Student Portal Management System

SUBMITTED BY:

Kamran Ali (21011519-026)

Shahan Ghani (21011519-122)

SUBMITTED TO:

Dr. Zahid Iqbal

COURSE TITLE:

Data Base Management System

COURSE CODE:

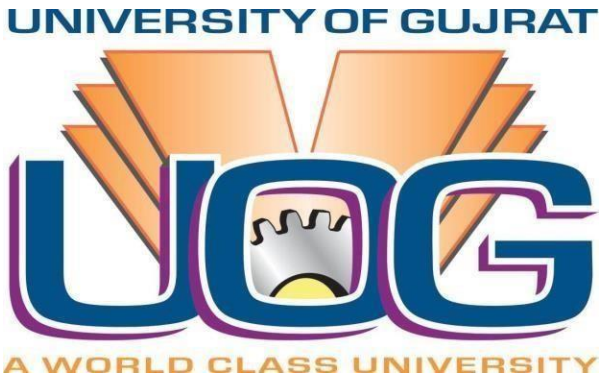
CS-241

DEPARTMENT:

DEPARTMENT OF COMPUTER SCIENCE

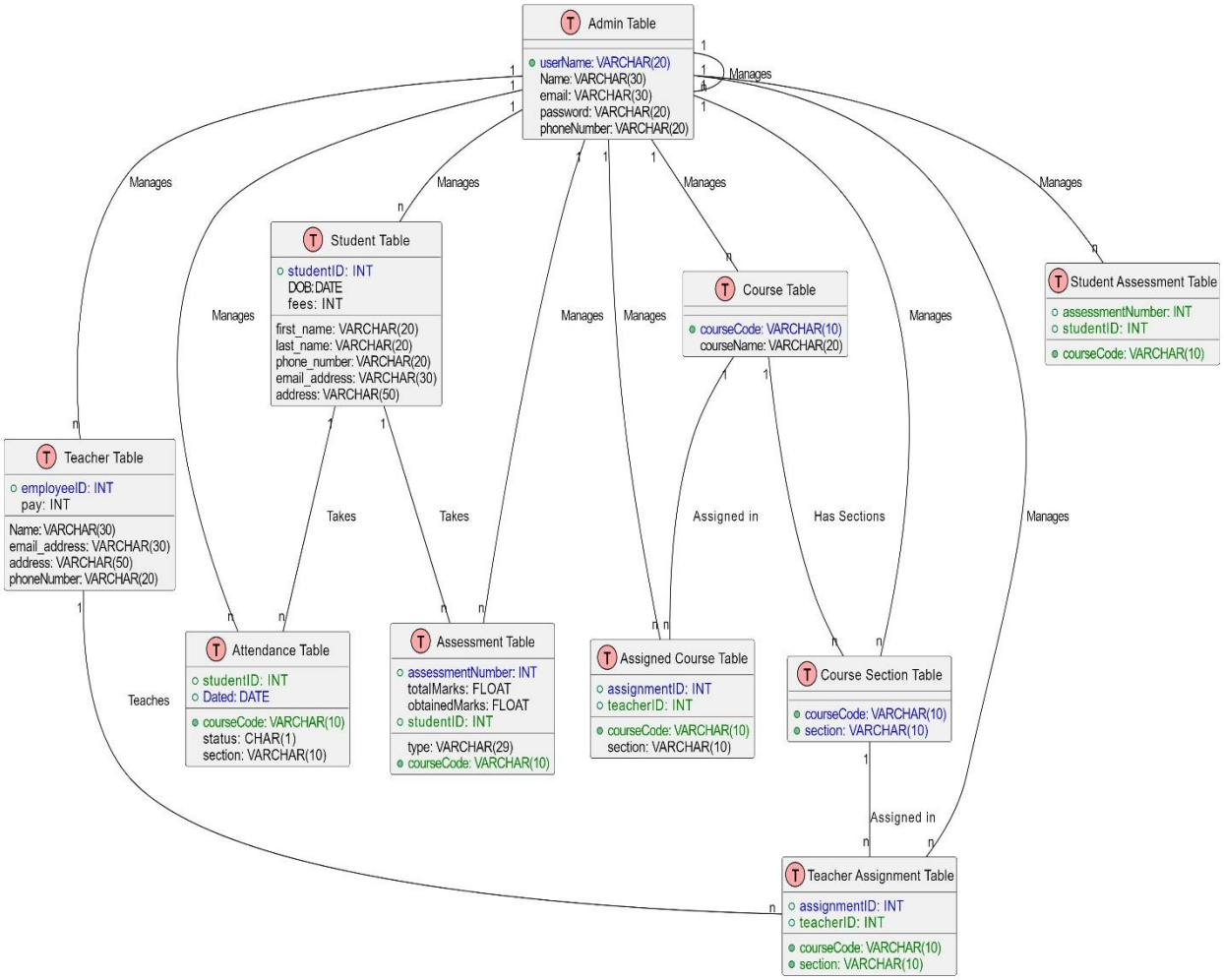
SECTION:

