Python Based Employee Attendance Monitoring With MySQL Integration

Group 10:

Bergas Ahmad Ardiansyah (2320010123)

Muhammad Ammar Abdullah (2320010151)

Faculty:

Ivan Firdaus, S.T

Class:

3CS1

CEP CCIT FACULTY OF ENGINEERING

UNIVERSITY OF INDONESIA

2024

PROJECT INFORMATION

Project Title : Python Based Employee Attendance Monitoring With MySQL Integration

Batch Code : 3CS1

Start Date : September 10th, 2024

End Date : September 29th, 2024

Name of Faculty : Ivan Firdaus, S.T

Names of Developers:

1. Bergas Ahmad Ardiannsyah
2. Muhammad Ammar Abdullah

Date of Submission: October 1st, 2024

ACKNOWLEDGEMENT

The authors would like to express their gratitude to Allah, the Most Merciful, for His guidance and blessings, which have enabled them to complete Project 3, titled "Python-Based Employee Attendance Monitoring With MySQL Integration." In addition, the authors wish to thank Mr. Ivan Firdaus, ST, for his insightful suggestions and continuous support throughout the writing of this project. His input has been invaluable in helping the authors enhance the quality of their work.

This paper provides an in-depth explanation of configuring a database in Python and integrating it with MySQL. The content also serves as a helpful resource for students to better understand the principles of Object-Oriented Programming, especially in the context of this third-semester project. Through this work, readers are expected to gain practical knowledge in developing database applications using Python.

