Cover Page

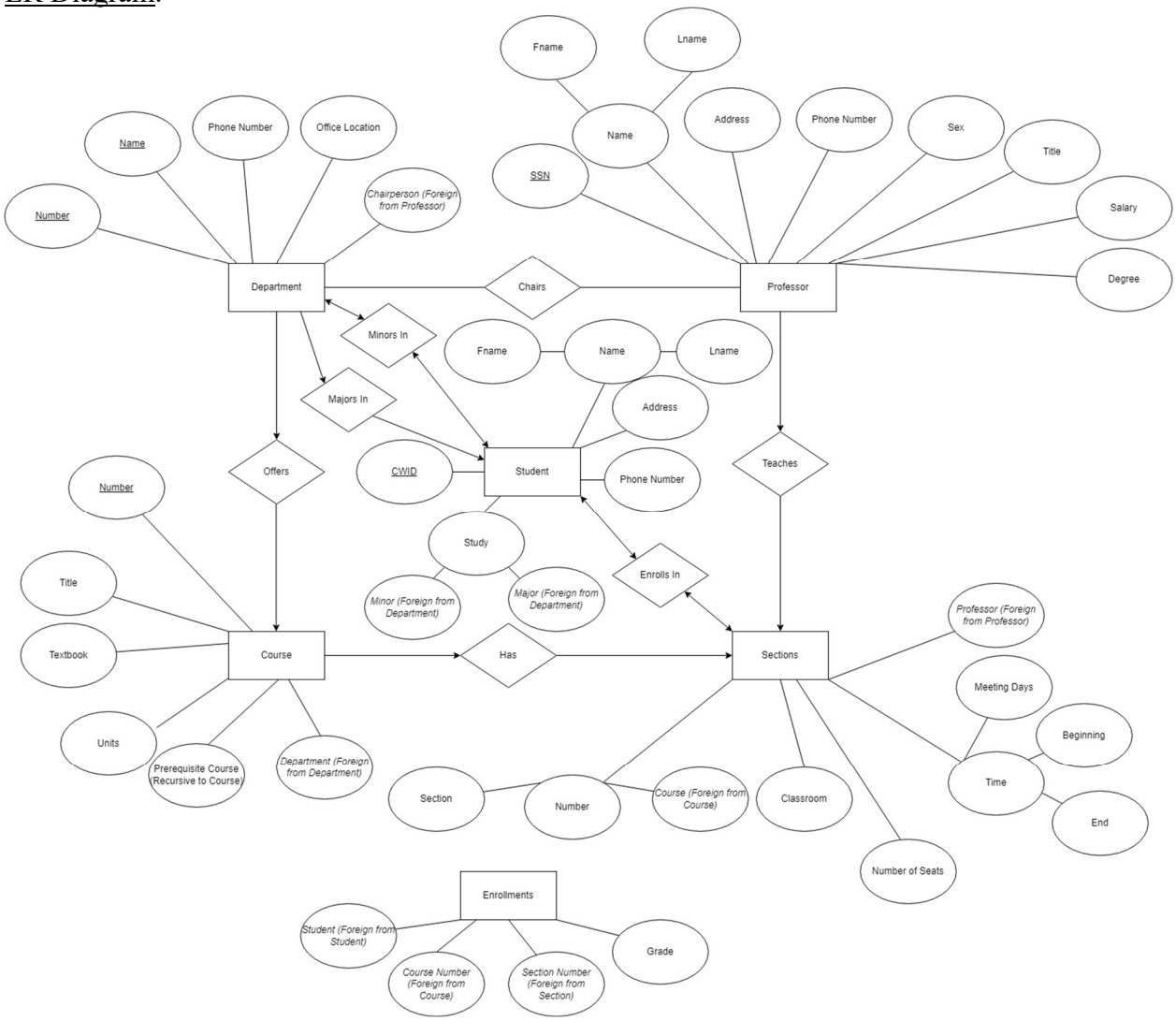
California State University, Fullerton
Computer Science
CPSC 332 Project - Database
Zeid Aldaas – 885097022
5/16/2024

Project Description

The objective of this project was to design and implement a web database application for a university. The database manages information related to professors, departments, courses, sections, students, and enrollments. The project was developed using MySQL for the database management and PHP for the web interface.

