**Individual project:** Student Registrations System at XYZ University

Disclaimer: This document has been created for internal usage at Florida International University (FIU), specifically for the course COP 4710: Database Management Systems students. Please do not share or distribute it with third parties, persons, or organizations outside FIU.

Please complete the following information:

**Student name: Carlos**

**Student surname: Moya Ramirez**

**Student email address: cmoya016@fiu.edu**

**Student panther ID: 6347108**

**Student Registrations System at XYZ University**

**Introduction**

In today's digital age, most universities have adopted sophisticated database systems to manage their students' information. University XYZ is no exception and has developed its own Student Registrations System to keep track of student records. However, the system lacks a crucial feature that can make student registration easier - **a register and login page**. Currently, students need to visit the university's enrolment department in person to register their information manually. This approach is not only time-consuming but can also lead to errors in record-keeping.

To address this issue, the university has decided to undertake a project to develop a database system that will allow students to register and manage their information quickly. The project aims to create a user-friendly system for students to register their details, such as personal information, academic records, and extracurricular activities. The system will also provide features to allow students to view and update their records, track their academic progress, and communicate with the university staff.

The project is an essential step for the university toward digital transformation, making it easier for students to manage their academic life. The new system will improve the student experience and help the university staff streamline administrative processes, reduce errors, and enhance data security. With the successful implementation of the new system, the university can expect to see increased student satisfaction, improved administrative efficiency, and a better overall academic environment.

The project will involve creating a Conceptual E-R model and a Relational model for the database, using Structured Query Language in MySQL database, creating a simple GUI for only a register and login page of an application using a programming language to connect to the MySQL database, and performing select, insert, update, and delete statements through GUI.

**Entities**

It would help if you considered defining **at least six entities**. For example, the following six entities **may be** required for the database system:

**Student** - This entity represents a student and includes attributes such as student ID, name, address, phone number, email address, and date of birth.

Example:

| Student ID | First Name | Last Name | Address | Phone Number | Email | Date of Birth |

|---------------|----------------|----------------|------------|---------------------|---------|-------------------|

| 1 | John | Doe | 123 Main St. | (555) 555-1234 | johndoe@email.com | 01/01/2000 |

**Course** - This entity represents a course and includes attributes such as course ID, course name, instructor name, start time, end time, and room number.

Example:

| Course ID | Course Name | Instructor Name | Start Time | End Time | Room Number |

|--------------|-------------------|------------------------|---------------|--------------|---------------------|

| 1 | Introduction to Computer Science | Dr. Smith | 9:00am | 10:30am | Room 101 |

**Enrolment** - This entity represents a student's enrolment in a course and includes attributes such as enrolment ID, student ID, course ID, and grade.

Example:

| Enrolment ID | Student ID | Course ID | Grade |

|-------------------|---------------|---------------|---------|

| 1 | 1 | 1 | A |

**Schedule** - This entity represents a student's schedule and includes attributes such as student ID, course ID, start time, end time, and room number.

Example:

| Student ID | Course ID | Start Time | End Time | Room Number |

|---------------|---------------|---------------|--------------|--------------------|

| 1 | 1 | 9:00am | 10:30am | Room 101 |

**Department** - This entity represents a department and includes attributes such as department ID and name.

Example:

| Department ID | Department Name |

|---------------------|--------------------------|

| 1 | Computer Science |

**Professor** - This entity represents a professor and includes attributes such as professor ID, name, address, phone number, and email address.

Example:

| Professor ID | First Name | Last Name | Address | Phone Number | Email |

|-----------------|----------------|----------------|------------|---------------------|---------|

| 1 | Alex | Doe | 369 Sun St. | (888) 333-6699 | alexdoe@email.com |

**Task 1**: Create a Conceptual E-R Model for the Database

The primary objective of the first task of this project is to develop a conceptual E-R model for the database that will define the structure of the database, relationships, and attributes. To create the E-R model, at least six entities, such as students, faculty, courses, departments, grades, and extracurricular activities, need to be defined along with their relationships and attributes.

These entities will be the foundation of the database and ensure that the data can be accurately captured and easily accessed. In addition, the E-R model will allow University XYZ to understand the data flow within the system, ensuring that the database structure meets the organization's needs.