BSCS-C (4^{th})



ER DIAGRAM:

ER Diagram ER Diagram shows the basic concept and idea of a database. It tells us about the relations between the tables and the relation between the tables and their attributes. Following is the ER Diagram of our project



Tables Of this Project:

- Student
- Teacher
- Admin
- Course
- Assigned_Course
- Attendance
- Assessment
- Course_Section
- Teacher_Assignment
- Student_Assessment

Complete Code Project :

```
-- Create the database

-- CREATE DATABASE student_portal_management;

-- USE student_portal_management;


-- Creating the "Student" table

CREATE TABLE Student (

  studentID INT PRIMARY KEY,

  first_name VARCHAR(20),

  last_name VARCHAR(20),

  phone_number VARCHAR(20),

  email_address VARCHAR(30),

  DOB DATE,

  address VARCHAR(50),

  fees INT

);


-- Creating the "Teacher" table

CREATE TABLE Teacher (

  employeeID INT PRIMARY KEY,
```

```
  Name VARCHAR(30),
  email_address VARCHAR(30),
  address VARCHAR(50),
  phoneNumber VARCHAR(20),
  pay INT
);

-- Creating the "Admin" table
CREATE TABLE Admin (
  userName VARCHAR(20) PRIMARY KEY,
  Name VARCHAR(30),
  email VARCHAR(30),
  password VARCHAR(20),
  phoneNumber VARCHAR(20)
);

-- Creating the "Course" table
CREATE TABLE Course (
  courseCode VARCHAR(10) PRIMARY KEY,
  courseName VARCHAR(20)
);

-- Creating the "Assigned_Course" table
CREATE TABLE Assigned_Course (
  assignmentID INT PRIMARY KEY,
  teacherID INT,
  courseCode VARCHAR(10),
  section VARCHAR(10),
  FOREIGN KEY (teacherID) REFERENCES Teacher (employeeID),
```

```
  FOREIGN KEY (courseCode) REFERENCES Course (courseCode)
);

-- Creating the "Attendance" table
CREATE TABLE Attendance (
  studentID INT,
  courseCode VARCHAR(10),
  Dated DATE,
  status CHAR(1),
  section VARCHAR(10),
  FOREIGN KEY (studentID) REFERENCES Student (studentID),
  FOREIGN KEY (courseCode) REFERENCES Course (courseCode),
  PRIMARY KEY (studentID, courseCode, Dated)
);

-- Creating the "Assessment" table
CREATE TABLE Assessment (
  assessmentNumber INT PRIMARY KEY,
  type VARCHAR(29),
  totalMarks FLOAT,
  obtainedMarks FLOAT,
  studentID INT,
  courseCode VARCHAR(10),
  FOREIGN KEY (studentID) REFERENCES Student (studentID),
  FOREIGN KEY (courseCode) REFERENCES Course (courseCode)
);

-- Applying normalization concepts:
-- Creating a new table "Course_Section" to store course sections
```

```
CREATE TABLE Course_Section (
  courseCode VARCHAR(10),
  section VARCHAR(10),
  PRIMARY KEY (courseCode, section),
  FOREIGN KEY (courseCode) REFERENCES Course (courseCode)
);

-- Creating a new table "Teacher_Assignment" to store teacher-course assignments
CREATE TABLE Teacher_Assignment (
  assignmentID INT PRIMARY KEY,
  teacherID INT,
  courseCode VARCHAR(10),
  section VARCHAR(10), -- Adding section to match the referenced table
  FOREIGN KEY (teacherID) REFERENCES Teacher (employeeID),