Database Design

ER Diagram:



The Entity-Relationship (ER) diagram was designed to model the university database system. The primary entities included Professors, Departments, Courses, Sections, Students, and Enrollments. Relationships between these entities were established to reflect the real-world connections, such as professors teaching sections, students enrolling in sections, and courses being offered by departments.

NOTE:
Rectangle = Entity
Rounded Rectangle = Associative Link Between Entities

Oval = Attributes
Diamond = Relationships
Underlined = Primary Key
Italicized = Foreign Key
Arrows indicate the "Many" in the One-to-Many/Many-to-One/Many-to-Many relationships.

Relational Model:

The relational model was derived from the ER diagram:

TABLES

1. Professor

SSN (Primary Key)
FirstName
LastName
StreetAddress
City
State
ZipCode
AreaCode
PhoneNumber
Sex
Title
Salary
CollegeDegrees

2. Department

DeptNumber (Primary Key)
DeptName
Telephone
OfficeLocation
ChairpersonSSN (Foreign Key referencing Professor.SSN)

3. Course

CourseNumber (Primary Key)
Title
Textbook
Units
DeptNumber (Foreign Key referencing Department.DeptNumber)

4. Prerequisite

CourseNumber (Composite Key, Foreign Key referencing Course.CourseNumber)
PrerequisiteCourseNumber (Composite Key, Foreign Key referencing Course.CourseNumber)

5. Section

CourseNumber (Composite Key, Foreign Key referencing Course.CourseNumber)

SectionNumber (Composite Key)
Classroom
NumberOfSeats
MeetingDays
BeginTime
EndTime
ProfessorSSN (Foreign Key referencing Professor.SSN)

6. Student

CampusWideID (Primary Key)
FirstName
LastName
StreetAddress
City
State
ZipCode
AreaCode
PhoneNumber
MajorDeptNumber (Foreign Key referencing Department.DeptNumber)

7. StudentMinor

CampusWideID (Composite Key, Foreign Key referencing Student.CampusWideID)
DeptNumber (Composite Key, Foreign Key referencing Department.DeptNumber)

8. Enrollment

CampusWideID (Foreign Key referencing Student.CampusWideID)
CourseNumber (Foreign Key referencing Section.CourseNumber)
SectionNumber (Foreign Key referencing Section.SectionNumber)
Grade

FOREIGN KEY CONSTRAINTS:
Department.ChairpersonSSN → Professor.SSN
Course.DeptNumber → Department.DeptNumber
Prerequisite.CourseNumber → Course.CourseNumber
Prerequisite.PrerequisiteCourseNumber → Course.CourseNumber
Section.CourseNumber → Course.CourseNumber
Section.ProfessorSSN → Professor.SSN
Student.MajorDeptNumber → Department.DeptNumber
StudentMinor.CampusWideID → Student.CampusWideID
StudentMinor.DeptNumber → Department.DeptNumber
Enrollment.CampusWideID → Student.CampusWideID
Enrollment.CourseNumber → Section.CourseNumber
Enrollment.SectionNumber → Section.SectionNumber
Composite Keys:
Section: (CourseNumber, SectionNumber)

Prerequisite: (CourseNumber, PrerequisiteCourseNumber)
StudentMinor: (CampusWideID, DeptNumber)

Implementation

Database Creation:

SQL scripts (create_tables.sql and insert_data.sql) were written to create the tables and populate them with sample data. The create_tables.sql script defined the structure of each table, specifying primary keys and foreign keys to enforce relationships. The insert_data.sql script inserted sample records into each table.

PHP Web Interface:

A series of PHP scripts were developed to provide interfaces for different users:
Students: Students can log in using their campus-wide ID to view their enrolled courses and grades. They can also look up details about specific courses.
Professors: Professors can log in using their SSN to view the sections they are teaching. They can also view the grades of students in each section.
Administrators: Administrators can log in to access features similar to both students and professors, including looking up professors, students, courses, and grades.

