Project Report: To-Do List Management System using Python and MySQL

Project Overview

The project is a Python-based command-line To-Do List Management System that uses a MySQL database to store tasks. It allows users to add, view, update, and delete tasks from their To-Do list. The project demonstrates the integration of Python and MySQL for database management using the pymysql library, focusing on basic CRUD (Create, Read, Update, Delete) operations.

Objectives

- To develop a simple To-Do List Management System.
- To provide an interface for users to interact with tasks: add new tasks, view existing tasks, update their status, and delete tasks.
- To demonstrate the use of MySQL databases in Python applications for persistent data storage.

System Architecture

Programming Language: Python

Database Management System: MySQL

Python Libraries: Pymysql: For MySQL database connectivity.

Implementation Details

Database Setup

The MySQL database is set up programmatically using pymysql. The following steps are followed to set up the database:

- ✓ Database Creation: A new MySQL database called todo_list_db is created if it doesn't exist.
- ✓ Table Creation: A table named todos is created with the following structure:

1) Id: A unique identifier for each task (Primary Key).
2) Task: A string representing the task description.
3) Status: An ENUM field to track task status (Pending or Completed).
4) Created_at: A timestamp indicating when the task was created.

CRUD Operations

The system implements the following operations using SQL queries:

- ✓ Add Task: A function to add a new task to the database if it doesn't already exist.
- ✓ View Tasks: A function to fetch and display all tasks stored in the database.
- ✓ Update Task: A function to update the status of a task (from 'Pending' to 'Completed').
- ✓ Delete Task: A function to remove a task from the database.

Main Menu

A text-based menu is provided to allow users to interact with the system. The menu options include:

1. Add a new task.

2. View all tasks.

3. Update the status of a task.

4. Delete a task.

5. Exit the program.

Code Structure

The code is organized into functions, with each function handling a specific operation:

1. Database Connection:

Pymysql.connect() is used to establish a connection with MySQL, first to create the database and then to connect to the todo_list_db.

2. Task Operations:

Add_task(task): Adds a new task to the table after checking if it already exists.

View_tasks(): Displays all tasks in the table along with their status and creation time.

Update_task(task_id, new_status): Updates the status of a task by task ID.

Delete_task(task_id): Deletes a task by task ID.

Main Menu:

The main_menu() function provides a loop where users can choose different operations until they exit the program.

Challenges and Solutions

1. **Database Connectivity Issues:**

The program might fail if MySQL server credentials or configurations are incorrect. To handle this, proper error handling could be introduced with exception handling (try-except blocks).

2. **Duplicate Task Entries:**

The add_task() function checks if a task already exists before inserting it to avoid duplicate entries.

3. **User Input Validation:**

User inputs for task status and task IDs could potentially cause errors. Additional input validation and error handling would help ensure the program runs smoothly.

Project code

```
import pymysql


connection = pymysql.connect(
    host='localhost',
    user='root',
    password='mathi'
)


cursor = connection.cursor()


cursor.execute("CREATE DATABASE IF NOT EXISTS todo_list_db")


connection.close()
```

```
connection = pymysql.connect(
    host='localhost',
    user='root',
    password='mathi',
    database='todo_list_db'
)

cursor = connection.cursor()

create_table_query = """CREATE TABLE IF NOT EXISTS todos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    task VARCHAR(255) NOT NULL,
    status ENUM('Pending', 'Completed') DEFAULT 'Pending',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)
"""
cursor.execute(create_table_query)
print("Database and Table Created Successfully!")

def add_task(task):

    check_query = "SELECT * FROM todos WHERE task = %s"
    cursor.execute(check_query, (task,))
```

```
    existing_task = cursor.fetchone()

    if existing_task:
        print(f"Task '{task}' already exists!")
    else:
        insert_query = "INSERT INTO todos (task) VALUES (%s)"
        cursor.execute(insert_query, (task,))
        connection.commit()
        print(f"Task '{task}' added successfully!")

def view_tasks():
    select_query = "SELECT * FROM todos"
    cursor.execute(select_query)
    tasks = cursor.fetchall()

    if tasks:
        for task in tasks:
            print(f"ID: {task[0]} | Task: {task[1]} | Status: {task[2]} | Created At:
{task[3]}")
    else:
        print("No tasks found!")

def update_task(task_id, new_status):
    update_query = "UPDATE todos SET status = %s WHERE id = %s"
    cursor.execute(update_query, (new_status, task_id))
```

```
    connection.commit()
    print(f"Task ID {task_id} status updated to '{new_status}'")

def delete_task(task_id):
    delete_query = "DELETE FROM todos WHERE id = %s"
    cursor.execute(delete_query, (task_id,))
    connection.commit()
    print(f"Task ID {task_id} deleted successfully!")

def main_menu():
    while True:
        print("\n--- To-Do List Menu ---")
        print("1. Add Task")
        print("2. View Tasks")
        print("3. Update Task Status")
        print("4. Delete the task")
        print("5. Exit")

        choice = input("Enter your choice: ")

        if choice == '1':
            task = input("Enter the task: ")
            add_task(task)
```

```
        elif choice == '2':
            view_tasks()

        elif choice == '3':
            task_id = input("Enter the task ID to update: ")
            new_status = input("Enter new status (Pending/Completed): ")
            update_task(task_id, new_status)

        elif choice == '4':
            task_id = input("Enter the task ID to delete: ")
            delete_task(task_id)
if __name__ == "__main__":
    main_menu()

conn.close()
```

