WorkTrack: Employee Timesheet Tracker
Database Project Documentation

By

FRANZ LLOYD A. DIAZ

ARJONEL M. MENDOZA, MIT
Lecturer

PROJECT OVERVIEW

The Employee Timesheet Tracker is a system designed to record and manage employee work hours by tracking their clock-in and clock-out times. It aims to streamline attendance management, improve productivity monitoring, and ensure accurate payroll processing. This system was created by using Python, CustomTkinter, and MySQL.

ENTITY-RELATIONSHIP DIAGRAM (ERD)

The ERD represents key entities in the Employee Timesheet Tracker such as employees and their attendance. Each entity has attributes essential for data management.

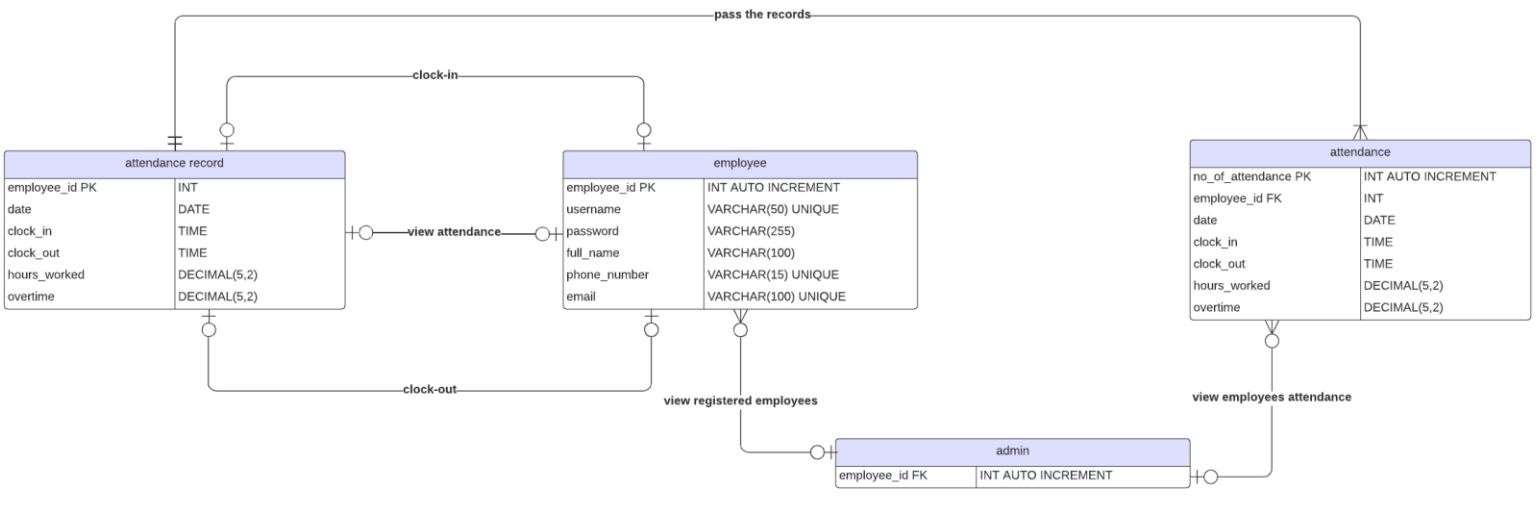

Figure 1. Entity Relationship Diagram

Entities and their Relationships

1. **Employee & attendance record:**

- **Type**: One-to-One (1:1)
- **Description**: An employee can clock-in once per day in • his/her attendance. Attendance record can be clocked in per employee once in a day. An employee can clock-out once per day in his/her attendance if and only if the employee clocked in. Attendance record can be clocked

out per employee once in a day. An employee can view his/her attendance. Attendance record can be viewed per employee. An employee can view his/her attendance. Attendance record can be viewed per employee.

2. **attendance record & attendance:**

   - **Type:** One-to-Many (1:N)
   - **Description:** Attendance record must pass the records to attendance. Attendance can accept many attendance records.

3. **admin & attendance:**

   - **Type:** One-to-One (1:N)
   - **Description:** Admin can view the attendance of all employees. Attendance can be viewed by one and only one admin.

4. **admin & employee:**

   - **Type:** One-to-One (1:N)
   - **Description:** Admin can view all registered employees. Employees can be viewed by one and only one admin.

