Mini project report on

Student Database Management System

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering

UE21CS351 – DBMS Project

Submitted by:

Heemaj Yadav	**PES2UG21CS198**
Isha V Rao	**PES2UG21CS205**

Under the guidance of

Dr. Mannar Mannan

Assistant Professor

Designation

PES University

AUG - DEC 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

CERTIFICATE

This is to certify that the mini project entitled

Student Database Management System

is a bonafide work carried out by

Heemaj Yadav	**PES2UG21CS198**
Isha V Rao	**PES2UG21CS205**

In partial fulfilment for the completion of fifth semester DBMS Project (UE20CSS301) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period AUG. 2022 – DEC. 2022. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project has been approved as it satisfies the 5th semester academic requirements in respect of project work.

Signature
Dr. Mannar Mannan
Assistant Professor

DECLARATION

We hereby declare that the DBMS Project entitled **Student Database Management System** has been carried out by us under the guidance of **Dr. Mannar Mannan, Assistant Professor** and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester AUG – DEC 2023.

Heemaj Yadav	**PES2UG21CS198**
Isha V Rao	**PES2UG21CS205**

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. Mannar Mannan, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE21CS351 - DBMS Project.

I take this opportunity to thank Dr. Sandesh B J, C, Professor,ChairPerson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this DBMS Project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

The Student Management System (SMS) is a comprehensive database-driven solution designed to streamline and enhance the management of student information within educational institutions. This system serves as a centralized repository for storing, retrieving, and managing diverse student-related data, providing administrators, faculty, and staff with a powerful tool to efficiently handle various aspects of student administration.

Key features of the Student Management System include student enrollment management, department management, performance reviews, attendance tracking, and document management. The system ensures data accuracy and integrity by implementing robust validation mechanisms, while its user-friendly interface facilitates easy navigation for all stakeholders.

TABLE OF CONTENTS

INTRODUCTION

Welcome to the Student Database Management System powered by Streamlit and Python. This comprehensive system is designed to efficiently manage various aspects of student information, attendance tracking, performance reviews, departmental details, leave records, class teacher assignments, and document management.

System Features:

1. **Student Information:**
   - The core of our system revolves around the student table, where detailed information about each student is stored. This includes personal details, contact information, and other relevant data.

2. **Attendance Tracking:**
   - Keep a close eye on student attendance with the dedicated attendance table. This feature allows for easy tracking of student presence and absence, providing insights into their regularity.

3. **Performance Review:**
   - Evaluate student performance using the performance review table. This module helps in recording and analyzing academic achievements, grades, and overall progress, facilitating better decision-making for educators and administrators.

4. **Department Details:**
   - Manage department-related information with the department table. This includes details about different academic departments, faculty members, and other relevant data to streamline administrative processes.

5. **Leave Records:**
   - Keep track of student leave records efficiently. The system maintains a comprehensive leave record table that captures details of student leaves, reasons, and duration, helping administrators and faculty members manage student absences effectively.

6. **Class Teacher:**
   - Easily assign and track class teachers with the classteacher table. This feature simplifies the process of assigning responsible educators to specific classes, enhancing communication and coordination.

7. **Document Management:**

- Store and organize essential documents securely using the document table. This feature enables the easy retrieval and management of various documents related to students, faculty, and other academic records.

User-Friendly Interface:

Our system is built on the user-friendly Streamlit framework, ensuring a seamless and intuitive experience for both administrators and users. With Streamlit, you can interact with the database effortlessly through a web-based interface, allowing for real-time updates and easy navigation.

Conclusion:

The Student Database Management System using Streamlit and Python provides a comprehensive solution for educational institutions to streamline administrative tasks, enhance communication, and maintain accurate records. Whether you're an educator, administrator, or student, this system is designed to meet the diverse needs of academic management. Dive into the functionalities and discover a powerful tool to transform the way to manage student information.

PROBLEM DEFINITION

In educational institutions, managing diverse aspects of student data and administrative tasks can be challenging and time-consuming. This complexity arises from the need to efficiently handle information related to students, attendance, performance reviews, departments, leave records, class teacher assignments, and documents. Traditional methods of record-keeping often result in inefficiencies, inaccuracies, and difficulties in accessing timely information. To address these challenges, the Student Database Management System (DBMS) using Streamlit and Python is proposed.

Challenges:

1. **Data Discrepancies:**
   - Traditional record-keeping methods can lead to discrepancies and errors in student data, affecting the accuracy of information available to educators, administrators, and other stakeholders.

2. **Time-Consuming Administrative Tasks:**
   - Managing attendance, performance reviews, departmental details, leave records, class teacher assignments, and document management manually can consume a significant amount of time for administrators and educators.

3. **Inefficient Document Handling:**
   - Handling and organizing various documents related to students, faculty, and academic records can be inefficient and prone to errors, leading to challenges in document retrieval and management.

4. **Limited Accessibility:**
   - Accessing and updating student information in real-time is crucial for effective decision-making. Traditional systems may lack the accessibility and responsiveness needed for timely updates.

5. **Communication Gaps:**
   - Inadequate communication channels between faculty, administrators, and students can lead to misunderstandings and a lack of coordination. Clear communication is essential for the smooth functioning of an educational institution.

Proposed Solution:

The Student Database Management System using Streamlit and Python aims to address these challenges by providing a comprehensive, user-friendly, and efficient solution. The proposed system leverages the capabilities of Streamlit for creating an intuitive web-based interface, ensuring easy navigation and real-time updates. The integration of Python allows for robust database management, ensuring the accuracy and security of student-related information.

Objectives:

1. Streamlined Data Management:
   - Ensure accurate and up-to-date student information by centralizing data management through a dedicated database. This helps in reducing data discrepancies and improving the overall quality of information.

2. Efficient Administrative Processes:
   - Streamline administrative tasks such as attendance tracking, performance reviews, leave records, and class teacher assignments to save time and improve overall efficiency.

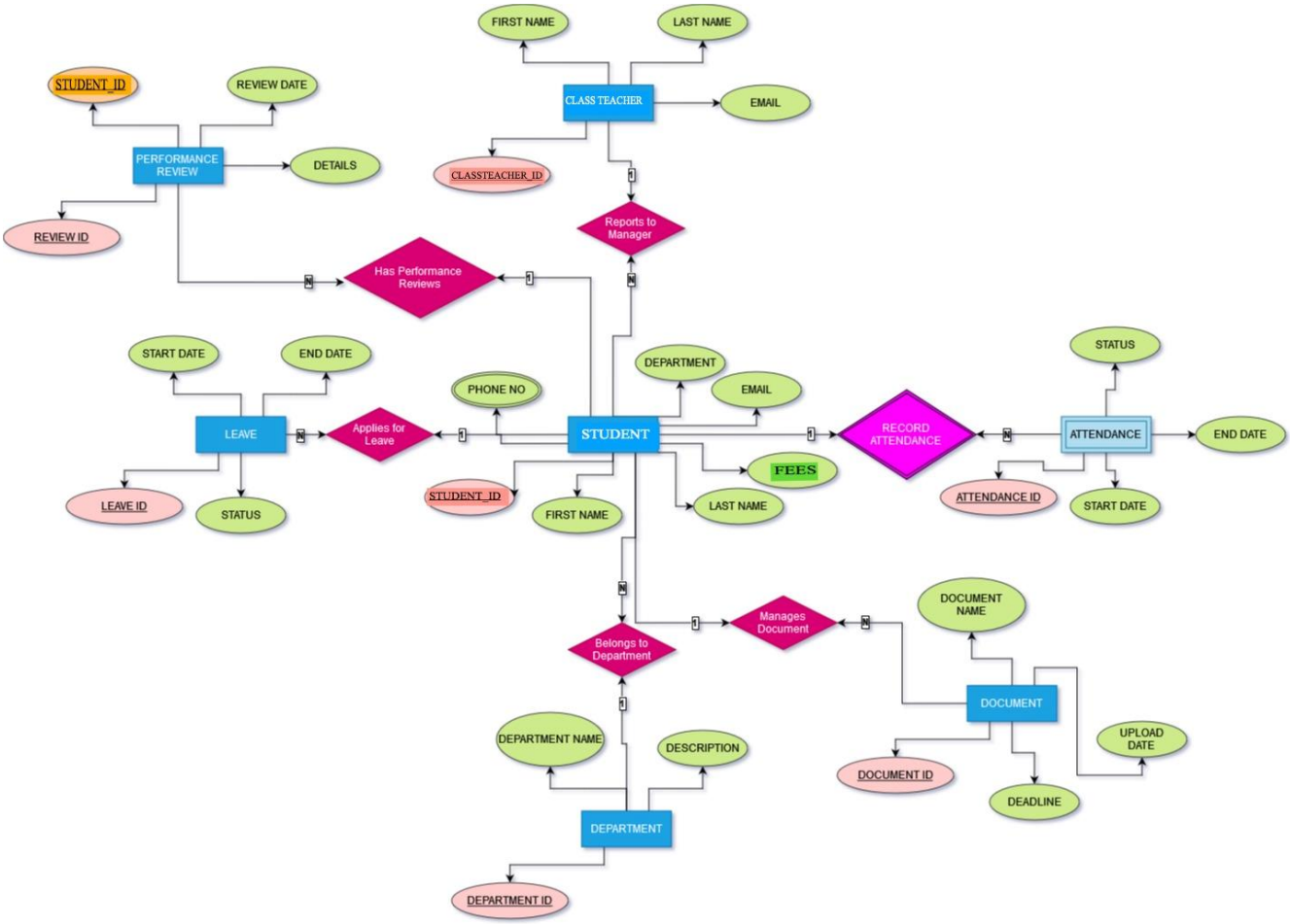
3. Enhanced Document Management:
   - Provide a secure and efficient document management system to store, organize, and retrieve essential documents related to students, faculty, and academic records.