  FOREIGN KEY (courseCode, section) REFERENCES Course_Section (courseCode, section) -- Matching the referenced columns
);

-- Creating a new table "Student_Assessment" to store student assessments
CREATE TABLE Student_Assessment (
  assessmentNumber INT,
  studentID INT,
  courseCode VARCHAR(10),
  FOREIGN KEY (assessmentNumber) REFERENCES Assessment (assessmentNumber),
  FOREIGN KEY (studentID) REFERENCES Student (studentID),
  FOREIGN KEY (courseCode) REFERENCES Course (courseCode)
);

-- Inserting sample values into "Course" table
```

```
INSERT INTO Course (courseCode, courseName)
VALUES
  ('C101', 'Mathematics'),
  ('C102', 'Science'),
  ('C103', 'History'),
  ('C104', 'English'),
  ('C105', 'Physics');

-- Inserting sample values into "Course_Section" table
INSERT INTO Course_Section (courseCode, section)
VALUES
  ('C101', 'A'),
  ('C102', 'B'),
  ('C103', 'A'),
  ('C104', 'A'),
  ('C105', 'B');

-- Inserting sample values into "Teacher" table
INSERT INTO Teacher (employeeID, Name, email_address, address, phoneNumber, pay)
VALUES
  (101, 'Michael Johnson', 'michael.johnson@example.com', '789 Elm St', '5551234567', 3500),
  (102, 'Emily Williams', 'emily.williams@example.com', '456 Pine Rd', '5559876543', 3800),
  (103, 'David Smith', 'david.smith@example.com', '123 Oak Ave', '5552223333', 3200),
  (104, 'Sarah Adams', 'sarah.adams@example.com', '789 Maple Rd', '5554445555', 3400),
  (105, 'James Brown', 'james.brown@example.com', '456 Elm St', '5556667777', 3700);

-- Inserting sample values into "Teacher_Assignment" table with existing values in "Course_Section"
table
INSERT INTO Teacher_Assignment (assignmentID, teacherID, courseCode, section)
```

```
VALUES
  (201, 101, 'C101', 'A'),
  (202, 101, 'C102', 'B'),
  (203, 102, 'C103', 'A'),
  (204, 103, 'C104', 'A'),
  (205, 104, 'C105', 'B');

-- Inserting sample values into "Student" table
INSERT INTO Student (studentID, first_name, last_name, phone_number, email_address, DOB, address, fees)
VALUES
  (1, 'John', 'Doe', '1234567890', 'john.doe@example.com', '2000-01-15', '123 Main St', 1000),
  (2, 'Jane', 'Smith', '9876543210', 'jane.smith@example.com', '2001-05-20', '456 Oak Ave', 1200),
  (3, 'Michael', 'Johnson', '5551234567', 'michael.johnson@example.com', '1999-10-30', '789 Elm St', 800),
  (4, 'Emily', 'Williams', '5559876543', 'emily.williams@example.com', '2002-03-25', '456 Pine Rd', 1100),
  (5, 'Robert', 'Lee', '5558887777', 'robert.lee@example.com', '2003-07-12', '789 Oak St', 900);

-- Inserting sample values into "Admin" table
INSERT INTO Admin (userName, Name, email, password, phoneNumber)
VALUES
  ('admin1', 'Admin One', 'admin1@example.com', 'adminpass1', '5551111111'),
  ('admin2', 'Admin Two', 'admin2@example.com', 'adminpass2', '5552222222'),
  ('admin3', 'Admin Three', 'admin3@example.com', 'adminpass3', '5553333333'),
  ('admin4', 'Admin Four', 'admin4@example.com', 'adminpass4', '5554444444'),