SQL SCRIPTS

```
-- Create Database
CREATE DATABASE timesheet;

-- Use Database
USE timesheet;

-- Create Tables
CREATE TABLE employee (
    employee_id INT(11) PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    password VARCHAR(50) NOT NULL,
    full_name VARCHAR(255) NOT NULL,
    phone_number VARCHAR(15) NOT NULL,
    email VARCHAR(100) NOT NULL,
);
```

```
MariaDB [timesheet]> describe employees;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| employee_id  | int(11)      | NO   | PRI | NULL    | auto_increment |
| username     | varchar(50)  | NO   |     | NULL    |                |
| password     | varchar(255) | NO   |     | NULL    |                |
| full_name    | varchar(100) | NO   |     | NULL    |                |
| phone_number | varchar(15)  | NO   | MUL | NULL    |                |
| email        | varchar(100) | NO   |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
6 rows in set (0.012 sec)
```

```
CREATE TABLE attendance (
    no_of_attendance INT(100) AUTO INCREMENT PRIMARY KEY,
    date DATE NOT NULL,
    clock_in TIME NOT NULL,
    clock_out TIME NOT NULL,
    hours_worked DECIMAL(5,2) NOT NULL,
    overtime DECIMAL(5,2) NOT NULL
);
```

```
MariaDB [timesheet]> describe attendance;
+------------------+--------------+------+-----+---------+----------------+
| Field            | Type         | Null | Key | Default | Extra          |
+------------------+--------------+------+-----+---------+----------------+
| no_of_attendance | int(100)     | NO   | PRI | NULL    | auto_increment |
| employee_id      | int(11)      | NO   | MUL | NULL    |                |
| date             | date         | NO   |     | NULL    |                |
| clock_in         | time         | NO   |     | NULL    |                |
| clock_out        | time         | NO   |     | NULL    |                |
| hours_worked     | decimal(5,2) | NO   |     | NULL    |                |
| overtime         | decimal(5,2) | NO   |     | NULL    |                |
+------------------+--------------+------+-----+---------+----------------+
7 rows in set (0.013 sec)
```

‘attendance’ table records:

```
MariaDB [timesheet]> select * from attendance
    -> ;
+------------------+-------------+------------+----------+-----------+--------------+----------+
| no_of_attendance | employee_id | date       | clock_in | clock_out | hours_worked | overtime |
+------------------+-------------+------------+----------+-----------+--------------+----------+
|                1 |           6 | 2024-12-01 | 15:04:44 | 15:04:45  |         0.00 |     0.00 |
|                2 |           1 | 2024-12-01 | 15:04:49 | 15:04:50  |         0.00 |     0.00 |
|                3 |           2 | 2024-12-01 | 15:04:55 | 15:04:56  |         0.00 |     0.00 |
|                4 |           7 | 2024-12-01 | 15:05:01 | 15:05:02  |         0.00 |     0.00 |
|                5 |           3 | 2024-12-01 | 15:05:06 | 15:05:07  |         0.00 |     0.00 |
|                6 |           4 | 2024-12-01 | 15:05:11 | 15:05:12  |         0.00 |     0.00 |
|                7 |          10 | 2024-12-01 | 15:07:11 | 15:07:27  |         0.00 |     0.00 |
|                8 |          11 | 2024-12-01 | 15:07:16 | 15:07:17  |         0.00 |     0.00 |
|                9 |           8 | 2024-12-01 | 15:07:21 | 15:07:22  |         0.00 |     0.00 |
+------------------+-------------+------------+----------+-----------+--------------+----------+
9 rows in set (0.000 sec)
```

‘employees’ table records:

```
MariaDB [timesheet]> select * from employees;
+-------------+----------+----------+-----------------------+--------------+--------------------+
| employee_id | username | password | full_name             | phone_number | email              |
+-------------+----------+----------+-----------------------+--------------+--------------------+
|           1 | franz123 | 123      | Franz Lloyd Diaz      | 09053895203  | franz123@gmail.com |
|           2 | lance123 | 123      | Lance Edward Dela Rosa | 094285723845 | lance123@gmail.com |
|           3 | cj123    | 123      | Cristian Joshua Javier | 09485723485  | cj123@gmail.com    |
|           4 | brent123 | 123      | Brent Draniel Aclan   | 09862348592  | brent123@gmail.com |
|           5 | nica123  | 123      | Eunica De Villa       | 09582384293  | nica123@gmail.com  |
|           6 | test1    | 123      | test1                 | 09832748293  | test1@gmail.com    |
|           7 | test2    | 123      | test2                 | 0942847283   | test2@gmail.com    |
|           8 | test3    | 123      | test3                 | 09242982984  | test3@gmail.com    |
|           9 | test4    | 123      | test4                 | 0955984934   | test4@gmail.com    |
|          10 | test5    | 123      | test5                 | 09382787242  | test5@gmail.com    |
|          11 | test6    | 123      | test6                 | 09482948298  | test6@gmail.com    |
|          12 | test7    | 123      | test7                 | 09489724293  | test7@gmail.com    |
+-------------+----------+----------+-----------------------+--------------+--------------------+
12 rows in set (0.000 sec)
```