output:

Database and Table Created Successfully!

--- To-Do List Menu ---

1. Add Task

2. View Tasks

3. Update Task Status

4. Delete the task

5. Exit

Enter your choice: 1

Enter the task: Brushing

Task 'Brushing' added successfully!

--- To-Do List Menu ---

1. Add Task

2. View Tasks

3. Update Task Status

4. Delete the task

5. Exit

Enter your choice: 2

ID: 2 | Task: buy groceries | Status: Pending | Created At: 2024-09-24 18:14:12

ID: 3 | Task: do laundry | Status: Pending | Created At: 2024-09-24 18:14:27

ID: 4 | Task: walking | Status: Pending | Created At: 2024-09-24 18:20:37

ID: 5 | Task: bathing | Status: Pending | Created At: 2024-09-24 18:22:20

ID: 6 | Task: Brushing | Status: Pending | Created At: 2024-09-24 18:25:18

--- To-Do List Menu ---

1. Add Task

2. View Tasks

3. Update Task Status

4. Delete the task

5. Exit

Enter your choice: 3

Enter the task ID to update: 6

Enter new status (Pending/Completed): completed

Task ID 6 status updated to 'completed'

--- To-Do List Menu ---

1. Add Task

2. View Tasks

3. Update Task Status

4. Delete the task

5. Exit

Enter your choice: 2

ID: 2 | Task: buy groceries | Status: Pending | Created At: 2024-09-24 18:14:12

ID: 3 | Task: do laundry | Status: Pending | Created At: 2024-09-24 18:14:27

ID: 4 | Task: walking | Status: Pending | Created At: 2024-09-24 18:20:37

ID: 5 | Task: bathing | Status: Pending | Created At: 2024-09-24 18:22:20

ID: 6 | Task: Brushing | Status: Completed | Created At: 2024-09-24 18:25:18

--- To-Do List Menu ---

1. Add Task

2. View Tasks

3. Update Task Status

4. Delete the task

5. Exit

Enter your choice: 4

Enter the task ID to delete: 6

Task ID 6 deleted successfully!

--- To-Do List Menu ---

1. Add Task

2. View Tasks

3. Update Task Status

4. Delete the task

5. Exit

Enter your choice: 2

ID: 2 | Task: buy groceries | Status: Pending | Created At: 2024-09-24 18:14:12

ID: 3 | Task: do laundry | Status: Pending | Created At: 2024-09-24 18:14:27

ID: 4 | Task: walking | Status: Pending | Created At: 2024-09-24 18:20:37

ID: 5 | Task: bathing | Status: Pending | Created At: 2024-09-24 18:22:20

--- To-Do List Menu ---

1. Add Task

2. View Tasks

3. Update Task Status

4. Delete the task

5. Exit

Enter your choice: 5

--- To-Do List Menu ---

1. Add Task

2. View Tasks

3. Update Task Status

4. Delete the task

5. Exit

Conclusion

This To-Do List Management System provides an efficient way to manage tasks using Python and MySQL. The system supports adding, viewing, updating, and deleting tasks, which are fundamental operations for managing a To-Do list. The project demonstrates the successful use of pymysql for database operations and CRUD functionalities.