  ('admin5', 'Admin Five', 'admin5@example.com', 'adminpass5', '5555555555');

-- Inserting sample values into "Attendance" table
INSERT INTO Attendance (studentID, courseCode, Dated, status, section)
```

```
VALUES
  (1, 'C101', '2023-07-15', 'P', 'A'),
  (1, 'C102', '2023-07-15', 'A', 'B'),
  (2, 'C102', '2023-07-15', 'P', 'B'),
  (3, 'C103', '2023-07-15', 'P', 'A'),
  (4, 'C101', '2023-07-15', 'P', 'A');

-- Inserting sample values into "Assessment" table
INSERT INTO Assessment (assessmentNumber, type, totalMarks, obtainedMarks, studentID,
courseCode)
VALUES
  (301, 'Midterm', 100, 85, 1, 'C101'),
  (302, 'Final', 100, 90, 1, 'C102'),
  (303, 'Final', 100, 80, 2, 'C102'),
  (304, 'Midterm', 100, 70, 2, 'C103'),
  (305, 'Final', 100, 88, 3, 'C103');

-- Inserting sample values into "Student_Assessment" table
INSERT INTO Student_Assessment (assessmentNumber, studentID, courseCode)
VALUES
  (301, 1, 'C101'),
  (302, 1, 'C102'),
  (303, 2, 'C102'),
  (304, 2, 'C103'),
  (305, 3, 'C103');

  SELECT * FROM Student;
SELECT * FROM Teacher;
SELECT * FROM Admin;
```

```
SELECT * FROM Course;

SELECT * FROM Assigned_Course;

SELECT * FROM Attendance;

SELECT * FROM Assessment;

SELECT * FROM Course_Section;

SELECT * FROM Teacher_Assignment;

SELECT * FROM Student_Assessment;



--INNER JOIN:   An inner join returns only the rows that have matching values in both tables.

SELECT Student.studentID, first_name, last_name, Attendance.courseCode, courseName

FROM Student

INNER JOIN Attendance ON Student.studentID = Attendance.studentID

INNER JOIN Course ON Attendance.courseCode = Course.courseCode;



--LEFT JOIN (or LEFT OUTER JOIN):  A left join returns all rows from the left table and the matching rows from the right table. If there is no match, it returns NULL values for the right table columns.

--INNERLEFTsql


SELECT Student.studentID, first_name, last_name, Attendance.courseCode, courseName

FROM Student

LEFT JOIN Attendance ON Student.studentID = Attendance.studentID

LEFT JOIN Course ON Attendance.courseCode = Course.courseCode;

--INNERLEFTsqlRIGHT JOIN (or RIGHT OUTER JOIN):

--INNERLEFTsqlRIGHTA right join returns all rows from the right table and the matching rows from the left table. If there is no match, it returns NULL values for the left table columns.


SELECT Student.studentID, first_name, last_name, Attendance.courseCode, courseName
```

```
FROM Student

RIGHT JOIN Attendance ON Student.studentID = Attendance.studentID

LEFT JOIN Course ON Attendance.courseCode = Course.courseCode;

---FULL OUTER JOIN (or FULL JOIN):

---FULLA full outer join returns all rows when there is a match in either the left or the right table. If there is no match, it returns NULL values for the non-matching side.


SELECT COALESCE(Student.studentID, Attendance.studentID) AS studentID, first_name, last_name, Attendance.courseCode, courseName

