# Event Management Project

-Pranjali Datkhile

Project 1 Report

## Abstract

This project presents an Event Management System developed in Python, integrating CRUD operations with a MySQL database. The application allows users to create, view, delete events, manage attendees, and export data to CSV. This report details the design, functionality, database interactions, and code logic used in the project.

## Introduction

This project demonstrates the application of software engineering concepts like modular programming, database integration, and CRUD operations in Python. The CRUD model is the backbone of any data-centric application, enabling users to interact with databases in a structured and reliable manner.

This system uses MySQL as the backend database and Python for the application logic. Data from the system can be exported to CSV files, making it easier to back up or share information.

## System Overview

The Event Management System has the following main components:

* **Event Management:** Users can create events with details like name, date, location, and description.
* **Attendee Management:** Attendees can be registered to a particular event using their name and email.
* **Viewing Records:** All events and attendees can be viewed using menu options.
* **Deleting Events:** Events can be deleted using their unique IDs.
* **Export Functionality:** Events and attendees can be exported to CSV files for external use.
* **Error Handling:** Basic error checking and exception handling ensures the system is robust against invalid inputs.

## **Technologies Used**

|  |  |
| --- | --- |
| **Technology** | **Purpose** |
| **Python 3.x** | Core programming language |
| **MySQL** | Backend database for storing data |
| **MySQL Connector (Python)** | Connecting Python to MySQL |
| **CSV Module** | Exporting data to .csv files |
| **SQL** | Query language for CRUD operations |
| **Command Line Interface** | User interaction method |

## SQL Database Schema

CREATE TABLE Events (

event\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255),

date DATE,

location VARCHAR(255),

description TEXT

);

CREATE TABLE Attendees (

attendee\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255),

email VARCHAR(255),

event\_id INT,

FOREIGN KEY (event\_id) REFERENCES Events(event\_id)

);

## Code

### main.py

# main.py

from event\_manager import create\_event, get\_all\_events, delete\_event, export\_events\_to\_csv, export\_all\_attendees\_to\_csv

from attendee\_manager import add\_attendee, get\_attendees\_by\_event

def main():

print("=" \* 40)

print(" Welcome to Event Management System ")

print("=" \* 40)

while True:

try:

print("\nMenu:")

print("1. Create Event")

print("2. View Events")

print("3. Delete Event")

print("4. Add Attendee")

print("5. View Attendees by Event")

print("6. Exit")

print("7. Export Events to CSV")

print("8. Export All Attendees to CSV")

choice = input("Enter your choice (1-8): ").strip()

if choice == '1':

print("\n--- Create Event ---")

name = input("Event Name: ").strip()

date = input("Event Date (YYYY-MM-DD): ").strip()

location = input("Location: ").strip()

description = input("Description: ").strip()

if name and date:

create\_event(name, date, location, description)

print(" Event added successfully.")

else:

print(" Event name and date are required.")

elif choice == '2':

print("\n--- All Events ---")

get\_all\_events()

elif choice == '3':

print("\n--- Delete Event ---")

try:

event\_id = int(input("Enter Event ID to delete: "))

delete\_event(event\_id)

print(" Event deleted.")

except ValueError:

print(" Invalid Event ID. Please enter a number.")

elif choice == '4':

print("\n--- Add Attendee ---")

name = input("Attendee Name: ").strip()

email = input("Email: ").strip()

try:

event\_id = int(input("Event ID to register for: "))

add\_attendee(name, email, event\_id)

print(" Attendee added.")

except ValueError:

print(" Invalid Event ID.")

elif choice == '5':

print("\n--- View Attendees ---")

try:

event\_id = int(input("Enter Event ID to view attendees: "))

get\_attendees\_by\_event(event\_id)

except ValueError:

print(" Invalid Event ID.")

elif choice == '6':

print(" Exiting program. Goodbye!")

break

elif choice == '7':

print("\n--- Export Events to CSV ---")

filename = input("Enter filename (e.g., events.csv): ").strip()

if filename:

export\_events\_to\_csv(filename)

print(f" Events exported to {filename}")

else:

print(" Filename cannot be empty.")

elif choice == '8':

print("\n--- Export All Attendees to CSV ---")

filename = input("Enter filename (e.g., attendees.csv): ").strip()

if filename:

export\_all\_attendees\_to\_csv(filename)

print(f" All attendees exported to {filename}")

else:

print(" Filename cannot be empty.")

else:

print(" Invalid choice. Please enter a number from 1 to 8.")

except Exception as e:

print(f"\n An error occurred: {e}\nPlease check your database and try again.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

### attendee\_manager.py

# attendee\_manager.py

from db\_connection import get\_connection

import csv

def add\_attendee(name, email, event\_id):

conn = get\_connection()

cursor = conn.cursor()

query = "INSERT INTO Attendees (name, email, event\_id) VALUES (%s, %s, %s)"

cursor.execute(query, (name, email, event\_id))

conn.commit()

cursor.close()

conn.close()

def get\_attendees\_by\_event(event\_id, return\_data=False):

conn = get\_connection()

cursor = conn.cursor(dictionary=True) # dictionary=True to get dict rows

query = "SELECT name, email FROM Attendees WHERE event\_id = %s"

cursor.execute(query, (event\_id,))

attendees = cursor.fetchall()

cursor.close()

conn.close()

if return\_data:

return attendees

if attendees:

for attendee in attendees:

print(attendee)

else:

print("No attendees found for this event.")