Depok, 29th September 2024

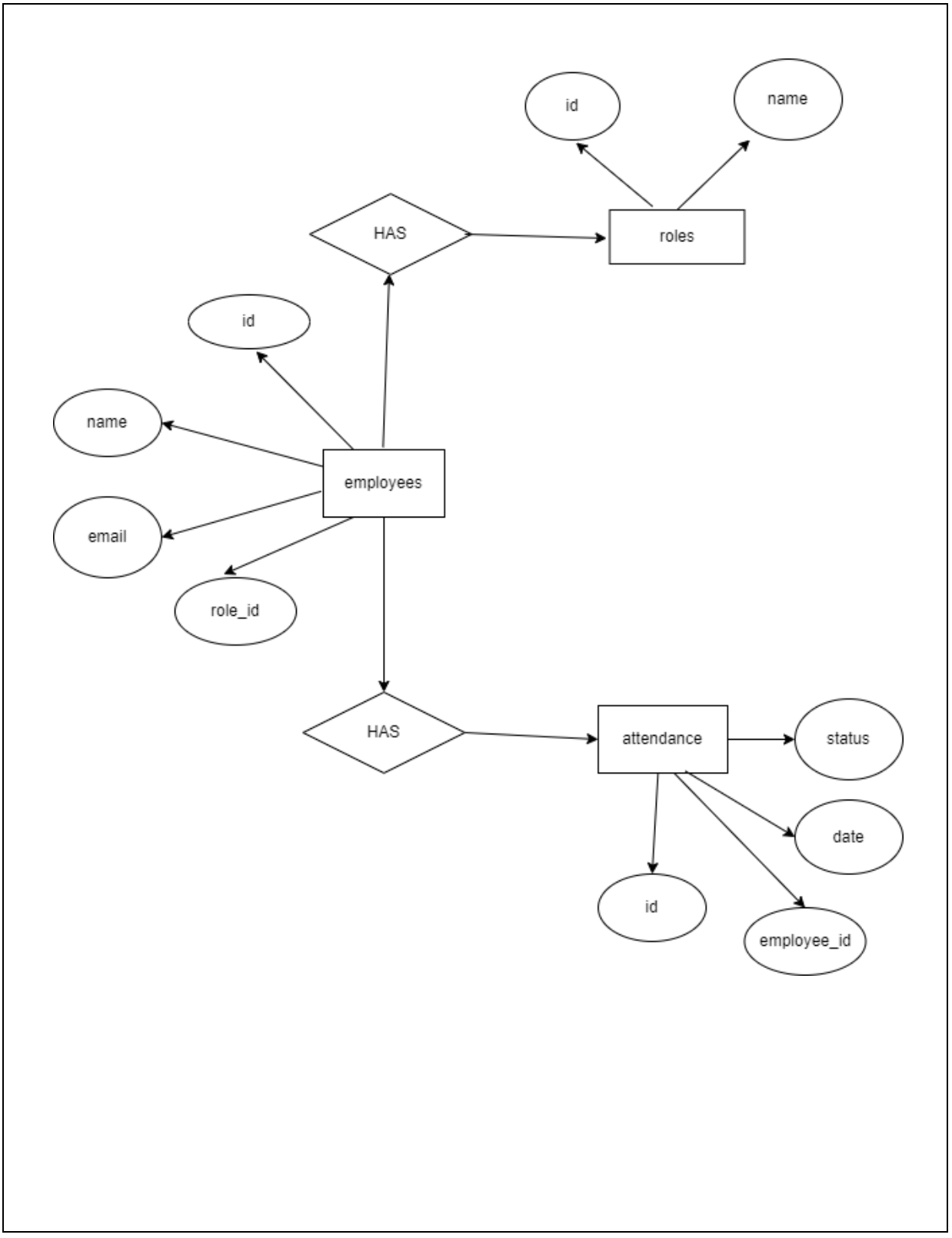
Authors

SYSTEM ANALYSIS

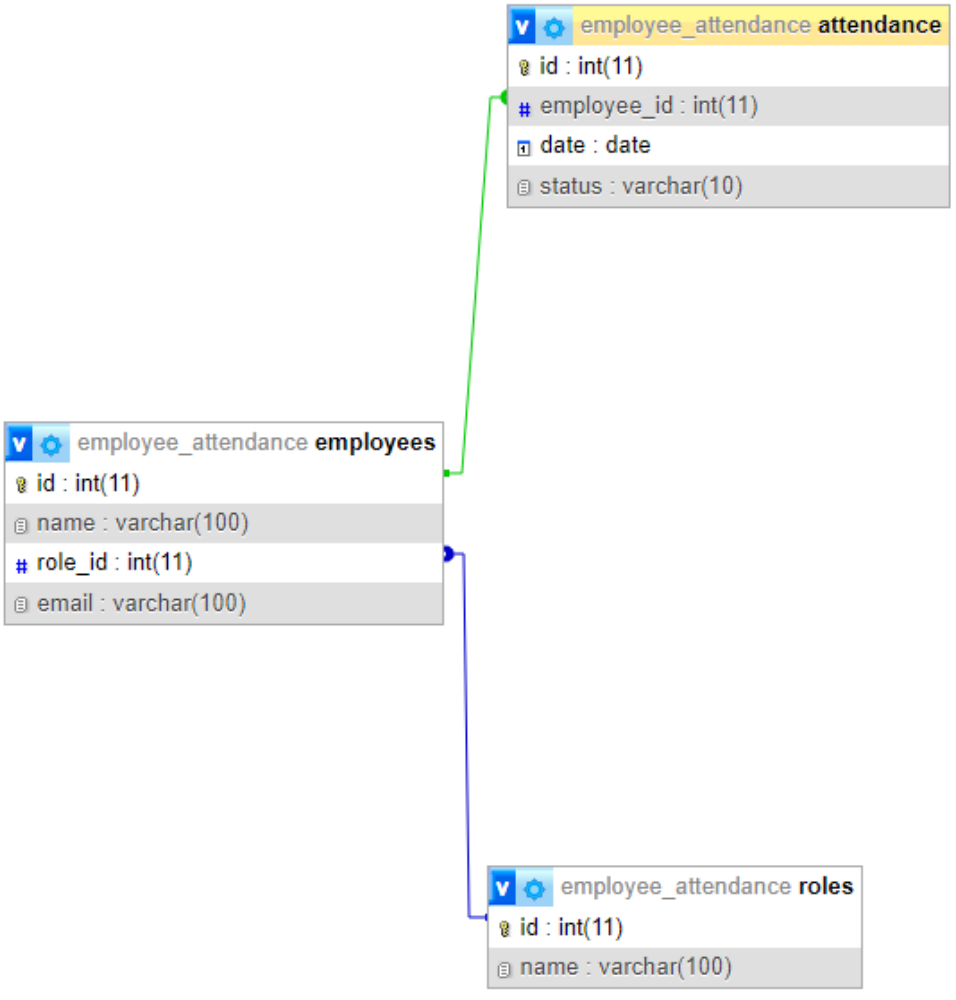
For this system, the authors developed a MySQL database integrated with Python using Visual Studio Code. The database created revolves around a Restaurant System Management, which consists of three tables: Menu Category, Main Menu, and Sales. This project demonstrates how Python can be used to interact with a MySQL database to effectively manage restaurant data.

The benefit of this project lies in the ability to both create and modify the MySQL database using Python. By working on this project, the authors gained practical experience in developing a database system and learned how to manage it programmatically, which is highly applicable for real-world use cases.