A tool like Lucidchart can be used to create the E-R model. The E-R model created will serve as a blueprint for designing the database, guiding the development of data storage and access capabilities for University XYZ.

In conclusion, creating a conceptual E-R model is a critical first step in designing a database system that is functional and efficient for University XYZ. Defining at least six entities, relationships, and attributes will help ensure that the system meets the organization's needs, allowing for accurate data capture and easy access.

**Note**: several other tools are available for creating E-R models besides Lucidchart. Some of the most popular tools for creating E-R diagrams include:

Draw.io: A free and open-source diagramming tool that can create a wide range of diagrams, including E-R diagrams.

Gliffy: A cloud-based diagramming tool that allows users to create E-R and other diagrams.

SmartDraw: A diagramming tool that provides a range of templates, including E-R diagrams, to help users create professional-looking diagrams quickly.

Creately: A cloud-based diagramming tool that offers a range of templates, including E-R diagrams and collaboration features for teams.

ConceptDraw DIAGRAM: A diagramming tool that provides a range of templates and libraries, including E-R diagrams, and allows users to create custom templates.

These tools offer functionality similar to Lucidchart, each with strengths and weaknesses. The choice of tool depends on factors such as the user's preferences, budget, and the project's specific needs.

**Task 2**: Create a Relational Model for the Database in MySQL

In this task, you will create a relational model for the database in MySQL. This approach is an essential step as the relational model will define the structure of the database, including the tables, columns, and relationships between them. The relational model will be based on the conceptual E-R model created in Task 1, which identifies at least six entities that will be used in the database.

To create the relational model, the project team will need to define the tables used in the database, along with their respective columns and data types. In addition, the relationships between the tables must also be defined, such as the primary key-foreign key relationships that link records in different tables.

For example, the project team can create a "students" table using SQL script. The "students" table may contain columns such as student\_id, first\_name, last\_name, address, phone\_number, email, and date\_of\_birth. The student\_id column will serve as the primary key for the table. Other columns, such as first\_name, last\_name, and email, will be defined as varchar data types, while columns, such as date\_of\_birth, will be defined as the date data type.

It would help if you created additional tables as the conceptual E-R model required, ensuring that each table is appropriately linked to other tables through primary key and foreign key relationships. By creating a well-defined relational model, the database can be structured easily to query and manage, allowing for efficient data storage and retrieval. Following are two pieces of SQL script to create two tables as two examples for you:

CREATE TABLE students (

student\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

address VARCHAR(100),

phone\_number VARCHAR(20),

email VARCHAR(50),

date\_of\_birth DATE

);

CREATE TABLE courses (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(50),

instructor\_name VARCHAR(50),

start\_time TIME,

end\_time TIME,

room\_number VARCHAR(20)

);

**Task 3**: Use Structured Query Language

Task 3 requires using Structured Query Language (SQL) to manage the data in the database. SQL is a standard language used for managing relational databases. You will use SQL to create at least six tables and insert, update, delete, and retrieve data from the database.

In this task, you will focus on performing operations related to the **registrations and login of a student**. Therefore, you must write SQL statements such as select, insert, update, and delete for the Student table. In addition, you should write at least six SQL queries to retrieve information from the Student table based on specific criteria. For instance, you might want to retrieve the names of all students in the table or retrieve a student's password based on their ID.

In addition to retrieving data, you will write one SQL statement to insert data into the Student table, one SQL query to update data in the Student table based on the student's ID, and one SQL query to delete a student based on their ID.

By writing SQL queries, you will show how to retrieve, manipulate, and manage data in the database. This approach will help you better understand how the database system works and how data can be organized, queried, and managed efficiently.

**Task 4**: Create a Simple GUI for an Application

This task is focused on creating a simple graphical user interface (GUI) for an application that can interact with the MySQL database created in Task 3. The GUI will allow users to select, insert, update, and delete statements on the database. In addition, the students are required to implement the **register and login page of the application using any programming language** like Java or Python. For example, JDBC (Java Database Connectivity) can connect the Java application to the MySQL database. In contrast, PyMySQL can be used in Python to connect to the MySQL database.

The GUI will be designed to make it easy for users to interact with the database using buttons, text fields, and other user interface components. The students can use any GUI development tools, such as JavaFX or Swing in Java, to create the interface. In Python, the Tkinter library can create a graphical interface to communicate with the MySQL database.

