_date := TO_DATE('2023-12-15', 'YYYY-MM-DD');
  v_course.department_id := 5;
  -- use the v_course record as needed
END;
```

Department:

```
DECLARE
  TYPE department_rec IS RECORD (
    department_id NUMBER,
    department_name VARCHAR2(50),
    department_head VARCHAR2(100),
    budget NUMBER,
    location VARCHAR2(100)
  );
  v_department department_rec;
BEGIN
  v_department.department_id := 5;
  v_department.department_name := 'Computer Science';
  v_department.department_head := 'Alice Johnson';
  v_department.budget := 1000000;
  v_department.location := 'Building A, Room 101';
  -- use the v_department record as needed
END;
```

Cursor: To inccease course fees

```
DECLARE
  CURSOR payment_cur IS
    SELECT course_id, cost_per_credit, total_fees
```

```
    FROM student_payment;

  v_course_id student_payment.course_id%TYPE;
  v_cost_per_credit student_payment.cost_per_credit%TYPE;
  v_total_fees student_payment.total_fees%TYPE;
BEGIN
  FOR payment_rec IN payment_cur LOOP
    v_course_id := payment_rec.course_id;
    v_cost_per_credit := payment_rec.cost_per_credit * 1.1; -- increase fees per credit by 10%
    v_total_fees := payment_rec.cost_per_credit * payment_rec.total_credits;

    UPDATE student_payment
    SET cost_per_credit = v_cost_per_credit, total_fees = v_total_fees
    WHERE course_id = v_course_id;
  END LOOP;
  COMMIT;
END;
```

Trigger: Student added , Course Added, Teacher Added

```
CREATE OR REPLACE TRIGGER student_audit_trigger
AFTER INSERT ON student1

DECLARE
BEGIN
  INSERT INTO student_audit (student_id, action, audit_timestamp)
  VALUES (:NEW.student_id, 'INSERT', SYSTIMESTAMP);
END;
/
2.
CREATE OR REPLACE TRIGGER course_audit_trigger
```

```
AFTER INSERT ON courses
FOR EACH ROW
DECLARE
BEGIN
  INSERT INTO course_audit (course_id, action, audit_timestamp)
  VALUES (:NEW.course_id, 'INSERT', SYSTIMESTAMP);
END;
/
```

3.

```
CREATE OR REPLACE TRIGGER teacher_audit_trigger
AFTER INSERT ON instructor
FOR EACH ROW
DECLARE
BEGIN
  INSERT INTO teacher_audit (instructor_id, action, audit_timestamp)
  VALUES (:NEW.instructor_id, 'INSERT', SYSTIMESTAMP);
END;
/
```

Package: package to find students in same department

```
CREATE OR REPLACE PACKAGE BODY Find_Student AS
  PROCEDURE find_students_in_department(p_department IN VARCHAR2) AS
    CURSOR student_cursor IS SELECT name FROM Student1 WHERE department = p_department;
    student_name Students.name%TYPE;
  BEGIN
    OPEN student_cursor;
    LOOP
      FETCH student_cursor INTO student_name;
      EXIT WHEN student_cursor%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(student_name);
```

```
    END LOOP;
    CLOSE student_cursor;
  END find_students_in_department;
END Find_Student;
/
```

RELATIONAL ALZEBRA

1.Find the name of the STUDENT whose student_id is 3

ANS: P STUDENTS_NAME(s

STUDENTS_ID=3(STUDENT))

2.Find the email of account holder where acc_name is Bob.

ANS: P EMAIL(s ACC_NAME='Bob'(STUDENT_ACCOUNT))

3.Find out T_id,T_NAME.

ANS: P T_ID,T_NAME(TEACHER)

4.Find out Gender,Password,DOB.

ANS: P GOAL_ID,GOALNO,PLAYERS_ID(STUDENT_ACCOUNT)

5.Find out DOB is less than '1995-12-31' .

ANS: (s DOB<1995-12-31(STUDENT_ACCOUNT)).

Conclusion:

The course registration management system will provide educational institutions to managing course registration. It will simplify the registration process for students. This system will save time and effort.