ERD


id
name
HAS
roles
id
name
employees
email
role_id
HAS
attendance
status
date
id
employee_id

DATABASE DIAGRAM



TABLES

1. employee Table :

Column Name	Data Type	Size	Description	Status
id	INT	Auto Increment	Unique identity number	PK
name	VARCHAR	100	Employee's name	-
department_id	INT	-	Unique identity number	FK (departments.id)
email	VARCHAR	100	Employee's email	

2. attendance Table :

Column Name	Data Type	Size	Description	Status
id	INT	Auto Increment	Unique identity number	PK
employee_id	INT	-	Unique identity number	FK (employee.id)
date	DATE	-	Attendance's date	-
status	INT	-	Attendance's status	-

3. departments Table :

Column Name	Data Type	Size	Description	Status
id	INT	Auto Increment	Unique identity number	PK
name	INT	-	Department's name	-

AD-AD CONFIGURATION

1. employee Table :

Validation Requires	Validation Perfomance
id : No user input needed since it is auto-generated, ensuring uniqueness and non-null values.	id : The **auto increment** feature guarantees uniqueness and non-null values, enforced by the primary key constraint.
departmens_id : Must be a INT and not null as it references another table (foreign key employee.id).	Id : A foreign key constraint (FK) ensures it references a valid category ID in the departments.id table.

2. attendance Table :

Validation Requires	Validation Perfomance
id : No user input needed since it is auto-generated, ensuring uniqueness and non-null values..	id: The **auto increment** feature guarantees uniqueness and non-null values, enforced by the primary key constraint.
employee_id : Must be a INT and not null as it references another table (foreign key employee.id).	id_category : A foreign key constraint (FK) ensures it references a valid category ID in the employee.id table.

3. departments Table :

Validation Requires	Validation Perfomance
id: No user input needed since it is auto-generated, ensuring uniqueness and non-null values..	id : The **auto increment** feature guarantees uniqueness and non-null values, enforced by the primary key constraint.

TYPESCRIPT

Query Creating Database in MYSQL

```
CREATE DATABASE employee_attendance;

USE employee_attendance;

CREATE TABLE departments (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL
);

CREATE TABLE employees (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    department_id INT,
    email VARCHAR(100) NOT NULL,
    FOREIGN KEY (department_id) REFERENCES departments(id)
);

CREATE TABLE attendance (
    id INT AUTO_INCREMENT PRIMARY KEY,
    employee_id INT,
    date DATE,
    status VARCHAR(10),
    FOREIGN KEY (employee_id) REFERENCES employees(id)
);
```

TYPESCRIPT

Connecting Python to Database MySQL

```
import mysql.connector
from datetime import datetime

# Connection to MySQL database
def get_connection():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        database="employee_attendance"
    )
```

TYPESCRIPT

CRUD Function Script in Pytthon

```
# Function to add an employee (Create)
def add_employee(name, role_id, email):
    conn = get_connection()
    cursor = conn.cursor()
    query = "INSERT INTO employees (name, role_id, email) VALUES (%s, %s, %s)"
    cursor.execute(query, (name, role_id, email))
    conn.commit()
    conn.close()
    print("Employee successfully added.")


# Function to list all employees (Read)
def list_employees():
    conn = get_connection()
    cursor = conn.cursor()
    cursor.execute("SELECT employees.id, employees.name, roles.name, employees.email FROM employees
JOIN roles ON employees.role_id = roles.id")
    result = cursor.fetchall()
    conn.close()
    for row in result:
        print(f"ID: {row[0]}, Name: {row[1]}, Role: {row[2]}, Email: {row[3]}")

# Function to update employee data (Update)
def update_employee(employee_id, name=None, role_id=None, email=None):
    conn = get_connection()
    cursor = conn.cursor()
    updates = []
    params = []

    if name:
        updates.append("name=%s")
        params.append(name)
    if role_id:
        updates.append("role_id=%s")
        params.append(role_id)
    if email:
        updates.append("email=%s")
        params.append(email)

    params.append(employee_id)

    query = f"UPDATE employees SET {', '.join(updates)} WHERE id=%s"
    cursor.execute(query, params)
    conn.commit()
    conn.close()
    print("Employee data successfully updated.")
```