4. Improved Accessibility:
   - Enhance accessibility to student information for administrators, faculty, and other stakeholders, ensuring timely updates and facilitating informed decision-making.

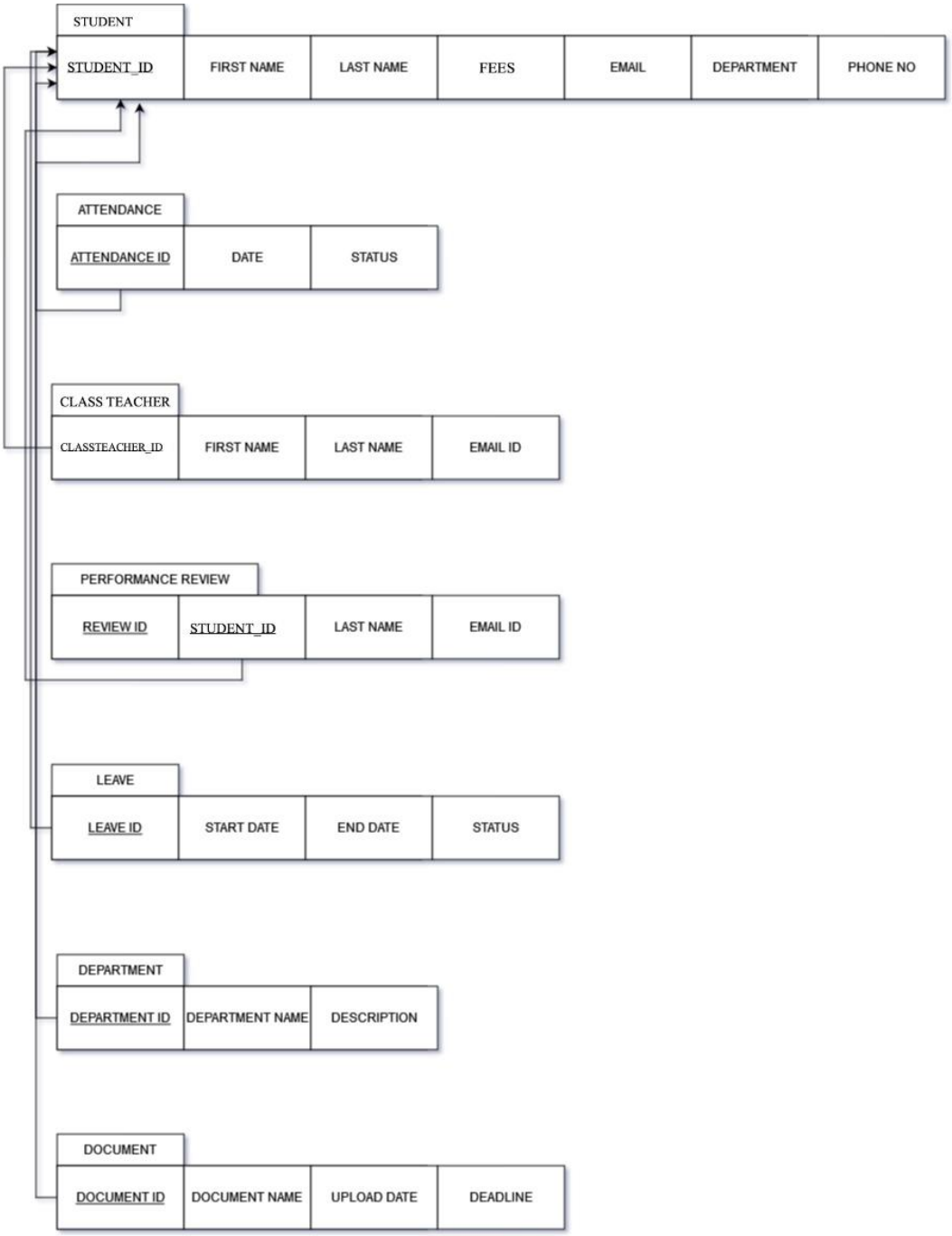
5. Effective Communication:
   - Bridge communication gaps by providing a platform that facilitates clear and efficient communication between educators, administrators, and students.

By addressing these challenges and achieving the outlined objectives, the proposed Student Database Management System aims to transform the management of student-related data in educational institutions, fostering efficiency, accuracy, and collaboration.

ER MODEL


FIRST NAME
LAST NAME
STUDENT_ID
REVIEW DATE
CLASS TEACHER
EMAIL
PERFORMANCE REVIEW
DETAILS
CLASSTEACHER_ID
REVIEW ID
Reports to Manager
Has Performance Reviews
START DATE
END DATE
DEPARTMENT
STATUS
PHONE NO
EMAIL
LEAVE
Applies for Leave
STUDENT
RECORD ATTENDANCE
ATTENDANCE
END DATE
FEES
LEAVE ID
STATUS
STUDENT_ID
FIRST NAME
LAST NAME
ATTENDANCE ID
START DATE
DOCUMENT NAME
Manages Document
Belongs to Department
DOCUMENT
DEPARTMENT NAME
DESCRIPTION
UPLOAD DATE
DOCUMENT ID
DEADLINE
DEPARTMENT
DEPARTMENT ID

ER TO RELATIONAL MAPPING



DDL STATEMENTS

CREATING THE TABLES

```sql
-- Create a new database

CREATE DATABASE student15;

-- Use the 'employee' database
USE student15;

-- Create a table for Employee
CREATE TABLE Student (
    StudentID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Fees DECIMAL(10, 2)
);

-- Create a table for EmployeePhoneNumbers
CREATE TABLE StudentPhoneNumbers (
    PhoneNumberID INT PRIMARY KEY,
    StudentID INT,
    PhoneNumber VARCHAR(15),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID)
);

-- Create a table for Document
CREATE TABLE Document (
    DocumentID INT PRIMARY KEY,
    DocumentName VARCHAR(100),
    UploadDate DATE,
    Deadline DATE,
```

```sql
-- Create a table for Document
CREATE TABLE Document (
    DocumentID INT PRIMARY KEY,
    DocumentName VARCHAR(100),
    UploadDate DATE,
    Deadline DATE,
    StudentID INT,
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID)
);

-- Create a table for Department
CREATE TABLE Department (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(50),
    Description TEXT,
    StudentID INT,
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID)
);

-- Create a table for Attendance
CREATE TABLE Attendance (
    AttendanceID INT PRIMARY KEY,
    StudentID INT,
    Status VARCHAR(10),
    StartDate DATE,
    EndDate DATE,
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID)
);
```