Once the GUI is created, users can register by providing their details, which will be inserted into the MySQL database using the insert statement. Users can then log in using their registered credentials, and the GUI should authenticate them by querying the MySQL database using the select statement. Users should also be able to update their details and delete their accounts using the update and delete statements.

Overall, Task 4 requires you to create a functional GUI that interacts with the MySQL database created in Task 3, enabling users to perform basic CRUD (Create, Read, Update, Delete) operations on the database.

**Note**: Here are a few more programming language examples and the corresponding database connections that can be used for Task 4:

C# - You can use the .NET Framework's ADO.NET library to connect to a MySQL database. The connection string looks something like this:

string connString = "SERVER=localhost;DATABASE=myDatabase;UID=myUsername;PASSWORD=myPassword;";

PHP - PHP has built-in support for MySQL databases through the MySQLi and PDO extensions. Here is an example connection string using MySQLi:

$conn = mysqli\_connect("localhost", "myUsername", "myPassword", "myDatabase");

Ruby - You can use the Ruby MySQL library to connect to a MySQL database. Here is an example connection code:

require 'mysql2'

client = Mysql2::Client.new(

host: 'localhost',

username: 'myUsername',

password: 'myPassword',

database: 'myDatabase'

)

Node.js - You can use the mysql2 library in Node.js to connect to a MySQL database. Here is an example code:

const MySQL = require('mysql2/promise');

const pool = mysql.createPool({

host: 'localhost',

user: 'myUsername',

password: 'myPassword',

database: 'myDatabase'

});

These are just a few examples, and many other programming languages and libraries can be used to connect to a MySQL database.

**Task 5**: Perform Select, Insert, Update, and Delete Statements Through GUI

In this task, you must ensure that the GUI you created in task 4 can perform the necessary SQL statements for registration and login scenarios. This approach means that application users can interact with the database using the GUI interface to perform select, insert, update, and delete operations.

For example, a user might use the GUI to input their registration details, such as their name, email address, and password. The application will then perform an insert statement on the database to add the user's details to the relevant table. Similarly, users might use the GUI to input their login details, such as their email address and password. The application will then perform a select statement on the database to check if the user's details match those in the relevant table and, if so, allow the user to log in.

You will need to ensure that the necessary SQL statements are written and implemented in the backend code of the application, allowing the GUI to communicate with the database and perform the desired operations. This approach may involve using libraries and frameworks specific to the programming language you have chosen to create the application and implementing error handling and validation to ensure data integrity and prevent security vulnerabilities.

**Conclusion**:

In conclusion, developing a user-friendly database system that allows XYZ university students to register and manage their information efficiently is a crucial step for University XYZ in today's digital age. By implementing this project, the university is taking a significant step towards improving the overall academic experience for its students. The new system will make it easier for students to register and manage their information and help the university staff streamline administrative processes, reduce errors, and enhance data security. With the successful implementation of this project, the university can expect to see increased student satisfaction, improved administrative efficiency, and a better overall academic environment. This project represents an essential investment in the university's future and will undoubtedly positively impact students and staff.

**What should you deliver?**

1. Write all the entity names and their columns and other information similar to how I wrote in the **Entities** section (9 points).

YOUR ANSWER:

**Student** - This entity represents a student and includes attributes such as id, first name, last name, address, phone number, email address, and date of birth.

Example:

| ID | First Name | Last Name | Email | Password| Phone Number | Date of Birth |Address |

|----|----------------|----------------|--------|--------------|---------------------|-------------------|-----------|

| 1 | John | Doe | johndoe@email.com | 123456 | (555) 555-1234 | 2000-5-8 |123 Main St.|

**Course** - This entity represents a course and includes attributes such as id, course name, course code, credits, and professor id.

Example:

| ID | Course Name | Course Code | Credits | Professor ID

|----|-------------------|-------------------|-----------|----------------|

| 1 | Introduction to Computer Science | COP 4512 | 3 | 1 |

**Enrolment** - This entity represents a student's enrolment in a course and includes attributes such as id, semester, year, program of study, degree, status, student id, and course id.

