Mini Project: Task Management System

Submitted by: Akshed Kottoth  
Course: python + Django  
Institution: Entri  
  
Date: 18 May, 2025

# 1. Introduction

This mini project is a simple Task Management System built using Python and SQLite. The main goal is to help manage team projects by organizing tasks, assigning them to people, tracking hours, and updating progress.  
  
As someone working as an engineering Program Manager, I often must track tasks, billing hours, and project status. I wanted to build a small tool that can help do this in a clean and easy way, and also learn more about Python and databases.

# 2. Objectives

- Create and manage projects with details like PO number and approved hours  
- Add tasks under each project  
- Assign tasks to specific team members  
- Track estimated vs actual (billing) hours  
- View a dashboard showing task status and workload  
- Edit or delete tasks and projects easily

# 3. System Requirements

- Python 3.10 or higher  
- SQLite (built into Python)  
- Visual Studio Code (VS Code)  
- DB Browser for SQLite

# 4. Database Design Projects Table:

|  |  |
| --- | --- |
| Item Name | Type |
| Id | INTEGER (PK) |
| Name | TEXT |
| PO Number | TEXT |
| Approved Hours | INTEGER |

**Tasks Table:**

|  |  |
| --- | --- |
| Item Name | Type |
| Id | INTEGER (PK) |
| Title | TEXT |
| Description | TEXT |
| Project ID | INTEGER (FK) |
| Assigned To | TEXT |
| Estimated Hours | INTEGER |
| Billing Hours | INTEGHER |
| Status | TEXT |

# 5. Features Implemented

|  |  |  |
| --- | --- | --- |
| Sl. No | Feature Name | Description |
| 1 | Create New Project | Add new project clusters with PO number and total hours |
| 2 | Add Task | Create specific tasks under each project with full details |
| 3 | Assign Team Members | Assign tasks to engineers |
| 4 | Update Task | Modify task fields like title, status, hours, etc. |
| 5 | View Dashboard | See summary of all tasks with project-wise status |
| 6 | Delete Task or Project | Remove a task or a whole project (with confirmation) |

# 6. Screenshots

**-The main menu + Adding a project, PO, Estimated Hours**

A screenshot of a computer program

AI-generated content may be incorrect.

**- Adding a task and other details**

A white screen with black text

AI-generated content may be incorrect.

**- Viewing dashboard**

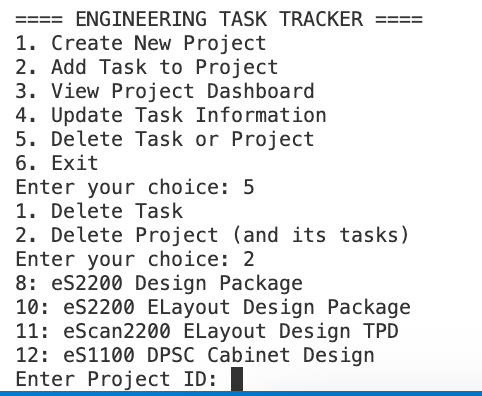
A close-up of a computer screen

AI-generated content may be incorrect.  
**- Updating a task**

A screenshot of a computer program

AI-generated content may be incorrect.

**- Deleting a task or project**



**-DB Browser**

**A screenshot of a computer

AI-generated content may be incorrect.**

# A screenshot of a computer AI-generated content may be incorrect.

# 7. Challenges Faced

- Connecting two tables using foreign keys was confusing at first  
- SQLite queries needed trial-and-error to get right  
- I didn’t understand CLI structure initially but got better with practice  
- Had trouble debugging when the same output kept showing (fixed by isolating features)

# 8. Conclusion

This project helped me understand how software can organize real-world tasks in a team setting. I also learned how to use Python with SQLite, and how to structure a simple terminal-based tool with proper features.

Full Code

import sqlite3

# ==============================

# DATABASE SETUP

# ==============================

def setup\_database():

conn = sqlite3.connect("task\_manager.db")

cursor = conn.cursor()

cursor.execute('''

CREATE TABLE IF NOT EXISTS projects (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

po\_number TEXT,

approved\_hours INTEGER

)

''')

cursor.execute('''

CREATE TABLE IF NOT EXISTS tasks (

id INTEGER PRIMARY KEY AUTOINCREMENT,

title TEXT NOT NULL,

description TEXT,

project\_id INTEGER,

assigned\_to TEXT,

estimated\_hours INTEGER,

billing\_hours INTEGER,

status TEXT,

FOREIGN KEY (project\_id) REFERENCES projects(id)

)

''')

conn.commit()

conn.close()

# ==============================

# FEATURE 1: Create New Project

# ==============================

def create\_project():

name = input("Enter project name: ")

po\_number = input("Enter PO number: ")

approved\_hours = input("Enter approved hours: ")

conn = sqlite3.connect("task\_manager.db")

cursor = conn.cursor()

cursor.execute("INSERT INTO projects (name, po\_number, approved\_hours) VALUES (?, ?, ?)", (name, po\_number, approved\_hours))

conn.commit()

conn.close()

print("Project added successfully.\n")