Output

```sql
-- Create a table for LeaveRequest
CREATE TABLE LeaveRequest (
    ReviewID INT PRIMARY KEY,
    StudentID INT,
    ReviewDate DATE,
    Details TEXT,
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID)
);

-- Rename the 'LeaveRequest' table to 'PerformanceReview'
RENAME TABLE LeaveRequest TO PerformanceReview;

-- Create a table for Leave
-- CREATE TABLE Leave (
--     LeaveID INT PRIMARY KEY,
--     EmployeeID INT,
--     StartDate DATE,
--     EndDate DATE,
--     Status VARCHAR(20),
--     FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
-- );

-- Create a table for LeaveRecords
CREATE TABLE LeaveRecords (
    LeaveID INT PRIMARY KEY,
    StudentID INT,
    StartDate DATE,
    EndDate DATE,
    Status VARCHAR(20),
```

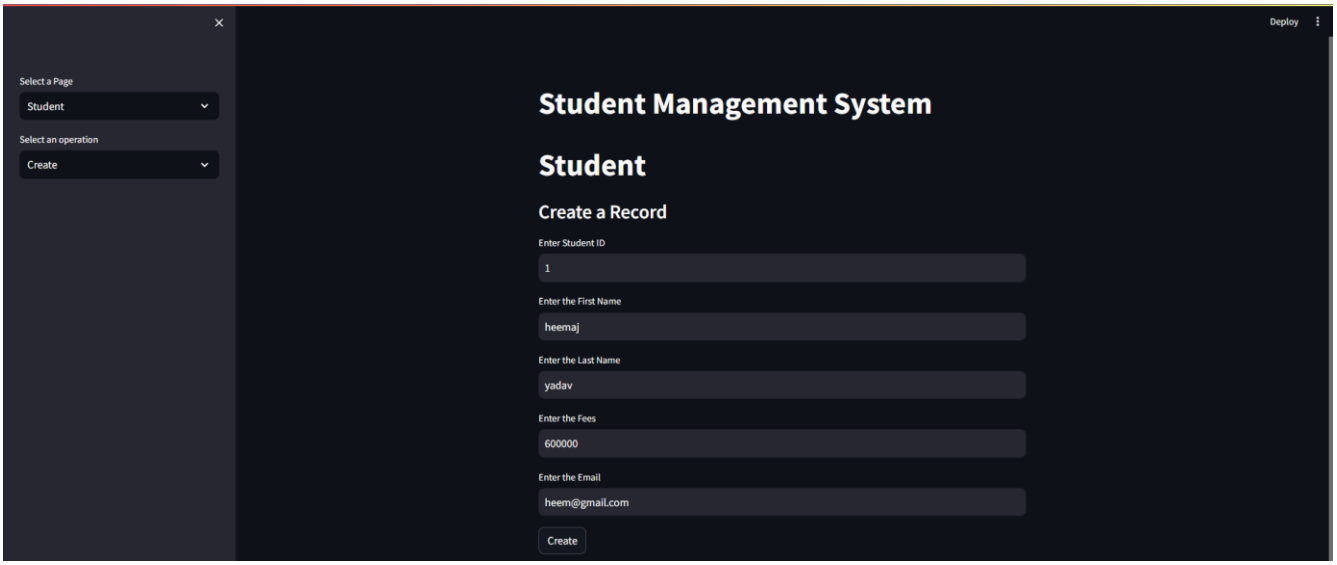
```sql
CREATE TABLE LeaveRecords (
    LeaveID INT PRIMARY KEY,
    StudentID INT,
    StartDate DATE,
    EndDate DATE,
    Status VARCHAR(20),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID)
);

-- Create a table for Manager
CREATE TABLE ClassTeacher (
    ClassTeacherID INT PRIMARY KEY,
    StudentID INT,
    Email VARCHAR(100),
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID)
);

-- Add an 'email' column to the 'Employee' table
ALTER TABLE Student ADD email VARCHAR(50);
```

Action Output

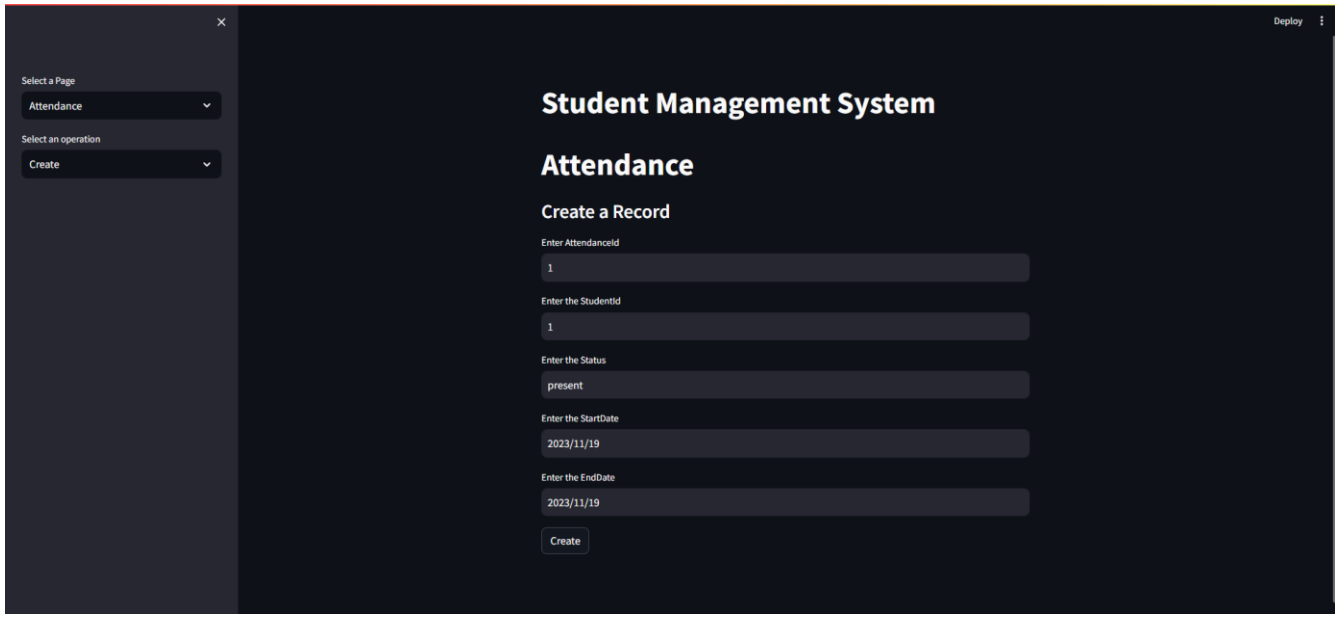
#	Time	Action	Message	Duration / Fetch
1	10:23:51	CREATE DATABASE student15	1 row(s) affected	0.016 sec
2	10:23:51	USE student15	0 row(s) affected	0.000 sec
3	10:23:51	CREATE TABLE Student (StudentID INT PRIMARY KEY, FirstName VARCHAR(50), LastName VARCHA...	0 row(s) affected	0.015 sec
4	10:23:51	CREATE TABLE StudentPhoneNumbers (PhoneNumberID INT PRIMARY KEY, StudentID INT, PhoneN...	0 row(s) affected	0.016 sec
5	10:23:51	CREATE TABLE Document (DocumentID INT PRIMARY KEY, DocumentName VARCHAR(100), Upload...	0 row(s) affected	0.031 sec
6	10:23:51	CREATE TABLE Department (DepartmentID INT PRIMARY KEY, DepartmentName VARCHAR(50), Des...	0 row(s) affected	0.016 sec
7	10:23:52	CREATE TABLE Attendance (AttendanceID INT PRIMARY KEY, StudentID INT, Status VARCHAR(10), ...	0 row(s) affected	0.031 sec
8	10:23:52	CREATE TABLE LeaveRequest (ReviewID INT PRIMARY KEY, StudentID INT, ReviewDate DATE, ...	0 row(s) affected	0.016 sec
9	10:23:52	RENAME TABLE LeaveRequest TO PerformanceReview	0 row(s) affected	0.015 sec
10	10:23:52	CREATE TABLE LeaveRecords (LeaveID INT PRIMARY KEY, StudentID INT, StartDate DATE, End...	0 row(s) affected	0.016 sec
11	10:23:52	CREATE TABLE ClassTeacher (ClassTeacherID INT PRIMARY KEY, StudentID INT, Email VARCHAR(1...	0 row(s) affected	0.016 sec
12	10:23:52	ALTER TABLE Student ADD email VARCHAR(50)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
13	10:23:52	INSERT INTO Student (StudentID, FirstName, LastName, Fees, email) VALUES (1, 'Rajesh', 'Kumar', 60000.00...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.000 sec
14	10:23:52	INSERT INTO Department (DepartmentID, StudentID, DepartmentName, Description) VALUES (1, 1, 'CS', 'CO...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.000 sec
15	10:23:52	INSERT INTO LeaveRecords (LeaveID, StudentID, StartDate, EndDate, Status) VALUES (1, 1, '2023-07-01', '...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.016 sec