Example:

| ID | Semester | Year| Program of Study| Degree | Status | Student ID | Course ID |

|-----|--------------|------|-----------------------|-----------|----------|----------------|--------------|

| 1 | Spring | 2023 | Computer Science | Bachelor | Enrolled | 1| 2|

**Academic Record** - This entity represents an academic record and includes attributes such as id, course name, course code, grade, credits, semester, year, and student id.

Example:

| ID | Course Name | Course Code | Grade | Credits | Semester | Year | Student ID |

|-----|-------------------|------------------|----------|----------|--------------|--------|----------------|

| 1 | Introduction to Computer Science | COP 4512 | A | 3 | Fall | 2022 | 2 |

**Extracurricular Activity** - This entity represents an extracurricular activity and includes attributes such as id, activity name, description, start date, end date, and student id.

Example:

| ID | Activity Name | Description | Start Date | End Date | Student ID |

|----|---------------------|-----------------|--------------|--------------|----------------|

| 5 | Streets Cleaning | Student goes to the streets and clean | 2023-1-3 | 2023-1-10 | 3 |

**Professor** - This entity represents a professor and includes attributes such as id, first name, last name, email, and phone number.

Example:

| ID | First Name | Last Name | Email | Phone Number |

|----|----------------|----------------|--------|-----------------------|

| 1 | Mykel | Doyle | alexdoe@email.com | 7861885469 |

2. Perform task 1 and capture an image of the Lucidchart's conceptual E-R model for the database and add it to this section. You may use any other tools for this task (9 points).

YOUR ANSWER:

Graphical user interface

Description automatically generatedDiagram

Description automatically generated

3. Perform task 2 and add all the images and SQL codes related to creating the relational model for the database in MySQL in this section (9 points).

YOUR ANSWER:

CREATE DATABASE xyz\_database;

CREATE TABLE xyz\_database.Students (

id INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

email VARCHAR(50) NOT NULL,

password VARCHAR(50) NOT NULL,

phone\_number VARCHAR(20) NOT NULL,

date\_of\_birth DATE NOT NULL,

address VARCHAR(100) NOT NULL

);

CREATE TABLE xyz\_database.Professors(

id INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

email VARCHAR(50) NOT NULL,

phone\_number VARCHAR(20) NOT NULL

);

CREATE TABLE xyz\_database.Courses (

id INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

course\_name VARCHAR(100) NOT NULL,

course\_code VARCHAR(20) NOT NULL,

credits INT NOT NULL,

professor\_id INT NOT NULL,

FOREIGN KEY (professor\_id) REFERENCES Professors(id)

);

CREATE TABLE xyz\_database.Enrollment (

id INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

semester VARCHAR(20) NOT NULL,

year INT NOT NULL,

program\_of\_study VARCHAR(50) NOT NULL,

degree VARCHAR(50) NOT NULL,

status VARCHAR(20) NOT NULL,

student\_id INT NOT NULL,

course\_id INT NOT NULL,

FOREIGN KEY (student\_id) REFERENCES Students(id),

FOREIGN KEY (course\_id ) REFERENCES Courses (id)

);

CREATE TABLE xyz\_database.Academic\_Records (

id INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

course\_name VARCHAR(100) NOT NULL,

course\_code VARCHAR(20) NOT NULL,

grade VARCHAR(5) NOT NULL,

credits INT NOT NULL,

semester VARCHAR(20) NOT NULL,

year INT NOT NULL,

student\_id INT NOT NULL,

FOREIGN KEY (student\_id) REFERENCES Students(id)

);

CREATE TABLE xyz\_database.Extracurricular\_Activites (

id INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

activity\_name VARCHAR(50) NOT NULL,

description VARCHAR(100) NOT NULL,

start\_date DATE NOT NULL,

end\_date DATE NOT NULL,

student\_id INT NOT NULL,

FOREIGN KEY (student\_id) REFERENCES Students(id)

);

4. Perform task 3 and write all the SQL queries for SELECT, INSERT, UPDATE, and DELETE statements of the register and login page of the system (15 points).