CODE SNIPPETS

connection of MySQL & python:

```python
import customtkinter as ctk
from tkinter import ttk  # Import the ttk module for Treeview
from tkinter import messagebox
from PIL import Image
import mysql.connector
import datetime


# Database Connection
def get_db_connection():
    return mysql.connector.connect(
        host="localhost",
        user="root",          # Replace with your MySQL username
        password="",          # Replace with your MySQL password
        database="timesheet"  # Database name
    )
```

authentication of employee login:

```python
# Employee Authentication
def authenticate_employee(username, password):
    try:
        conn = get_db_connection()
        cursor = conn.cursor()

        # Query to check username and password
        cursor.execute("SELECT employee_id, full_name FROM employees WHERE username = %s AND password = %s", (username, password))
        record = cursor.fetchone()

        conn.close()
        return record if record else None  # Return record (tuple) if found, otherwise None
    except Exception as e:
        print(f"Error during authentication: {e}")
        return None
```

clock-in function:

```python
# Clock In Function
def clock_in(employee_id):
    conn = get_db_connection()
    cursor = conn.cursor()

    today = datetime.date.today().strftime('%Y-%m-%d')
    now = datetime.datetime.now().strftime('%H:%M:%S')

    # Check if the employee has already clocked in today
    cursor.execute("SELECT * FROM attendance WHERE employee_id = %s AND date = %s", (employee_id, today))
    record = cursor.fetchone()

    if record:
        # If there's a record, check if the employee has clocked out already
        if record[3] is not None:  # Check if clock_out is not None (indicating clocked out)
            return "You have already clocked in today."

    else:
        cursor.execute("INSERT INTO attendance (employee_id, date, clock_in) VALUES (%s, %s, %s)", (employee_id, today, now))
        conn.commit()
        return f"Clocked in at {now}"
```

clock-out function:

```python
# Clock Out Function
def clock_out(employee_id):
    conn = get_db_connection()
    cursor = conn.cursor()

    today = datetime.date.today().strftime('%Y-%m-%d')
    now = datetime.datetime.now().strftime('%H:%M:%S')

    # Fetch clock-in time
    cursor.execute("SELECT clock_in FROM attendance WHERE employee_id = %s AND date = %s", (employee_id, today))
    record = cursor.fetchone()

    if not record:
        return "You have not clocked in today. Cannot clock out."

    clock_in_time = datetime.datetime.strptime(str(record[0]), '%H:%M:%S')
    clock_out_time = datetime.datetime.strptime(now, '%H:%M:%S')

    # Calculate total hours worked and overtime
    total_hours = (clock_out_time - clock_in_time).total_seconds() / 3600
    overtime = max(0, total_hours - 8)

    cursor.execute("""
        UPDATE attendance
        SET clock_out = %s, hours_worked = %s, overtime = %s
        WHERE employee_id = %s AND date = %s
    """, (now, total_hours, overtime, employee_id, today))
    conn.commit()
    conn.close()

    return f"Clocked out at {now}. Total hours: {total_hours:.2f}, Overtime: {overtime:.2f}"
```

get the attendance of the logged-in employee

```python
# Fetch Attendance Records
def get_attendance(employee_id):
    conn = get_db_connection()
    cursor = conn.cursor()

    # Fetch attendance records for the logged-in employee
    cursor.execute("""
        SELECT date, clock_in, clock_out, hours_worked, overtime
        FROM attendance
        WHERE employee_id = %s
        ORDER BY date DESC
    """, (employee_id,))
    records = cursor.fetchall()

    conn.close()
    return records
```

register employee

```python
def register_user(self):
    username = self.new_username_entry.get()
    password = self.new_password_entry.get()
    full_name = self.full_name_entry.get()
    phone_number = self.phone_number_entry.get()
    email = self.email_entry.get()

    if not username or not password or not full_name or not phone_number or not email:
        messagebox.showwarning("Invalid Input", "All fields are required.")
        return

    # Check if the username, phone_number, or email already exists in the database
    conn = get_db_connection()
    cursor = conn.cursor()

    # Check if username already exists
    cursor.execute("SELECT * FROM employees WHERE username = %s", (username,))
    if cursor.fetchone():
        messagebox.showwarning("Username Taken", "Username already exists.")
        return
    # Check if phone number already exists
    cursor.execute("SELECT * FROM employees WHERE phone_number = %s", (phone_number,))
    if cursor.fetchone():
        messagebox.showwarning("Phone Number Taken", "Phone number already exists.")
        return

    # Check if email already exists
    cursor.execute("SELECT * FROM employees WHERE email = %s", (email,))
    if cursor.fetchone():
        messagebox.showwarning("Email Taken", "Email already exists.")
        return

    # If no existing records, proceed to register the user
    try:
        cursor.execute("INSERT INTO employees (username, password, full_name, phone_number, email) VALUES (%s, %s, %s, %s, %s)",
                       (username, password, full_name, phone_number, email))
        conn.commit()

        messagebox.showinfo("Result", "Registration Successful.")

        self.show_login_screen()  # Go back to login screen after registration
    except Exception as e:
        messagebox.showerror("Error", f"Error: {e}")
    finally:
        conn.close()
```

in admin mode, show all attendance records:

```
# Fetch and display all attendance records
conn = get_db_connection()
cursor = conn.cursor()
cursor.execute("""SELECT employee_id, date, clock_in, clock_out, hours_worked, overtime FROM attendance ORDER BY date DESC""")
records = cursor.fetchall()
conn.close()
```

in admin mode, show all registered employees:

```
# Fetch and display all registered users
conn = get_db_connection()
cursor = conn.cursor()
cursor.execute("""SELECT employee_id, username, full_name, phone_number, email FROM employees ORDER BY employee_id""")
users = cursor.fetchall()
conn.close()
```