DML STATEMENTS

INSERTING THE VALUES INTO TABLES




```python
def create_record():
    st.subheader("Create a Record")
    StudentID = st.text_input("Enter Student ID")
    firstname = st.text_input("Enter the First Name")
    lastname = st.text_input("Enter the Last Name")
    Fees = st.text_input("Enter the Fees")
    email = st.text_input("Enter the Email")

    if st.button("Create"):
        try:
            sql = "INSERT INTO Student (StudentID, firstname, lastname, Fees, email) VALUES (%s, %s, %s, %s, %s)"
            val = (StudentID, firstname, lastname, Fees, email)
            mycursor.execute(sql, val)
            mydb.commit()
            st.success("Record Created Successfully!!!")
        except mysql.connector.Error as err:
            st.error(f"Error: {err}")
    else:
        st.warning("Record not created")
```

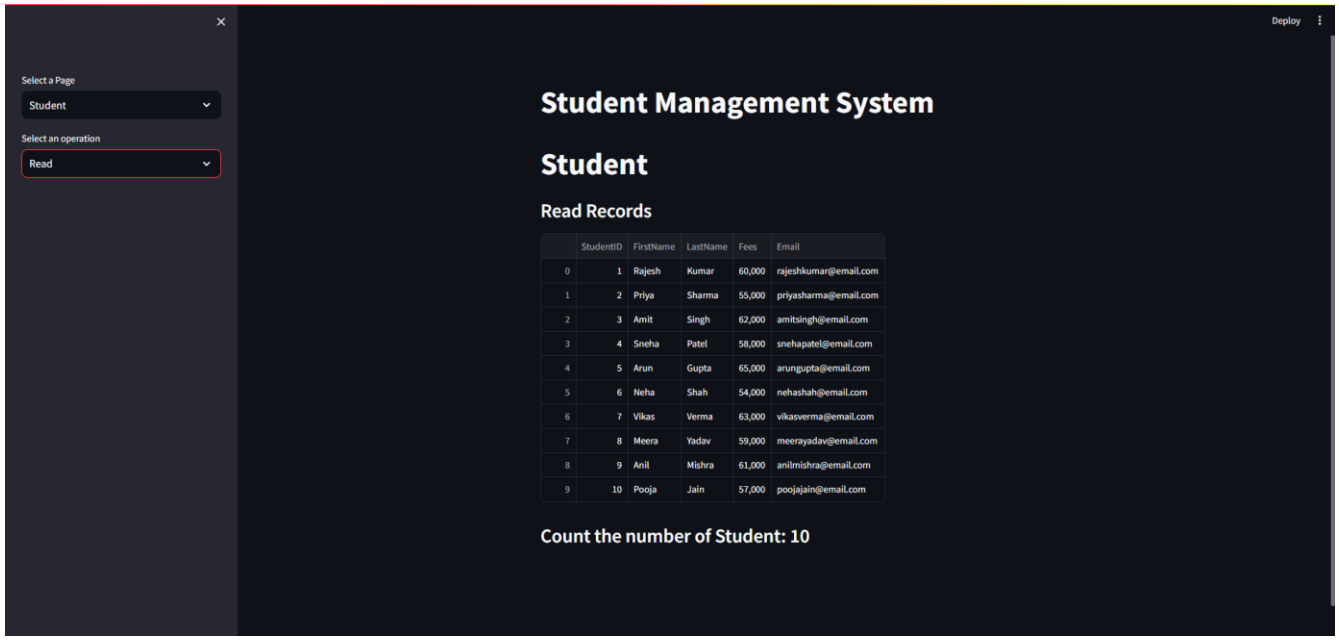


attendance.py > ...

```python
def create_record():
    st.subheader("Create a Record")
    AttendanceID = st.text_input("Enter AttendanceId")
    StudentID = st.text_input("Enter the StudentId")
    Status = st.text_input("Enter the Status")
    StartDate = st.date_input("Enter the StartDate")
    EndDate = st.date_input("Enter the EndDate")

    if st.button("Create"):
        try:
            # Modify the SQL query and values to match your table and columns (attendance table)
            sql = "INSERT INTO attendance (AttendanceID, StudentID, Status, StartDate, EndDate) VALUES (%s, %s, %s, %s, %s
            val = (AttendanceID, StudentID, Status, StartDate, EndDate)

            mycursor.execute(sql, val)
            mydb.commit()
            st.success("Record Created Successfully!!!")
        except mysql.connector.Error as err:
            st.error(f"Error: {err}")
```