YOUR ANSWER:

INSERT INTO `xyz\_database`.`students` (`id`, `first\_name`, `last\_name`, `email`, `password`, `phone\_number`, `date\_of\_birth`, `address`) VALUES ('1', 'Javier', 'Vergara', 'jver10@xyz.edu', '123', '80588954178', '1999-8-8', '718 N 78th Ave');

INSERT INTO `xyz\_database`.`students` (`id`, `first\_name`, `last\_name`, `email`, `password`, `phone\_number`, `date\_of\_birth`, `address`) VALUES ('2', 'Garry', 'Nunes', 'gnunes78@xyz.edu', '321', '7892567418', '2001-8-1', '121 W 8th St');

UPDATE `xyz\_database`.`students` SET `date\_of\_birth` = '2001-08-5' WHERE (`id` = '2');

DELETE FROM `xyz\_database`.`students` WHERE (`id` = '2');

SELECT first\_name, last\_name FROM xyz\_database.students;

SELECT first\_name FROM xyz\_database.students WHERE email = 'jver10@xyz.edu';

SELECT password FROM xyz\_database.students WHERE id = '1';

SELECT \* FROM xyz\_database.students;

SELECT \* FROM xyz\_database.students WHERE date\_of\_birth BETWEEN '1999-1-1' AND '2004-12-12';

SELECT email FROM xyz\_database.students WHERE phone\_number = '80588954178';

5. Perform task 4 and write what language did you use? What connectivity API did you use? What is your connection string using the chosen programming language? Why did you use this programming language? Capture the screen of the register and login page of the software application you designed (15 points).

YOUR ANSWER:

I employ Java to develop the graphical user interface (GUI) in conjunction with the Java Database Connectivity (JDBC) application programming interface (API). The JDBC API facilitates connectivity to MySQL through employment of the connection string "jdbc:mysql://localhost:3306/xyz\_database" utilizing the "root" user with a "123" password. JDBC is a preferred choice for interfacing with MySQL as it offers a uniform means of accessing databases in Java, rendering it system-agnostic and straightforward to adapt to diverse environments. Additionally, given my extensive familiarity with the Java programming language, it is my preferred medium for this task.

Login Page

Table

Description automatically generated

Register page

Table

Description automatically generated

6. Include all the source codes related to performing SELECT, INSERT, UPDATE, and DELETE Statements for the register and login page. The source should show the SQL commands in MySQL and the commands used in your programming language to call these SQL statements (9 points).

YOUR ANSWER:

For Login Page:

if (e.getSource() == loginButton) {  
 try {  
 // retrieve student info from database  
 String email = emailField.getText();  
 String password = new String(passwordField.getPassword());  
 Statement statement = connection.createStatement();  
 ResultSet result = statement.executeQuery("SELECT id FROM Students WHERE email = '" + email + "' AND password = '" + password + "'");  
 if (result.next()) {  
 int studentID = result.getInt("id");  
 getStudentInfo(studentID);  
 displayStudentInfo();  
 dispose();  
 } else {  
 JOptionPane.*showMessageDialog*(this, "Invalid email or password", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 } catch (SQLException ex){  
 JOptionPane.*showMessageDialog*(this, "Error retrieving student information from database", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 ex.printStackTrace();}  
}

}

For Register Page:

else if (e.getSource() == registerButton) {  
 // create new registration frame  
 JFrame registrationFrame = new JFrame("Student Registration");  
 registrationFrame.setSize(400, 400);  
 // create form panel  
 JPanel panel = new JPanel(new GridLayout(8, 2));  
 panel.add(new JLabel("First Name:"));  
 panel.add(new JTextField());  
 panel.add(new JLabel("Last Name:"));  
 panel.add(new JTextField());  
 panel.add(new JLabel("Email:"));  
 panel.add(new JTextField());  
 panel.add(new JLabel("Password:"));  
 panel.add(new JPasswordField());  
 panel.add(new JLabel("Phone Number:"));  
 panel.add(new JTextField());  
 panel.add(new JLabel("Date of Birth:"));  
 panel.add(new JTextField());  
 panel.add(new JLabel("Address:"));  
 panel.add(new JTextField());  
 panel.add(new JLabel());  
 JButton registerButton = new JButton("Register");  
 panel.add(registerButton);  
  
 // add action listener to register button  
 registerButton.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 // insert new student into database  
 try {  
 String firstName = ((JTextField)panel.getComponent(1)).getText();  
 String lastName = ((JTextField)panel.getComponent(3)).getText();  
 String email = ((JTextField)panel.getComponent(5)).getText();  
 String password = new String(((JPasswordField)panel.getComponent(7)).getPassword());  
 String phoneNumber = ((JTextField)panel.getComponent(9)).getText();  
 String dateOfBirth = ((JTextField)panel.getComponent(11)).getText();  
 String address = ((JTextField)panel.getComponent(13)).getText();  
  
