BACSE101 Problem Solving using Python

PROJECT REPORT

on

STUDENT TASK MANAGER

Prepared by

B BINDU HASINI - 25BCE2748
ELSA ROSE ANTONY- 25BCE2728
HAVISHA SURESH - 25BCE2708

Under the supervision of

SUBASHREE BARADA

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering
Vellore Institute of Technology, Vellore.

November 04, 2025

Table of Contents

Abstract

This project implements a simple Student Task Manager, a web-based application that allows students to create, view, update and delete tasks. The backend is written in Python using the Flask framework and MySQL for persistence. The frontend is a single-page HTML file that communicates with the backend rendered dynamically through Flask templates (Jinja2) combined with Bootsrap for responsiveness. The system demonstrates basic CRUD (Create, Read, Update, Delete) functionality, separation of concerns between frontend and backend, and can be extended with authentication, notifications, and richer UI later.

1. Introduction

The Student Task Manager aims to simplify how students track their academic responsibilities, such as assignments, lab submissions, and personal study goals. It bridges the gap between manual planning and efficient digital organization, offering a seamless way to monitor deadlines and progress. The backend handles all logic and database interactions, while the frontend provides a clean, user-friendly interface. Together, they create a minimal yet functional productivity tool that aligns with real student needs.

1.1 Domain Information

A student task manager helps students track assignments, lab work, and personal study tasks. It provides an easy interface to add new tasks with due dates, update status, and remove completed tasks. This lightweight project focuses on building a clear backend API in Python and a minimal, responsive HTML frontend that calls the API. It demonstrates server-side rendering using Flask templates, CRUD database handling, and layout responsiveness with Bootstrap. The use of Flask Jinja2 templates bridges the gap between static HTML and dynamic, data-driven interfaces.

Software Libraries Used

- Python 3.13
- MySQL (file-based database)
- Flask
- Flask-MySQLdb
- Bootstrap 5.3
- HTML5, CSS3, Jinja2 Template Engine

Supporting files: requirements.txt

- Flask>=2.0
- Flask-MySQLdb>=1.0
- Bootstrap>=5.3

1.2 Contributions by Team Members

• B Bindu Hasini: Frontend design and Bootstrap styling, integration with Flask templates

• Elsa Rose Antony: Backend route design, database schema, MySQL setup

•Havisha Suresh: Documentation, testing, Debugging and validation.

1.3 Challenges Faced

- Implementing a notification feature that reminds users about upcoming due dates - this feature was planned but not completed within the time frame.
- Ensuring synchronization between frontend UI updates and backend database

2. Problem Statement and Objectives

Problem statement:

Students need a lightweight, easy-to-run system to record and manage tasks (assignments, lab tasks, study items). The project should let users add tasks with title, description, due date and status, view a list of tasks, edit them, and delete them.

Objectives:

1. Design and implement a RESTful backend in Python that handles all CRUD (Create, Read, Update, Delete) operations for task data and ensures smooth communication between the server and client.
2. To integrate a MySQL database for reliable, persistent data storage and retrieval for seamless query handling.
3. To Build a responsive HTML frontend that calls the API and updates the UI dynamically.
4. Provide clear instructions to run the project locally.
5. To lay the groundwork for future enhancements such as user authentication, notifications for due dates, and task categorization or prioritization.
6. Implementation

3. Implementation

3.1 Features

- Add new tasks with a title, description, and due date.
- View all existing tasks in a clear, organized table.
- Search or filter tasks based on completion status or upcoming deadlines (future enhancement).
- Edit existing tasks to update details or change their completion status.
- Delete outdated or completed tasks easily.
- Responsive frontend interface with instant updates using JavaScript and REST API.
- Integration with MySQL for efficient and reliable task storage.

Implementation CODE:

-(main.py)-backend

3.2: INTEGRATION WITH MYSQL:

```
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root",
    database="PYPROJECT"
)
cursor = db.cursor(dictionary=True)
```

3.3: ADDING NEW TASK:

```
@app.route('/add', methods=['POST'])
def add_task():
    title = request.form['title']
    desc = request.form['description']
    deadline = request.form['deadline']
    priority = request.form['priority']
    cursor.execute("INSERT INTO tasks (title, description, deadline, priority, status) VALUES (%s,%s,%s,%s,%s)",
                   (title, desc, deadline, priority, 'Pending'))
    db.commit()
    return redirect('/')
```

3.4: UPDATING TASK STATUS:

```
@app.route('/update/<int:id>')
def update_status(id):
    cursor.execute("UPDATE tasks SET status='Completed' WHERE id=%s", (id,))
    db.commit()
    return redirect('/')
```

3.5: DELETION OF TASKS:

```
@app.route('/delete/<int:id>')
def delete_task(id):
    cursor.execute("DELETE FROM tasks WHERE id=%s", (id,))
    db.commit()
    return redirect('/')
```

-Key parts of the frontend (index.html):

3.6**:** THE ACTION LINKS TRIGGER BACKEND UPDATES: (.html)

```
<td>
    {% if t.status != 'Completed' %}
    <a href="/update/{{ t.id }}" class="btn btn-sm btn-success">✓ Done</a>
    {% endif %}
    <a href="/delete/{{ t.id }}" class="btn btn-sm btn-danger">🗑 Delete</a>
</td>
</tr>
```

4.Demo Screenshots

Database and table in MySQL:

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| pyproject          |
| sakila             |
| sys                |
| world              |
+--------------------+
7 rows in set (0.00 sec)
```