QUERIES

SIMPLE QUERY WITH GROUP BY, AGRREGATE

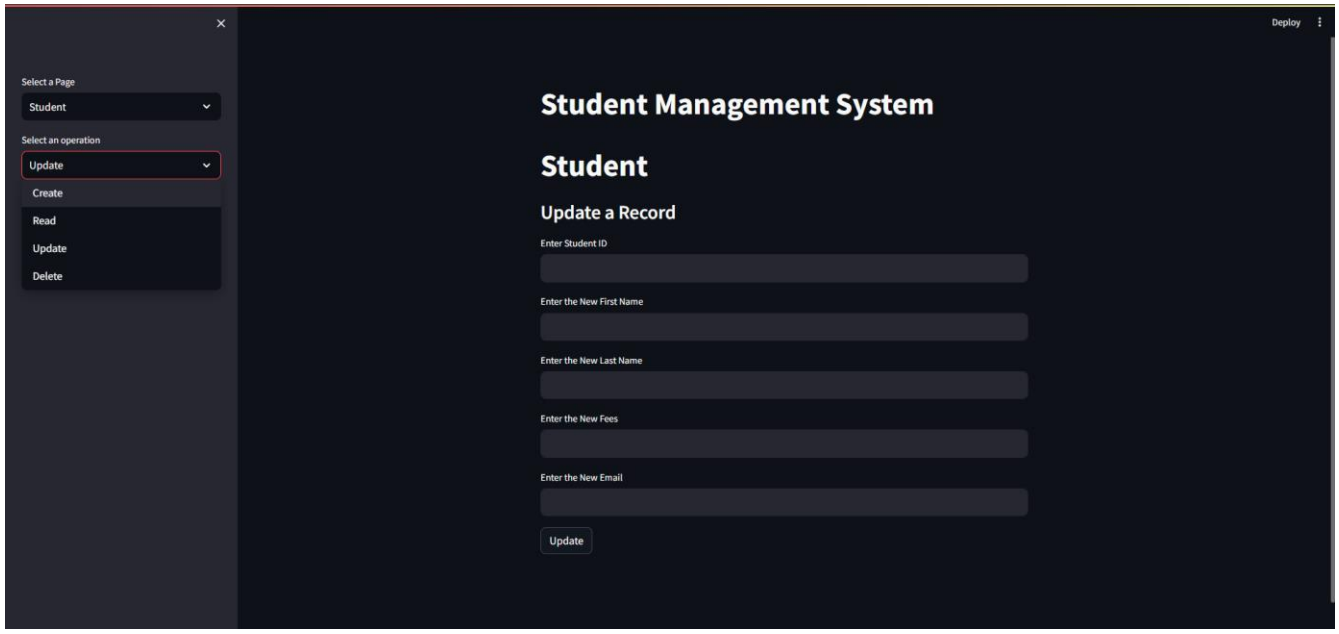


	StudentID	FirstName	LastName	Fees	Email
0	1	Rajesh	Kumar	60,000	rajeshkumar@email.com
1	2	Priya	Sharma	55,000	priyasharma@email.com
2	3	Amit	Singh	62,000	amitsingh@email.com
3	4	Sneha	Patel	58,000	snehapatel@email.com
4	5	Arun	Gupta	65,000	arungupta@email.com
5	6	Neha	Shah	54,000	nehashah@email.com
6	7	Vikas	Verma	63,000	vikasverma@email.com
7	8	Meera	Yadav	59,000	meerayadav@email.com
8	9	Anil	Mishra	61,000	anilmishra@email.com
9	10	Pooja	Jain	57,000	poojajain@email.com

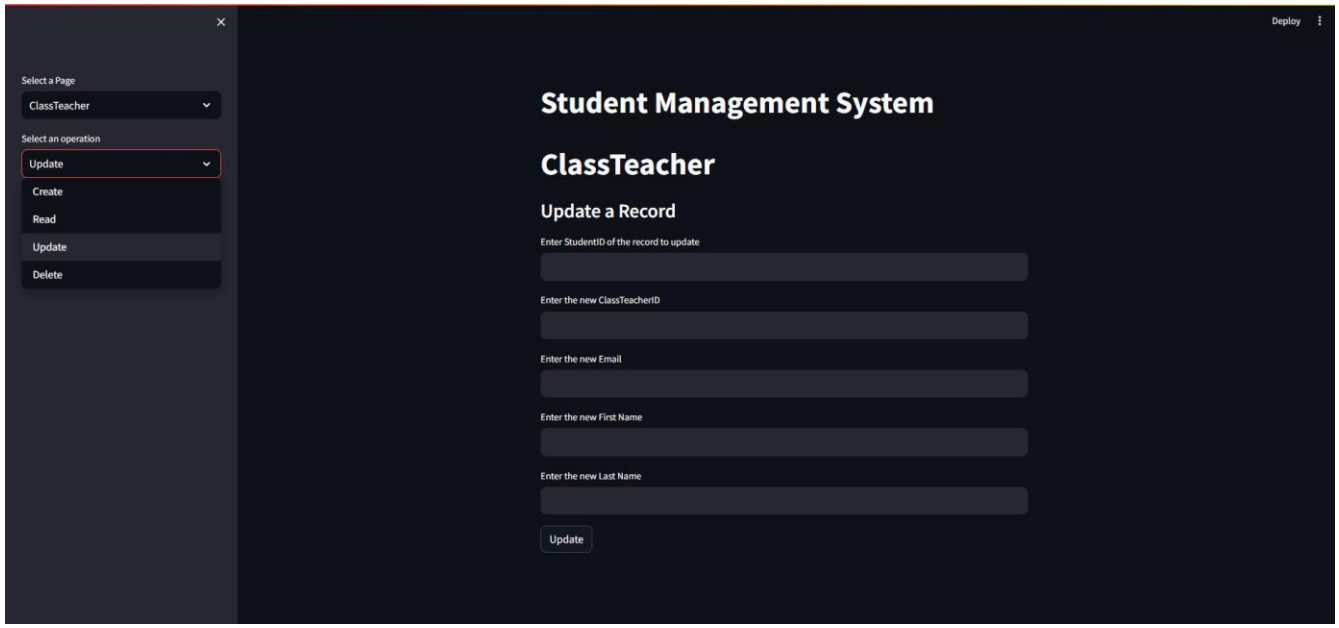
```python
def read_records():
    st.subheader("Read Records")
    mycursor.execute("SELECT * FROM Student")
    result = mycursor.fetchall()
    if result:
        df = pd.DataFrame(result, columns=["StudentID", "FirstName", "LastName", "Fees", "Email"])
        st.dataframe(df)
    else:
        st.write("No records found.")

    mycursor.execute("SELECT count(StudentID) FROM Student")
    result_count = mycursor.fetchone()
    print(result_count)
    count = result_count[0] if result_count else 0
    st.subheader(f"Count the number of Student: {count}")
```

UPDATE OPERATION

Update the table information by entering the primary key attributes



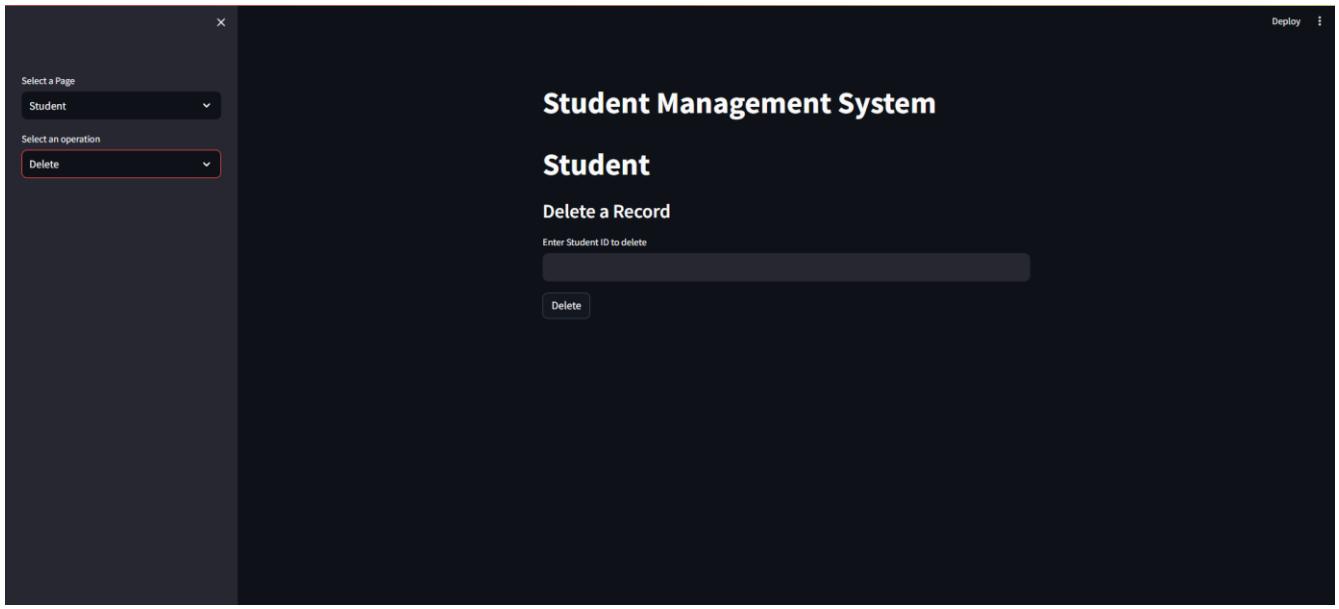

```python
def update_record():
    st.subheader("Update a Record")
    StudentID = st.text_input("Enter Student ID")
    new_firstname = st.text_input("Enter the New First Name")
    new_lastname = st.text_input("Enter the New Last Name")
    new_Fees = st.text_input("Enter the New Fees")
    new_email = st.text_input("Enter the New Email")
    if st.button("Update"):
        try:
            sql = "UPDATE Student SET firstname=%s, lastname=%s, Fees=%s, email=%s WHERE StudentID=%s"
            val = (new_firstname, new_lastname, new_Fees, new_email, StudentID)
            mycursor.execute(sql, val)
            mydb.commit()
            st.success("Record Updated Successfully!!!")
        except mysql.connector.Error as err:
            st.error(f"Error: {err}")
```