TYPESCRIPT

CRUD Function Script in Pytthon

```
# Function to get employee details by ID (Read)
def get_employee_by_id(employee_id):
    conn = get_connection()
    cursor = conn.cursor()
    query = """
        SELECT employees.id, employees.name, roles.name, employees.email
        FROM employees
        JOIN roles ON employees.role_id = roles.id
        WHERE employees.id = %s
    """
    cursor.execute(query, (employee_id,))
    result = cursor.fetchone()
    conn.close()

    if result:
        print(f"\n--- Employee Details ID {employee_id} ---")
        print(f"ID: {result[0]}")
        print(f"Name: {result[1]}")
        print(f"Role: {result[2]}")
        print(f"Email: {result[3]}")
    else:
        print(f"Employee with ID {employee_id} not found.")

# Function to delete an employee (Delete)
def delete_employee(employee_id):
    conn = get_connection()
    cursor = conn.cursor()
    cursor.execute("DELETE FROM employees WHERE id=%s", (employee_id,))
    conn.commit()
    conn.close()
    print("Employee successfully deleted.")

# Function to add check-in attendance (Create)
def add_attendance_in(employee_id):
    conn = get_connection()
    cursor = conn.cursor()
    date_today = datetime.now().date()
    time_in = datetime.now().time()
    query = "INSERT INTO attendance (employee_id, date, status) VALUES (%s, %s, %s)"
    cursor.execute(query, (employee_id, date_today, f"Check-in: {time_in}"))
    conn.commit()
    conn.close()
    print("Check-in attendance successfully added.")
```

TYPESCRIPT

CRUD Function Script in Pytthon

```
# Function to add check-out attendance (Create)
def add_attendance_out(employee_id):
    conn = get_connection()
    cursor = conn.cursor()
    date_today = datetime.now().date()
    time_out = datetime.now().time()
    query = "INSERT INTO attendance (employee_id, date, status) VALUES (%s, %s, %s)"
    cursor.execute(query, (employee_id, date_today, f"Check-out: {time_out}"))
    conn.commit()
    conn.close()
    print("Check-out attendance successfully added.")

# Function to list attendance for an employee (Read)
def list_attendance(employee_id):
    conn = get_connection()
    cursor = conn.cursor()
    query = "SELECT date, status FROM attendance WHERE employee_id=%s"
    cursor.execute(query, (employee_id,))
    result = cursor.fetchall()
    conn.close()
    for row in result:
        print(f"Date: {row[0]}, Status: {row[1]}")

# Function to choose a role (Read)
def choose_role():
    roles = ["Director", "Manager", "Marketing", "Employee"]
    for i, role in enumerate(roles, 1):
        print(f"{i}. {role}")
    choice = int(input("Choose Role (1-4): "))
    if choice in range(1, 5):
        return choice
    else:
        print("Invalid choice. Please try again.")
        return choose_role()
```

TYPESCRIPT

Menu Function Script in Pytthon

```
# Employee Main  Menu
def main_menu():
    while True:
        print("\n=== Employee Attendance System ===")
        print("1. Employee Administration")
        print("2. Attendance")
        print("99. Exit")
        choice = input("Choose an option: ")

        if choice == "1":
            employee_admin_menu()
        elif choice == "2":
            attendance_menu()
        elif choice == "99":
            print("Exiting the program...")
            break
        else:
            print("Invalid choice. Please try again.")

# Attendance Menu
def attendance_menu():
    while True:
        print("\n--- Attendance Menu ---")
        print("1. Add Check-in")
        print("2. Add Check-out")
        print("3. View Attendance")
        print("99. Return to Main Menu")
        choice = input("Choose an option: ")

        if choice == "1":
            employee_id = input("Employee ID: ")
            add_attendance_in(employee_id)
        elif choice == "2":
            employee_id = input("Employee ID: ")
            add_attendance_out(employee_id)
        elif choice == "3":
            employee_id = input("Employee ID: ")
            list_attendance(employee_id)
        elif choice == "99":
            return
        else:
            print("Invalid choice. Please try again.")
```

TYPESCRIPT

Menu Function Script in Pytthon

```
# Employee Administration Menu
def employee_admin_menu():
    while True:
        print("\n--- Employee Administration Menu ---")
        print("1. View All Employees")
        print("2. View Employee by ID")
        print("3. Update Employee")
        print("4. Delete Employee")
        print("99. Return to Main Menu")
        choice = input("Choose an option: ")

        if choice == "1":
            list_employees()
        elif choice == "2":
            employee_id = input("Enter Employee ID: ")
            get_employee_by_id(employee_id)
        elif choice == "3":
            employee_id = input("Employee ID: ")
            name = input("New name (leave blank if not changing): ")
            print("Choose new role (leave blank if not changing):")
            role_id = choose_role()
            email = input("New email (leave blank if not changing): ")
            update_employee(employee_id, name or None, role_id, email or None)
        elif choice == "4":
            employee_id = input("Employee ID: ")
            delete_employee(employee_id)
        elif choice == "99":
            return
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main_menu()
```

Hardware Configuration

Hardware : Asus TUF F15 & HP Pavilion G15

OS : WINDOWS 11

Software :Visual Code Studio, XAMPP MySQL, Python

PROJECT FILE DETAILS

NO	File Name	Remarks
1	.docx	Bergas A. Ardiansyah
2	.pptx	Muhammad Ammar Abdullah