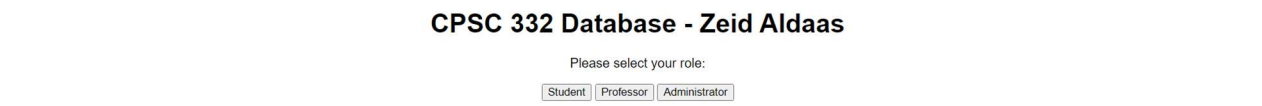
User Interactions:

The web interface allows users to interact with the database through forms and links. Validations and error handling were implemented to ensure proper input and feedback. For example, pop-up alerts inform users of invalid inputs like incorrect CWIDs, SSNs, course numbers, or section numbers.

Testing and Documentation

Screenshots & Sample Runs:

Screenshots were taken to document the interfaces and sample runs of the system. These screenshots demonstrate the functionality of the application, including user inputs and corresponding outputs:



Main Menu

CPSC 332 Database - Zeid Aldaas

Student Login

Enter your CWID Submit

Back

CPSC 332 Database - Zeid Aldaas

Professor Login

Enter your SSN Submit

Back

CPSC 332 Database - Zeid Aldaas

Administrator Login

Enter password Submit

Back

Robert Brown

Courses and Grades

Course Number: 101 - Course: Introduction to Computer Science - Grade: A
Course Number: 201 - Course: Calculus I - Grade: B-

Course Lookup

Enter Course Number | Submit

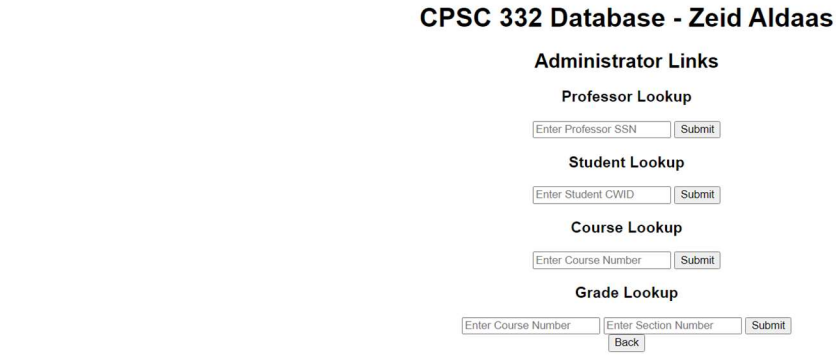
Back

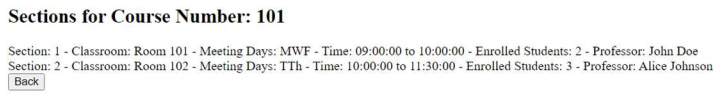
John Doe

Course: Introduction to Computer Science, 101 - Section: 1 - Classroom: Room 101 - Meeting Days: MWF - Time: 09:00:00 to 10:00:00
Course: Linear Algebra, 202 - Section: 1 - Classroom: Room 203 - Meeting Days: TTh - Time: 15:00:00 to 16:30:00