FROM Student

FULL OUTER JOIN Attendance ON Student.studentID = Attendance.studentID

FULL OUTER JOIN Course ON Attendance.courseCode = Course.courseCode;

-- Stored Procedures for each table

GO

-- Stored Procedure for "Student" table

CREATE PROCEDURE GetStudentDetails(@studentID INT)

AS

BEGIN

  SELECT * FROM Student WHERE studentID = @studentID;

END;

GO

-- Stored Procedure for "Teacher" table

CREATE PROCEDURE GetTeacherDetails(@employeeID INT)

AS

BEGIN

  SELECT * FROM Teacher WHERE employeeID = @employeeID;

END;

GO

-- Stored Procedure for "Admin" table
```

```
CREATE PROCEDURE GetAdminDetails(@userName VARCHAR(20))
AS
BEGIN
  SELECT * FROM Admin WHERE userName = @userName;
END;
GO
-- Stored Procedure for "Course" table
CREATE PROCEDURE GetCourseDetails(@courseCode VARCHAR(10))
AS
BEGIN
  SELECT * FROM Course WHERE courseCode = @courseCode;
END;
GO
-- Stored Procedure for "Assigned_Course" table
CREATE PROCEDURE GetAssignedCourseDetails(@assignmentID INT)
AS
BEGIN
  SELECT * FROM Assigned_Course WHERE assignmentID = @assignmentID;
END;
GO
-- Stored Procedure for "Attendance" table
CREATE PROCEDURE GetAttendanceDetails(@studentID INT, @courseCode VARCHAR(10), @Dated
DATE)
AS
BEGIN
  SELECT * FROM Attendance WHERE studentID = @studentID AND courseCode = @courseCode AND
Dated = @Dated;
END;
GO
```

```
-- Stored Procedure for "Assessment" table
CREATE PROCEDURE GetAssessmentDetails(@assessmentNumber INT)
AS
BEGIN
  SELECT * FROM Assessment WHERE assessmentNumber = @assessmentNumber;
END;
GO
-- Stored Procedure for "Course_Section" table
CREATE PROCEDURE GetCourseSectionDetails(@courseCode VARCHAR(10), @section VARCHAR(10))
AS
BEGIN
  SELECT * FROM Course_Section WHERE courseCode = @courseCode AND section = @section;
END;
GO
-- Stored Procedure for "Teacher_Assignment" table
CREATE PROCEDURE GetTeacherAssignmentDetails(@assignmentID INT)
AS
BEGIN
  SELECT * FROM Teacher_Assignment WHERE assignmentID = @assignmentID;
END;
GO
-- Stored Procedure for "Student_Assessment" table
CREATE PROCEDURE GetStudentAssessmentDetails(@assessmentNumber INT, @studentID INT,
@courseCode VARCHAR(10))
AS
BEGIN
  SELECT * FROM Student_Assessment WHERE assessmentNumber = @assessmentNumber AND
studentID = @studentID AND courseCode = @courseCode;
END;
```

```
GO

-- to call Procedures

EXEC GetStudentDetails @studentID = 1;

EXEC GetTeacherDetails @employeeID = 101;

EXEC GetAdminDetails @userName = 'admin1';

EXEC GetCourseDetails @courseCode = 'C101';

EXEC GetAssignedCourseDetails @assignmentID = 201;

EXEC GetAttendanceDetails @studentID = 1, @courseCode = 'C101', @Dated = '2023-07-15';

EXEC GetAssessmentDetails @assessmentNumber = 301;

EXEC GetCourseSectionDetails @courseCode = 'C101', @section = 'A';

EXEC GetTeacherAssignmentDetails @assignmentID = 201;

EXEC GetStudentAssessmentDetails @assessmentNumber = 301, @studentID = 1, @courseCode =
'C101';
```