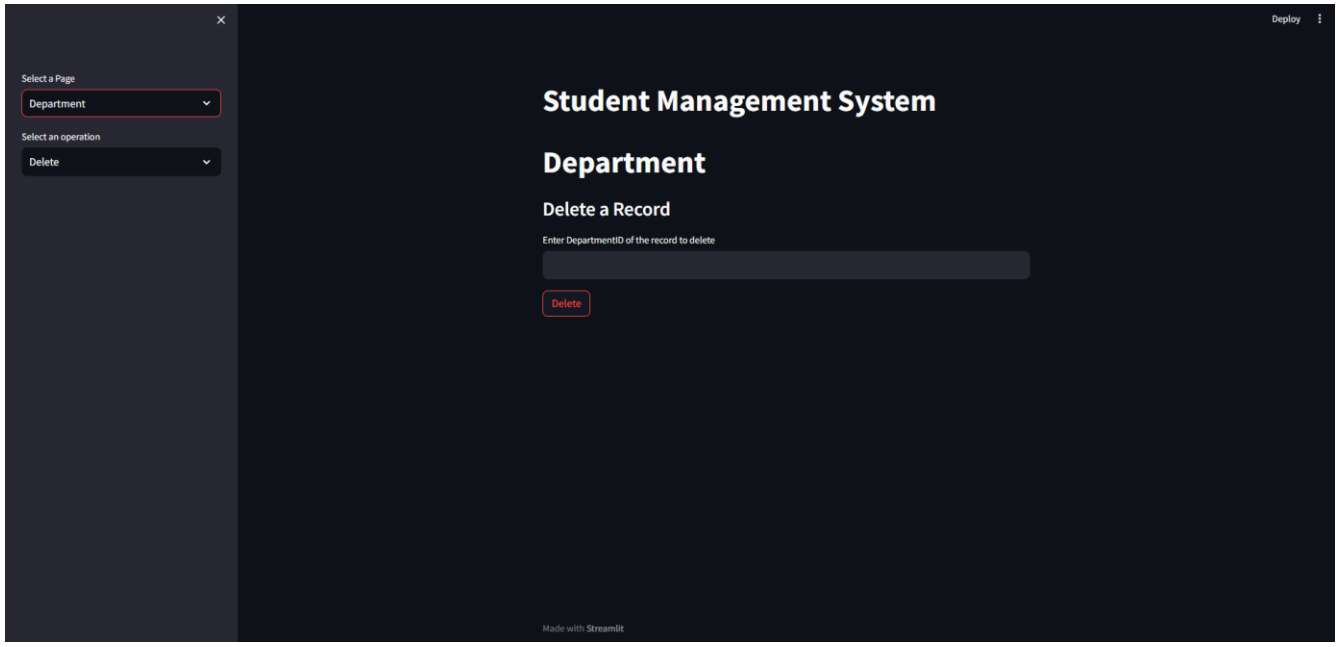

```python
def update_record():
    st.subheader("Update a Record")
    StudentID = st.text_input("Enter StudentID of the record to update")
    new_ClassTeacher_id = st.text_input("Enter the new ClassTeacherID")
    new_email = st.text_input("Enter the new Email")
    new_first_name = st.text_input("Enter the new First Name")
    new_last_name = st.text_input("Enter the new Last Name")

    if st.button("Update"):
        try:
            # Modify the SQL query to update the record with the specified StudentID
            sql = "UPDATE ClassTeacher SET ClassTeacherID = %s, Email = %s, FirstName = %s, LastName = %s WHERE StudentID
            val = (new_ClassTeacher_id, new_email, new_first_name, new_last_name, StudentID)

            mycursor.execute(sql, val)
            mydb.commit()
            st.success("Record Updated Successfully!!!")
        except mysql.connector.Error as err:
            st.error(f"Error: {err}")
```

DELETE OPERATION

Deletes the specified records taking StudentID as input



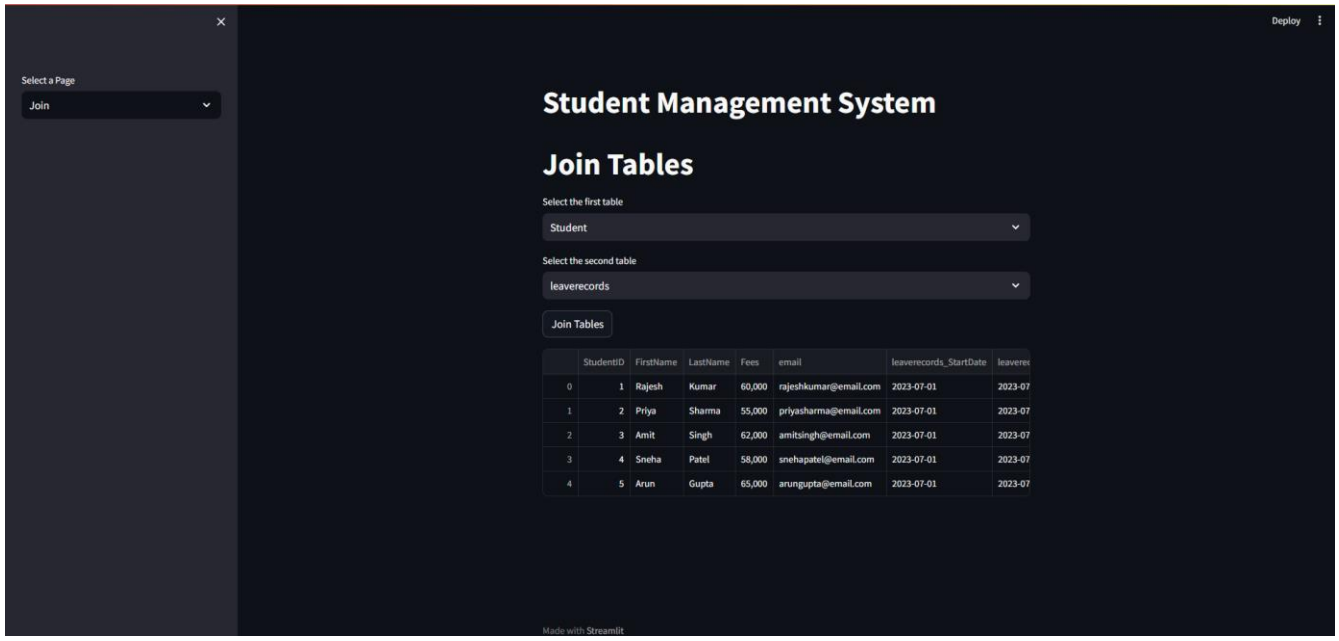

```python
def delete_record():
    st.subheader("Delete a Record")
    StudentID = st.text_input("Enter Student ID to delete")
    if st.button("Delete"):
        try:
            sql = "DELETE FROM Student WHERE StudentID=%s"
            val = (StudentID,)
            mycursor.execute(sql, val)
            mydb.commit()
            st.success("Record Deleted Successfully!!!")
        except mysql.connector.Error as err:
            st.error(f"Error: {err}")
```




```python
def delete_record():
    st.subheader("Delete a Record")
    DepartmentID = st.text_input("Enter DepartmentID of the record to delete")

    if st.button("Delete"):
        try:
            # Modify the SQL query to delete the record with the specified DepartmentID
            sql = "DELETE FROM department WHERE DepartmentID = %s"
            val = (DepartmentID,)

            mycursor.execute(sql, val)
            mydb.commit()
            st.success("Record Deleted Successfully!!!")
        except mysql.connector.Error as err:
            st.error(f"Error: {err}")
```