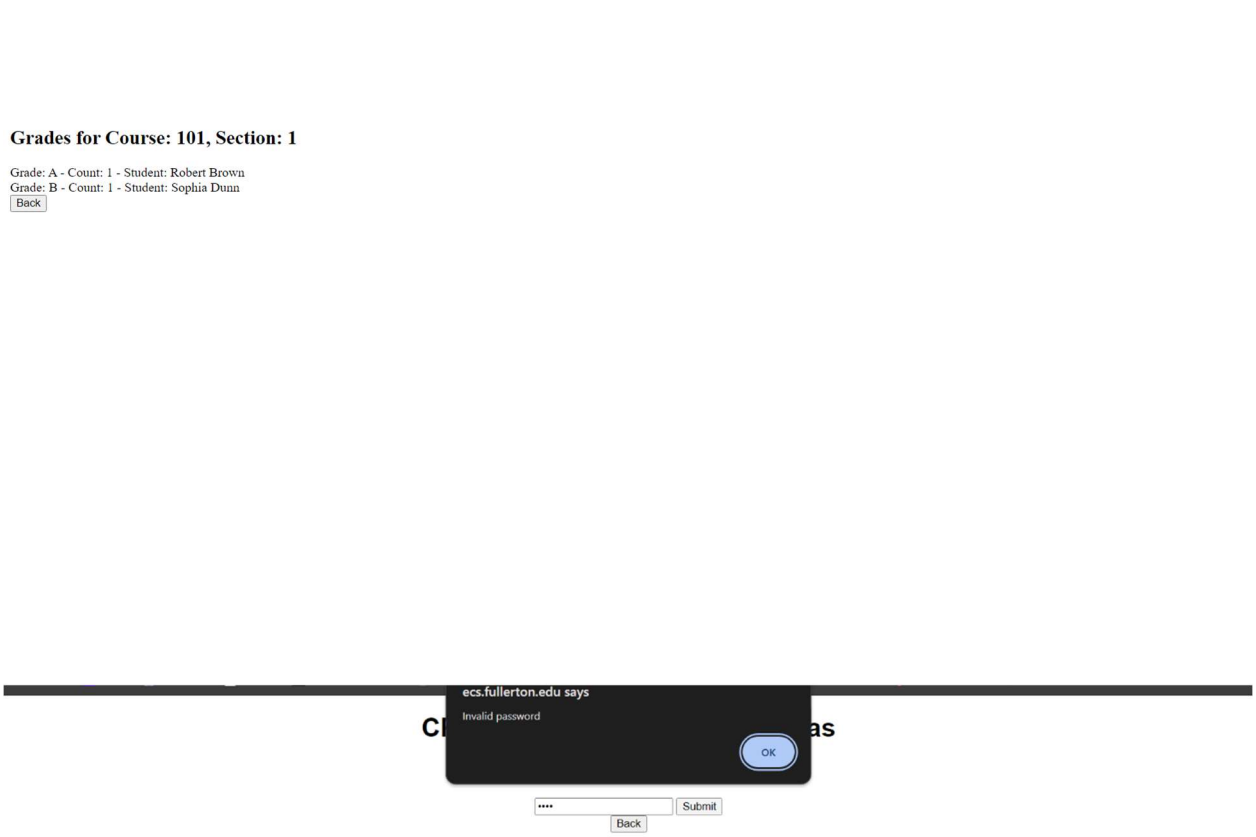
Back

Student and Professor Course View (CWID and SSN input: “123456789”) Other sample IDs available in insert_data.sql.





Administrator Options View and Student Course Lookup



Course Grades View and Invalid Input Prompt (Correct administrator password is “password”). Note: Invalid input popups show up for any invalid entries, not just for the administrator password.

Source Code

Below is the Source Code. Note, index.html is not here due to it being too large. I emailed the source code to your email just in case you’d like to view all of them in their file format and see index.html:

create_tables.sql:

```
USE cs332b4;

SET FOREIGN_KEY_CHECKS = 0;

DROP TABLE IF EXISTS Enrollments;
DROP TABLE IF EXISTS Students;
DROP TABLE IF EXISTS Sections;
```

```
DROP TABLE IF EXISTS Courses;
DROP TABLE IF EXISTS Departments;
DROP TABLE IF EXISTS Professors;

SET FOREIGN_KEY_CHECKS = 1;

CREATE TABLE Professors (
    social_security_number CHAR(9) PRIMARY KEY,
    name VARCHAR(100),
    street_address VARCHAR(100),
    city VARCHAR(50),
    state CHAR(2),
    zip_code CHAR(5),
    area_code CHAR(3),
    number CHAR(7),
    sex CHAR(1),
    title VARCHAR(50),
    salary DECIMAL(10, 2),
    college_degrees VARCHAR(255)
);

CREATE TABLE Departments (
    department_number INT PRIMARY KEY,
    name VARCHAR(50),
    telephone CHAR(10),
    office_location VARCHAR(100),
    chairperson_social_security_number CHAR(9),
    FOREIGN KEY (chairperson_social_security_number) REFERENCES
Professors(social_security_number)
);

CREATE TABLE Courses (
    course_number INT PRIMARY KEY,
    title VARCHAR(100),
    textbook VARCHAR(255),
    units INT,
    department_number INT,
    FOREIGN KEY (department_number) REFERENCES Departments(department_number)
);

CREATE TABLE Sections (
    section_number INT,
    course_number INT,
    classroom VARCHAR(50),
    number_of_seats INT,
    meeting_days VARCHAR(20),
```

```
    beginning_time TIME,
    ending_time TIME,
    professor_social_security_number CHAR(9),
    PRIMARY KEY (section_number, course_number),
    FOREIGN KEY (course_number) REFERENCES Courses(course_number),
    FOREIGN KEY (professor_social_security_number) REFERENCES
Professors(social_security_number)
);

CREATE TABLE Students (
    campus_wide_id CHAR(9) PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    street_address VARCHAR(100),
    city VARCHAR(50),
    state CHAR(2),
    zip_code CHAR(5),
    area_code CHAR(3),
    number CHAR(7),
    major_department_number INT,
    FOREIGN KEY (major_department_number) REFERENCES
Departments(department_number)
);

CREATE TABLE Enrollments (
    student_id CHAR(9),
    section_number INT,
    course_number INT,
    grade CHAR(2),
    PRIMARY KEY (student_id, section_number, course_number),
    FOREIGN KEY (student_id) REFERENCES Students(campus_wide_id),
    FOREIGN KEY (section_number, course_number) REFERENCES Sections(section_number,
course_number)
);
```