 Statement statement = connection.createStatement();  
 String query = "INSERT INTO Students (first\_name, last\_name, email, phone\_number, date\_of\_birth, address, password) " +  
 "VALUES ('" + firstName + "', '" + lastName + "', '" + email + "', '" + phoneNumber + "', '" + dateOfBirth + "', '" + address + "', '" + password + "')";  
 statement.executeUpdate(query);  
 JOptionPane.*showMessageDialog*(registrationFrame, "Registration successful", "Success", JOptionPane.*INFORMATION\_MESSAGE*);  
 registrationFrame.dispose();  
 } catch (SQLException ex) {  
 JOptionPane.*showMessageDialog*(registrationFrame, "Error registering student", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 ex.printStackTrace();  
 }  
 }  
 });  
  
 // add panel to frame  
 registrationFrame.add(panel);  
 registrationFrame.setVisible(true);  
}

Update Info:

if (e.getSource() == saveButton) {  
 try {  
 // update student information in the database  
 String updateQuery = "UPDATE students SET first\_name = ?, last\_name = ?, email = ?, phone\_number = ?, date\_of\_birth = ?, address = ? WHERE id = ?";  
 PreparedStatement preparedStatement = connection.prepareStatement(updateQuery);  
 preparedStatement.setString(1, firstNameField.getText());  
 preparedStatement.setString(2, lastNameField.getText());  
 preparedStatement.setString(3, emailField.getText());  
 preparedStatement.setString(4, phoneNumberField.getText());  
 preparedStatement.setString(5, dateOfBirthField.getText());  
 preparedStatement.setString(6, addressField.getText());  
 preparedStatement.setInt(7, studentInfo.getInt("id"));  
 preparedStatement.executeUpdate();  
  
 JOptionPane.*showMessageDialog*(this, "Student information updated successfully. Log in again to see changes.");  
 dispose();  
 } catch (SQLException exception) {  
 JOptionPane.*showMessageDialog*(this, "Error updating student information: " + exception.getMessage());  
 }  
}

7. Write any limitations and bugs your system includes. You can add a capture screen of your issues in the following section (6 points).

YOUR ANSWER:

To the best of my knowledge, the present version is devoid of any known bugs. However, in the previous version, there was an issue where the graphical user interface (GUI) did not display the buttons.

Graphical user interface

Description automatically generated

The program's scope is restricted to modifying exclusively the data stored in the student and enrolment tables. This constraint is deliberate, as the purpose of the application is to facilitate students' registration in the university's database and enable them to manage their courses efficiently.

8. Write any obstacles, issues, problems, difficulties, etc., you encountered during the design, development, implementation, and demonstration of the Student Registrations System at XYZ University (6 points).

YOUR ANSWER:

The primary challenge entailed acquiring proficiency in novel development skills such as designing a graphical user interface (GUI) and integrating it with the database. This project represented my initial exposure to such a task, which posed certain difficulties, including the initial lack of clarity regarding the integration of the JDBC Jar. Furthermore, iteratively refining the GUI aesthetics represented a time-intensive undertaking.

9. Make a video screen of your working system in which you demonstrate your database design, diagrams, tables, relationships, and software application for register and login functionalities. First, you should add your name, surname, and other required information to the MySQL database through your register and login GUI of your software application. Next, you need to perform INSERT, UPDATE, DELETE, and SELECT statements via the GUI. Once you have made each statement, go to the MySQL database and show that your command has been successfully operated.

Add your project source code, all the documents, including this word file, and all other related files of your project, including your SQL commands, images, etc., in a ZIP file and upload three files to Canvas: 1) a video demonstration file, 2) a ZIP file including all the files of your project, the documents, and images, and 3) the filled in version of this word document to Canvas portal (21 points).

YOUR ANSWER:

The documents have been submitted via the Canvas platform.

**Do not send your source code via the email address or Canvas message** (1 point). You will lose all 25 points for the individual project if you send your tasks via email or Canvas message.

End of the document.