# ==============================

# FEATURE 2: Add Task to Project

# ==============================

def add\_task():

conn = sqlite3.connect("task\_manager.db")

cursor = conn.cursor()

cursor.execute("SELECT id, name FROM projects")

projects = cursor.fetchall()

if not projects:

print("No projects found.\n")

return

print("Available Projects:")

for p in projects:

print(f"{p[0]}: {p[1]}")

project\_id = input("Enter Project ID: ")

title = input("Task title: ")

description = input("Description: ")

assigned\_to = input("Assigned to: ")

estimated = input("Estimated hours: ")

billed = input("Billing hours: ")

status = input("Status: ")

cursor.execute('''

INSERT INTO tasks (title, description, project\_id, assigned\_to, estimated\_hours, billing\_hours, status)

VALUES (?, ?, ?, ?, ?, ?, ?)

''', (title, description, project\_id, assigned\_to, estimated, billed, status))

conn.commit()

conn.close()

print("Task added.\n")

# ==============================

# FEATURE 3: View Project Dashboard

# ==============================

def view\_dashboard():

conn = sqlite3.connect("task\_manager.db")

cursor = conn.cursor()

cursor.execute("SELECT id, name FROM projects")

projects = cursor.fetchall()

for project in projects:

print(f"\nProject: {project[1]} (ID: {project[0]})")

cursor.execute("SELECT title, assigned\_to, status, estimated\_hours, billing\_hours FROM tasks WHERE project\_id = ?", (project[0],))

tasks = cursor.fetchall()

for task in tasks:

title, assigned, status, est, bill = task

burnout = f"{round(int(bill)/int(est)\*100)}%" if int(est) > 0 else "N/A"

print(f" Task: {title}, Assigned: {assigned}, Status: {status}, Estimated: {est}, Billed: {bill}, Burnout: {burnout}")

conn.close()

# ==============================

# FEATURE 4: Update Task Info

# ==============================

def update\_task():

conn = sqlite3.connect("task\_manager.db")

cursor = conn.cursor()

cursor.execute("SELECT id, title FROM tasks")

tasks = cursor.fetchall()

if not tasks:

print("No tasks found.\n")

return

for t in tasks:

print(f"{t[0]}: {t[1]}")

task\_id = input("Enter Task ID: ")

print("1. Title\n2. Assigned To\n3. Status\n4. Estimated Hours\n5. Billing Hours")

choice = input("Choose field to update: ")

field\_map = {'1': 'title', '2': 'assigned\_to', '3': 'status', '4': 'estimated\_hours', '5': 'billing\_hours'}

if choice not in field\_map:

print("Invalid choice.")

return

new\_val = input("New value: ")

cursor.execute(f"UPDATE tasks SET {field\_map[choice]} = ? WHERE id = ?", (new\_val, task\_id))

conn.commit()

conn.close()

print("Task updated.\n")

# ==============================

# FEATURE 5: Delete Task or Project

# ==============================

def delete\_task\_or\_project():

conn = sqlite3.connect("task\_manager.db")

cursor = conn.cursor()

print("1. Delete Task\n2. Delete Project (and its tasks)")

choice = input("Enter your choice: ")

if choice == '1':

cursor.execute("SELECT id, title FROM tasks")

tasks = cursor.fetchall()

for t in tasks:

print(f"{t[0]}: {t[1]}")

task\_id = input("Enter Task ID: ")

confirm = input("Are you sure? (yes/no): ")

if confirm.lower() == 'yes':

cursor.execute("DELETE FROM tasks WHERE id = ?", (task\_id,))

conn.commit()

print("Task deleted.\n")

elif choice == '2':

cursor.execute("SELECT id, name FROM projects")

projects = cursor.fetchall()

for p in projects:

print(f"{p[0]}: {p[1]}")

project\_id = input("Enter Project ID: ")

confirm = input("Are you sure? This will delete all tasks in it too. (yes/no): ")

if confirm.lower() == 'yes':

cursor.execute("DELETE FROM tasks WHERE project\_id = ?", (project\_id,))

cursor.execute("DELETE FROM projects WHERE id = ?", (project\_id,))

conn.commit()

print("Project and tasks deleted.\n")

else:

print("Invalid option.")

conn.close()

# ==============================

# MAIN MENU

# ==============================

def main():

while True:

print("\n==== ENGINEERING TASK TRACKER ====")

print("1. Create New Project")

print("2. Add Task to Project")

print("3. View Project Dashboard")

print("4. Update Task Information")

print("5. Delete Task or Project")

print("6. Exit")

choice = input("Enter your choice: ")

if choice == '1':

create\_project()

elif choice == '2':

add\_task()

elif choice == '3':

view\_dashboard()

elif choice == '4':

update\_task()

elif choice == '5':

delete\_task\_or\_project()

elif choice == '6':

print("Goodbye.")

break

else:

print("Invalid choice.\n")

# ==============================

# ENTRY POINT

# ==============================

setup\_database()

main()