insert_data.sql:

```
USE cs332b4;

-- Insert records into Professors table
INSERT INTO Professors (social_security_number, name, street_address, city, state, zip_code,
area_code, number, sex, title, salary, college_degrees) VALUES
('123456789', 'John Doe', '123 Moonbrook St', 'Los Angeles', 'CA', '90005', '323', '8546234', 'M',
'Professor', 90000.00, 'PhD in Computer Science'),
('987654321', 'Jane Smith', '456 Cathedral Dr', 'Irvine', 'CA', '92657', '949', '3629942', 'F',
'Associate Professor', 80000.00, 'MS in Mathematics'),
('111223333', 'Alice Johnson', '789 Barren St', 'Anaheim', 'CA', '92815', '714', '6629012', 'F',
```

'Assistant Professor', 70000.00, 'PhD in Software Engineering');

-- Insert records into Departments table
INSERT INTO Departments (department_number, name, telephone, office_location, chairperson_social_security_number) VALUES
(1, 'Computer Science', '6575551234', 'CS Building Room 110', '123456789'),
(2, 'Mathematics', '6575555678', 'Math Building Room 210', '987654321');

-- Insert records into Courses table
INSERT INTO Courses (course_number, title, textbook, units, department_number) VALUES
(101, 'Introduction to Computer Science', 'Computer Science 101', 3, 1),
(102, 'Data Structures', 'Data Structures and Algorithms', 3, 1),
(201, 'Calculus I', 'Calculus I', 4, 2),
(202, 'Linear Algebra', 'Linear Fundamentals', 3, 2);

-- Insert records into Sections table
INSERT INTO Sections (section_number, course_number, classroom, number_of_seats, meeting_days, beginning_time, ending_time, professor_social_security_number) VALUES
(1, 101, 'Room 101', 30, 'MWF', '09:00:00', '10:00:00', '123456789'),
(2, 101, 'Room 102', 25, 'TTh', '10:00:00', '11:30:00', '111223333'),
(1, 102, 'Room 103', 20, 'MWF', '11:00:00', '12:00:00', '111223333'),
(1, 201, 'Room 201', 30, 'TTh', '13:00:00', '14:30:00', '987654321'),
(2, 201, 'Room 202', 35, 'MWF', '14:00:00', '15:00:00', '987654321'),
(1, 202, 'Room 203', 40, 'TTh', '15:00:00', '16:30:00', '123456789');

-- Insert records into Students table
INSERT INTO Students (campus_wide_id, first_name, last_name, street_address, city, state, zip_code, area_code, number, major_department_number) VALUES
('123456789', 'Robert', 'Brown', '123 Elm St', 'Fullerton', 'CA', '92831', '657', '1234567', 1),
('987654321', 'Emily', 'Davis', '456 Oak Ave', 'Corona', 'CA', '92882', '951', '7654321', 1),
('112233445', 'Michael', 'Wilson', '789 Pine St', 'Fullerton', 'CA', '92833', '714', '3259012', 2),
('519283746', 'Sarah', 'Miller', '321 Birch St', 'Fullerton', 'CA', '92831', '657', '7823456', 2),
('998877665', 'David', 'Taylor', '654 Cedar Blvd', 'Tustin', 'CA', '92780', '949', '9217890', 1),
('591827364', 'Jessica', 'Anderson', '987 Maple St', 'Fullerton', 'CA', '92834', '657', '1302345', 1),
('808007911', 'Daniel', 'Thomas', '654 Spruce Cir', 'Riverside', 'CA', '92505', '951', '4106789', 2),
('829158368', 'Laura', 'Jackson', '321 Willow St', 'Fullerton', 'CA', '92831', '714', '6630123', 2),
('648385912', 'Paul', 'Smith', '839 Briar Ln', 'Anaheim', 'CA', '92816', '714', '8472723', 2),
('773828964', 'Sophia', 'Dunn', '934 Berry St', 'Fullerton', 'CA', '92833', '657', '7774813', 2);