def export\_attendees\_to\_csv(event\_id, filename=None):

attendees = get\_attendees\_by\_event(event\_id, return\_data=True)

if not attendees:

print("No attendees to export for this event.")

return

# Default filename if not provided

if not filename:

filename = f"attendees\_export\_event\_{event\_id}.csv"

with open(filename, mode='w', newline='', encoding='utf-8') as file:

writer = csv.writer(file)

writer.writerow(["Name", "Email"])

for attendee in attendees:

writer.writerow([

attendee.get('name', 'N/A'), # Use 'N/A' if 'name' is missing

attendee.get('email', 'N/A'), # Use 'N/A' if 'email' is missing

])

print(f"Attendees for event {event\_id} successfully exported to {filename}")

### event\_manager.py

# event\_manager.py

from db\_connection import get\_connection

import csv

from attendee\_manager import get\_attendees\_by\_event # Import the attendee manager

def create\_event(name, date, location, description):

conn = get\_connection()

cursor = conn.cursor()

query = "INSERT INTO Events (name, date, location, description) VALUES (%s, %s, %s, %s)"

cursor.execute(query, (name, date, location, description))

conn.commit()

cursor.close()

conn.close()

def get\_all\_events(return\_data=False):

conn = get\_connection()

cursor = conn.cursor(dictionary=True) # dictionary=True to get dict results

query = "SELECT \* FROM Events"

cursor.execute(query)

events = cursor.fetchall()

cursor.close()

conn.close()

if return\_data:

return events

if events:

for event in events:

print(event)

else:

print("No events found.")

def delete\_event(event\_id):

conn = get\_connection()

cursor = conn.cursor()

query = "DELETE FROM Events WHERE event\_id = %s"

cursor.execute(query, (event\_id,))

conn.commit()

cursor.close()

conn.close()

# creating csv files!

def export\_events\_to\_csv(filename='events\_export.csv'):

events = get\_all\_events(return\_data=True)

if not events:

print("No events to export.")

return

with open(filename, mode='w', newline='', encoding='utf-8') as file:

writer = csv.writer(file)

writer.writerow(["ID", "Name", "Date", "Location", "Description"])

for event in events:

writer.writerow([

event.get('event\_id'),

event.get('name'),

event.get('date'),

event.get('location'),

event.get('description')

])

print(f"Events successfully exported to {filename}")

def export\_all\_attendees\_to\_csv(filename='attendees\_export.csv'):

conn = get\_connection()

cursor = conn.cursor(dictionary=True)

query = "SELECT event\_id, name FROM Events" # Getting event ID and Name to export with attendees

cursor.execute(query)

events = cursor.fetchall()

cursor.close()

conn.close()

if not events:

print("No events found to export attendees.")

return

with open(filename, mode='w', newline='', encoding='utf-8') as file:

writer = csv.writer(file)

writer.writerow(["Event ID", "Event Name", "Attendee Name", "Attendee Email"])

for event in events:

event\_id = event['event\_id']

event\_name = event['name']

print(f"Exporting attendees for event ID: {event\_id} - {event\_name}") # Debugging line

attendees = get\_attendees\_by\_event(event\_id, return\_data=True)

if attendees:

for attendee in attendees:

# Debugging: Print the attendee data to inspect

print(f"Attendee Data: {attendee}") # Debugging line

# Safely access name and email fields

name = attendee.get('name', 'N/A') # Use 'N/A' if 'name' is missing

email = attendee.get('email', 'N/A') # Use 'N/A' if 'email' is missing

writer.writerow([

event\_id,

event\_name,

name,

email

])

else:

print(f"No attendees found for event ID {event\_id}") # Debugging line

print(f"Attendees for all events successfully exported to {filename}")

### db\_config.py

# db\_config.py

config = {

'host': 'localhost',

'user': 'root',

'password': 'pass@word1', # replace with your actual password

'database': 'EventManagement'

}

### db\_connection.py

# db\_connection.py

import mysql.connector

from db\_config import config

def get\_connection():

return mysql.connector.connect(\*\*config)

## Output

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Conclusion

This project successfully demonstrates how CRUD operations can be applied to real-life applications such as event management. By integrating Python with a MySQL database and providing features to manage data and export it, the system ensures usability, reliability, and data portability.

## Future Enhancements

* **Update Functionality**: Allow users to edit existing event or attendee details.
* **Search Feature**: Search events or attendees using filters.
* **GUI Interface**: Develop a graphical interface using Tkinter or PyQt.
* **Web App Version**: Migrate system to a web-based solution using Flask or Django.
* **User Authentication**: Add login system to secure access.