JOIN QUERY

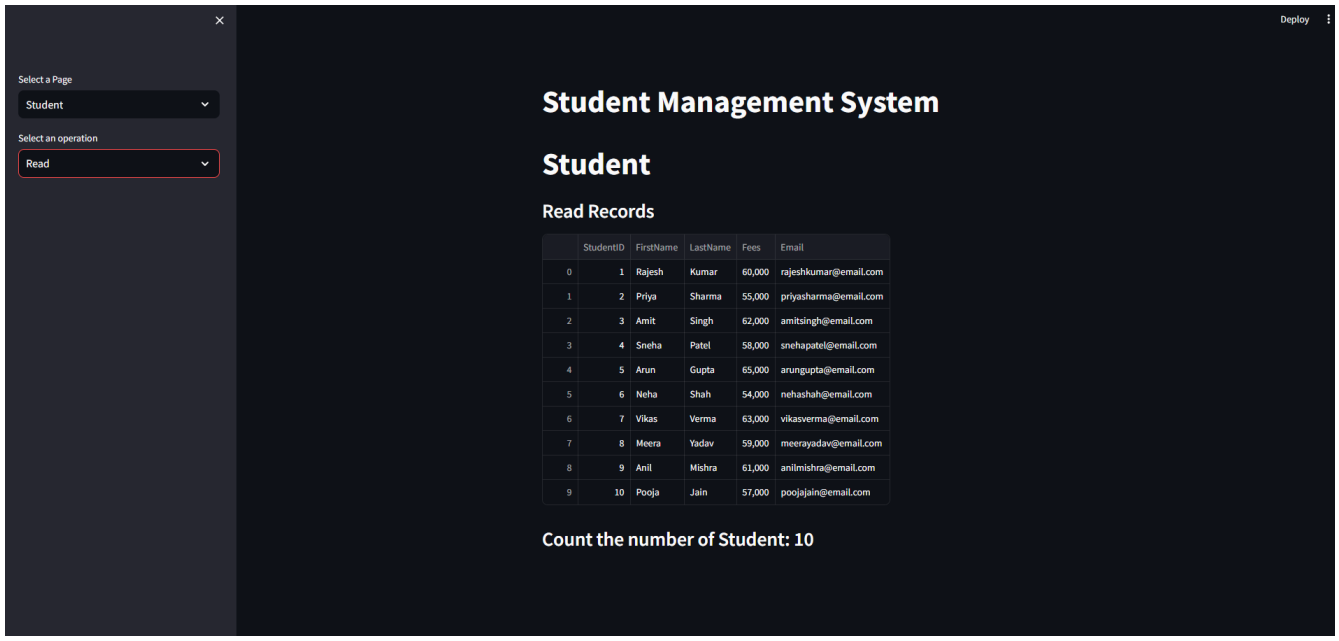
To join two tables with same StudentID using inner join




```python
def join_tables():
    st.title("Join Tables")

    # Let the user choose the tables to join
    table1_name = st.selectbox("Select the first table", ["Student", "attendance", "leaverecords"])
    table2_name = st.selectbox("Select the second table", ["Student", "attendance", "leaverecords"])

    if st.button("Join Tables"):
        try:
            # Modify the SQL query to join the selected tables based on StudentID
            sql = f"SELECT {table1_name}.*, {table2_name}.StartDate AS {table2_name}_StartDate,
            {table2_name}.EndDate AS {table2_name}_EndDate, {table2_name}.Status AS {table2_name}_Status FROM {table1_name} INNER JOIN {table2_name} ON {table1_name}.StudentID = {table2_name}.StudentID"

            mycursor.execute(sql)
            result = mycursor.fetchall()

            if result:
                columns = [desc[0] for desc in mycursor.description]
                df = pd.DataFrame(result, columns=columns)
                st.dataframe(df)
            else:
                st.write("No records found.")
        except mysql.connector.Error as err:
            st.error(f"Error: {err}")
```