-- Insert records into Enrollments table
INSERT INTO Enrollments (student_id, section_number, course_number, grade) VALUES
('123456789', 1, 101, 'A'),
('987654321', 2, 101, 'B+'),
('112233445', 1, 102, 'A-'),
('519283746', 2, 101, 'B'),
('998877665', 1, 201, 'C+'),

```
('591827364', 2, 201, 'B-'),
('808007911', 1, 202, 'A'),
('829158368', 2, 201, 'B+'),
('123456789', 2, 201, 'B-'),
('987654321', 1, 102, 'A'),
('112233445', 2, 201, 'B'),
('519283746', 1, 202, 'C'),
('998877665', 2, 101, 'A-'),
('591827364', 1, 102, 'B+'),
('808007911', 2, 201, 'C+'),
('829158368', 1, 202, 'B-'),
('648385912', 1, 102, 'A+'),
('773828964', 1, 101, 'B'),
('648385912', 1, 201, 'A-'),
('773828964', 2, 201, 'C+');
```

clear_data.sql

```
USE cs332b4;

-- Disable foreign key checks to avoid constraint issues
SET FOREIGN_KEY_CHECKS = 0;

-- Delete existing records from tables
DELETE FROM Enrollments;
DELETE FROM Students;
DELETE FROM Sections;
DELETE FROM Courses;
DELETE FROM Departments;
DELETE FROM Professors;

-- Re-enable foreign key checks
SET FOREIGN_KEY_CHECKS = 1;
```

professor_classes.php:

```
<?php
$servername = "mariadb";
$username = "cs332b4";
$password = "00vhYNjC";
$dbname = "cs332b4";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
```

```
}

$ssn = $_GET['ssn'];

// Fetch professor's name
$name_sql = "SELECT name FROM Professors WHERE social_security_number = '$ssn'";
$name_result = $conn->query($name_sql);

if ($name_result->num_rows > 0) {
    $name_row = $name_result->fetch_assoc();
    $professor_name = $name_row['name'];
    echo "<h2>$professor_name</h2>";
} else {
    echo "<script>alert('Invalid SSN'); window.history.back();</script>";
    $conn->close();
    exit();
}

$sql = "SELECT Courses.course_number, Courses.title, Sections.section_number,
Sections.classroom, Sections.meeting_days, Sections.beginning_time, Sections.ending_time
        FROM Professors
        JOIN Sections ON Professors.social_security_number =
Sections.professor_social_security_number
        JOIN Courses ON Sections.course_number = Courses.course_number
        WHERE Professors.social_security_number = '$ssn'";

$result = $conn->query($sql);

if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        echo "<a href='count_grades.php?course_number=" . $row["course_number"] .
"&section_number=" . $row["section_number"] . "'>";
        echo "Course: " . $row["title"]. ", " . $row["course_number"]. " - Section: " .
$row["section_number"]. " - Classroom: " . $row["classroom"]. " - Meeting Days: " .
$row["meeting_days"]. " - Time: " . $row["beginning_time"]. " to " . $row["ending_time"];
        echo "</a><br>";
    }
} else {
    echo "0 results";
}

echo '<button onclick="window.history.back()">Back</button>';

$conn->close();
?>
```

count_grades.php:

```
<?php
$servername = "mariadb";
$username = "cs332b4";
$password = "00vhYNjC";
$dbname = "cs332b4";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

$course_number = $_GET['course_number'];
$section_number = $_GET['section_number'];

$sql = "SELECT grade, COUNT(*) as count,
GROUP_CONCAT(CONCAT(Students.first_name, ' ', Students.last_name) SEPARATOR ', ') as
student_names
      FROM Enrollments
      JOIN Students ON Enrollments.student_id = Students.campus_wide_id
      WHERE course_number = '$course_number' AND section_number = '$section_number'
      GROUP BY grade";

$result = $conn->query($sql);

echo "<h2>Grades for Course: $course_number, Section: $section_number</h2>";
if ($result->num_rows > 0) {
   while($row = $result->fetch_assoc()) {
      echo "Grade: " . $row["grade"]. " - Count: " . $row["count"]. " - Student: " .
$row["student_names"]. "<br>";
   }
} else {
   echo "<script>alert('Invalid course/section number'); window.history.back();</script>";
   $conn->close();
   exit();
}

echo '<button onclick="window.history.back()">Back</button>';

$conn->close();
?>
```

student_courses.php:

```
<?php
$servername = "mariadb";
$username = "cs332b4";
$password = "00vhYNjC";
$dbname = "cs332b4";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$student_id = $_GET['student_id'];

// Fetch student's name
$name_sql = "SELECT first_name, last_name FROM Students WHERE campus_wide_id =
'$student_id'";
$name_result = $conn->query($name_sql);

if ($name_result->num_rows > 0) {
    $name_row = $name_result->fetch_assoc();
    $student_name = $name_row['first_name'] . ' ' . $name_row['last_name'];
    echo "<h2>$student_name</h2>";
} else {
    echo "<script>alert('Invalid CWID'); window.history.back();</script>";
    $conn->close();
    exit();
}

$sql = "SELECT Courses.course_number, Courses.title, Enrollments.grade
        FROM Enrollments
        JOIN Courses ON Enrollments.course_number = Courses.course_number
        WHERE Enrollments.student_id = '$student_id'";

$result = $conn->query($sql);

echo "<h2>Courses and Grades</h2>";
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        echo "Course Number: " . $row["course_number"]. " - Course: " . $row["title"]. " - Grade:
" . $row["grade"]. "<br>";
    }
} else {
```

```
   echo "0 results";
}

echo "<h3>Course Lookup</h3>";
echo '<form action="course_sections.php" method="get">
      <input type="text" name="course_number" placeholder="Enter Course Number" required>
      <button type="submit">Submit</button>
     </form>';

echo '<button onclick="window.history.back()">Back</button>';

$conn->close();
?>
```

course_sections.php:

```
<?php
$servername = "mariadb";
$username = "cs332b4";
$password = "00vhYNjC";
$dbname = "cs332b4";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

$course_number = $_GET['course_number'];

$sql = "SELECT Sections.section_number, Sections.classroom, Sections.meeting_days,
Sections.beginning_time, Sections.ending_time, COUNT(Enrollments.student_id) as
student_count, Professors.name as professor_name
      FROM Sections
      LEFT JOIN Enrollments ON Sections.section_number = Enrollments.section_number AND
Sections.course_number = Enrollments.course_number
      LEFT JOIN Professors ON Sections.professor_social_security_number =
Professors.social_security_number
      WHERE Sections.course_number = '$course_number'
      GROUP BY Sections.section_number, Sections.classroom, Sections.meeting_days,
Sections.beginning_time, Sections.ending_time, Professors.name";

$result = $conn->query($sql);

echo "<h2>Sections for Course Number: $course_number</h2>";
```

```
if ($result->num_rows > 0) {
   while($row = $result->fetch_assoc()) {
      echo "Section: " . $row["section_number"]. " - Classroom: " . $row["classroom"]. " -
Meeting Days: " . $row["meeting_days"]. " - Time: " . $row["beginning_time"]. " to " .
$row["ending_time"]. " - Enrolled Students: " . $row["student_count"]. " - Professor: " .
$row["professor_name"]. "<br>";
   }
} else {
   echo "<script>alert('Invalid course number'); window.history.back();</script>";
   $conn->close();
   exit();
}

echo '<button onclick="window.history.back()">Back</button>';

$conn->close();
?>
```

Conclusion

This project successfully implemented a comprehensive web database application for managing university data. It demonstrated the use of MySQL for database management and PHP for creating user-friendly web interfaces. The project covered the entire development lifecycle, from database design and implementation to interface development and testing.