COUNT FUNCTION

This function counts the total number of records

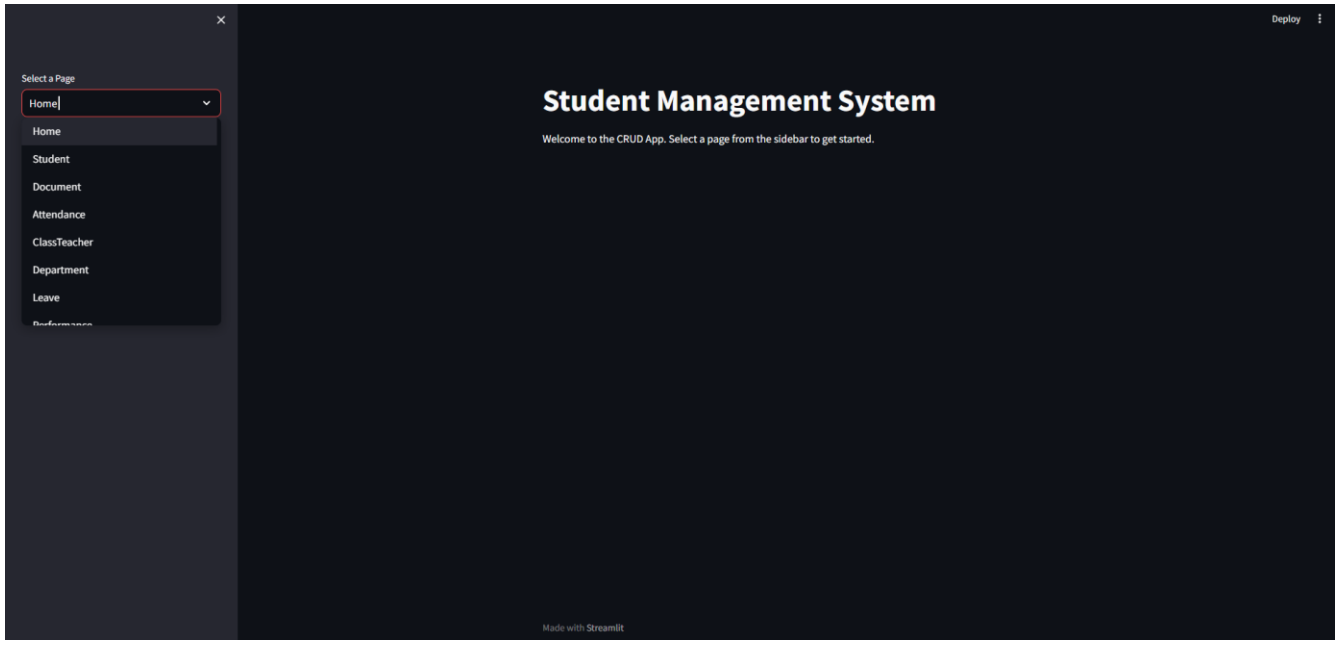


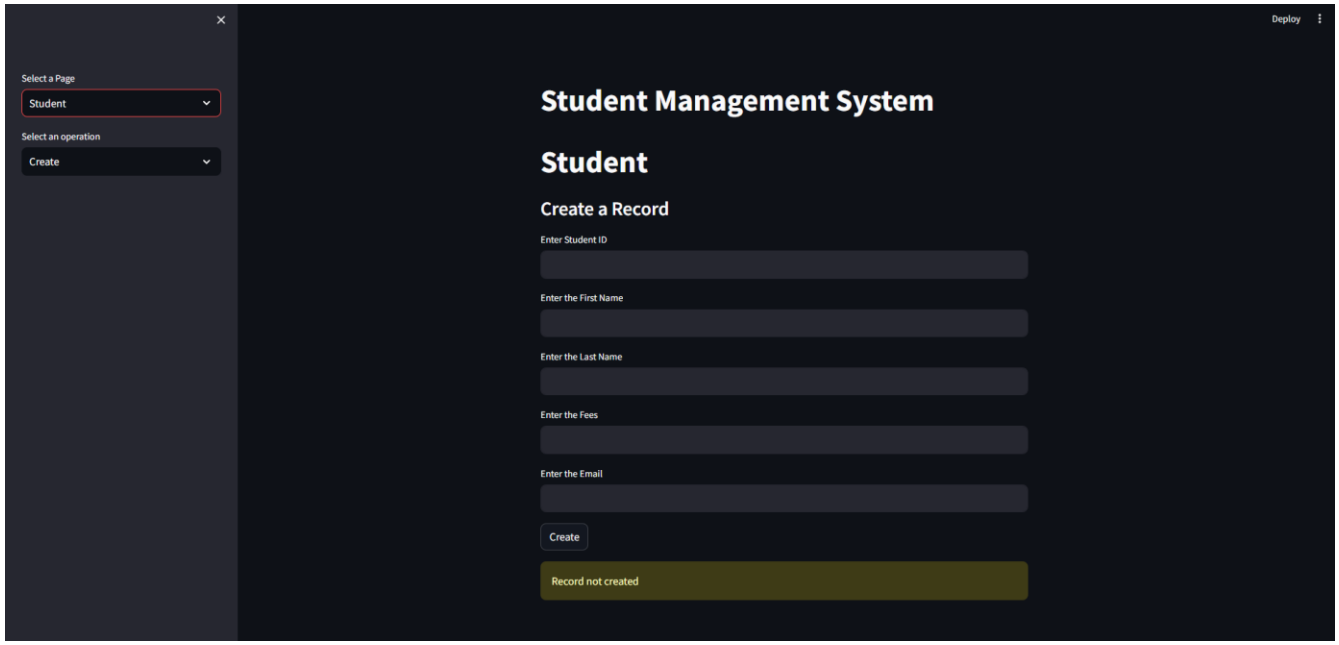
	StudentID	FirstName	LastName	Fees	Email
0	1	Rajesh	Kumar	60,000	rajeshkumar@email.com
1	2	Priya	Sharma	55,000	priyasharma@email.com
2	3	Amit	Singh	62,000	amitsingh@email.com
3	4	Sneha	Patel	58,000	snehapatel@email.com
4	5	Arun	Gupta	65,000	arungupta@email.com
5	6	Neha	Shah	54,000	nehashah@email.com
6	7	Vikas	Verma	63,000	vikasverma@email.com
7	8	Meera	Yadav	59,000	meerayadav@email.com
8	9	Anil	Mishra	61,000	anilmishra@email.com
9	10	Pooja	Jain	57,000	poojajain@email.com

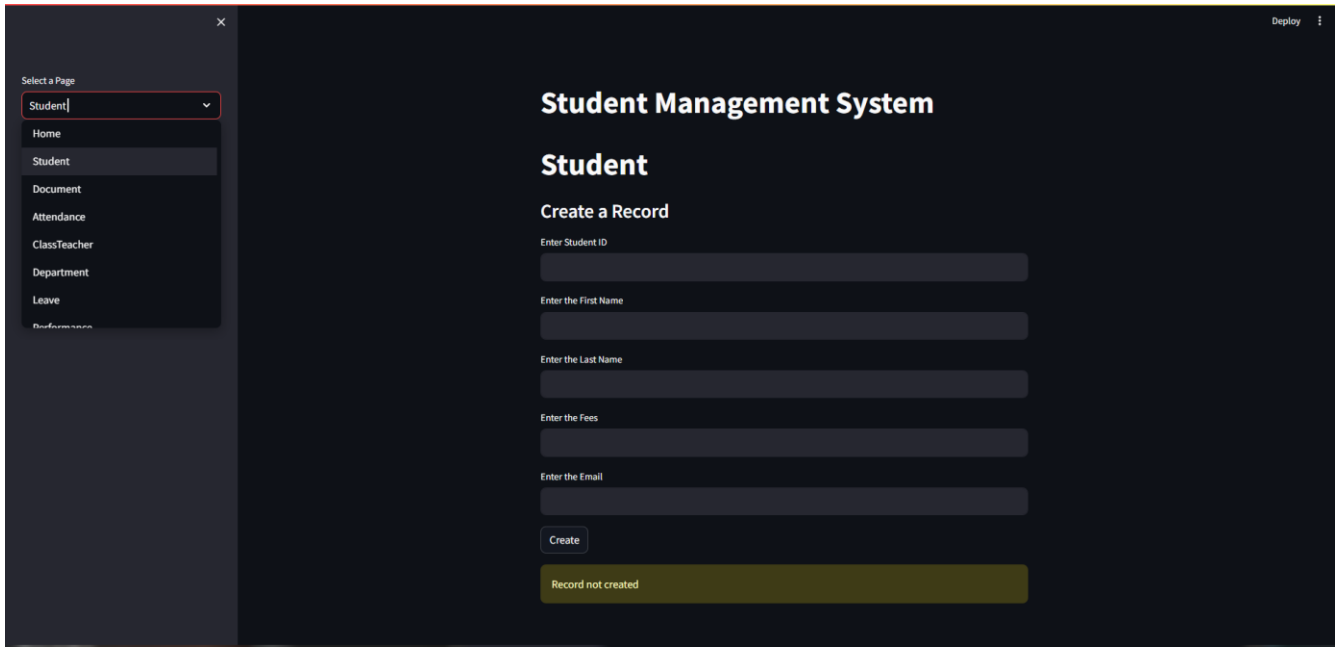
TRIGGERS

```
# Create triggers
triggers_sql = """
-- Create a table to log changes
CREATE TABLE IF NOT EXISTS Student_Log (
    LogID INT AUTO_INCREMENT PRIMARY KEY,
    Action VARCHAR(10),
    StudentID INT,
    FirstName VARCHAR(255),
    LastName VARCHAR(255),
    Fees DECIMAL(10,2),
    Email VARCHAR(255),
    Timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Create a trigger for INSERT operation
DELIMITER //
CREATE TRIGGER after_student_insert
AFTER INSERT ON Student
FOR EACH ROW
BEGIN
    INSERT INTO Student_Log (Action, StudentID, FirstName, LastName, Fees, Email)
    VALUES ('INSERT', NEW.StudentID, NEW.firstname, NEW.lastname, NEW.Fees, NEW.email);
END;
//

-- Create a trigger for UPDATE operation
DELIMITER //
CREATE TRIGGER after_student_update
AFTER UPDATE ON Student
FOR EACH ROW
BEGIN
    INSERT INTO Student_Log (Action, StudentID, FirstName, LastName, Fees, Email)
    VALUES ('UPDATE', NEW.StudentID, NEW.firstname, NEW.lastname, NEW.Fees, NEW.email);
END;
//

-- Create a trigger for DELETE operation
DELIMITER //
CREATE TRIGGER after_student_delete
AFTER DELETE ON Student
FOR EACH ROW
BEGIN
    INSERT INTO Student_Log (Action, StudentID, FirstName, LastName, Fees, Email)
    VALUES ('DELETE', OLD.StudentID, OLD.firstname, OLD.lastname, OLD.Fees, OLD.email);
END;
//
DELIMITER ;
"""

mycursor.execute(triggers_sql)
mydb.commit()
```

FRONT END DEVELOPEMNT






Deploy
Select a Page
Student
Home
Student
Document
Attendance
ClassTeacher
Department
Leave
Student Management System
Student
Create a Record
Enter Student ID
Enter the First Name
Enter the Last Name
Enter the Fees
Enter the Email
Create
Record not created

REFERENCES

1. www.youtube.com
2. www.github.com