```
mysql> use PYPROJECT
Database changed
mysql> Show tables
    -> ;
+----------------------+
| Tables_in_pyproject  |
+----------------------+
| tasks                |
+----------------------+
```

```
mysql> select * from tasks;
+----+----------------+---------------------------------+------------+----------+-----------+
| id | title          | description                     | deadline   | priority | status    |
+----+----------------+---------------------------------+------------+----------+-----------+
|  1 | Math Assignment | Complete exercises from chapter 5 | 2025-10-25 | High     | Completed |
|  2 | Science Project | Prepare volcano model           | 2025-10-28 | Medium   | Completed |
|  3 | History Essay   | Write essay on World War II     | 2025-10-30 | High     | Pending   |
|  5 | Computer Lab    | Finish Python lab exercises     | 2025-10-27 | Medium   | Completed |
|  7 | python project  | presentation                    | 2025-11-05 | High     | Pending   |
+----+----------------+---------------------------------+------------+----------+-----------+
5 rows in set (0.01 sec)
```

Backend: main.py (code)

main.py - C:\Users\BINDU HASINI\Desktop\py project\main.py (3.13.7)

File Edit Format Run Options Window Help

```
from flask import Flask, render_template, request, redirect
import mysql.connector
from datetime import date

app = Flask(__name__)

# Database connection
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root",
    database="PYPROJECT"
)
cursor = db.cursor(dictionary=True)

@app.route('/')
def index():
    cursor.execute("SELECT * FROM tasks ORDER BY deadline")
    tasks = cursor.fetchall()
    today = date.today()
    for t in tasks:
        if t['status'].lower() != 'completed':
            if t['deadline'] and t['deadline'] < today:
                t['remark'] = "Overdue"
            elif t['deadline'] == today:
                t['remark'] = "Due Today"
            else:
                t['remark'] = ""
        else:
            t['remark'] = "Done"
    return render_template('index.html', tasks=tasks)

@app.route('/add', methods=['POST'])
def add_task():
    title = request.form['title']
    desc = request.form['description']
    deadline = request.form['deadline']
    priority = request.form['priority']
    cursor.execute("INSERT INTO tasks (title, description, deadline, priority, status) VALUES (%s,%s,%s,%s,%s)",
                   (title, desc, deadline, priority, 'Pending'))
    db.commit()
    return redirect('/')

@app.route('/update/<int:id>')
def update_status(id):
    cursor.execute("UPDATE tasks SET status='Completed' WHERE id=%s", (id,))
    db.commit()
    return redirect('/')

@app.route('/delete/<int:id>')
def delete_task(id):
    cursor.execute("DELETE FROM tasks WHERE id=%s", (id,))
    db.commit()
    return redirect('/')

if __name__ == "__main__":
    app.run(debug=True)
```

Backend: main.py (output) :

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:\Users\BINDU HASINI\Desktop\py project\main.py ==========
 * Serving Flask app 'main'
 * Debug mode: on
[31m[1mWARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.[0m
 * Running on http://127.0.0.1:5000
[33mPress CTRL+C to quit[0m
 * Restarting with stat
```

Frontend: index.html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Student Task Manager</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body class="bg-light">
    <div class="container mt-5">
        <h2 class="text-center mb-4">🚿 Student Task Manager</h2>

        <form method="POST" action="/add" class="card p-3 mb-4 shadow-sm">
            <h5>Add New Task</h5>
            <div class="row g-2">
                <div class="col-md-3"><input name="title" class="form-control" placeholder="Task Title" required></div>
                <div class="col-md-3"><input name="description" class="form-control" placeholder="Description"></div>
                <div class="col-md-2"><input type="date" name="deadline" class="form-control"></div>
                <div class="col-md-2">
                    <select name="priority" class="form-select">
                        <option>Low</option>
                        <option>Medium</option>
                        <option>High</option>
                    </select>
                </div>
                <div class="col-md-2">
                    <button type="submit" class="btn btn-primary w-100">Add</button>
                </div>
            </div>
        </form>

        <table class="table table-striped table-bordered shadow-sm">
            <thead class="table-dark">
                <tr>
                    <th>ID</th>
                    <th>Title</th>
                    <th>Deadline</th>
                    <th>Priority</th>
                    <th>Status</th>
                    <th>Remark</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody>
                {% for t in tasks %}
                <tr class="
                    {% if t.remark == 'Overdue' %}table-danger
                    {% elif t.remark == 'Due Today' %}table-warning
                    {% elif t.remark == 'Done' %}table-success
                    {% endif %}
                ">
                    <td>{{ t.id }}</td>
                    <td>{{ t.title }}</td>
                    <td>{{ t.deadline }}</td>
                    <td>{{ t.priority }}</td>
                    <td>{{ t.status }}</td>
                    <td>{{ t.remark }}</td>
                    <td>
                        {% if t.status != 'Completed' %}
                        <a href="/update/{{ t.id }}" class="btn btn-sm btn-success">✓ Done</a>
                        {% endif %}
                        <a href="/delete/{{ t.id }}" class="btn btn-sm btn-danger">🗑 Delete</a>
                    </td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
    </div>
</body>
</html>
```

[The frontend uses Flask's Jinja2 template engine to dynamically generate HTML content. The main template (index.html) displays all tasks in a Bootstrap-styled table, color-coded based on remarks Each form submission or button click triggers a backend route, and Flask automatically re-renders the updated page, giving the effect of instant updates.]

HOW TO RUN:

Prerequisites:

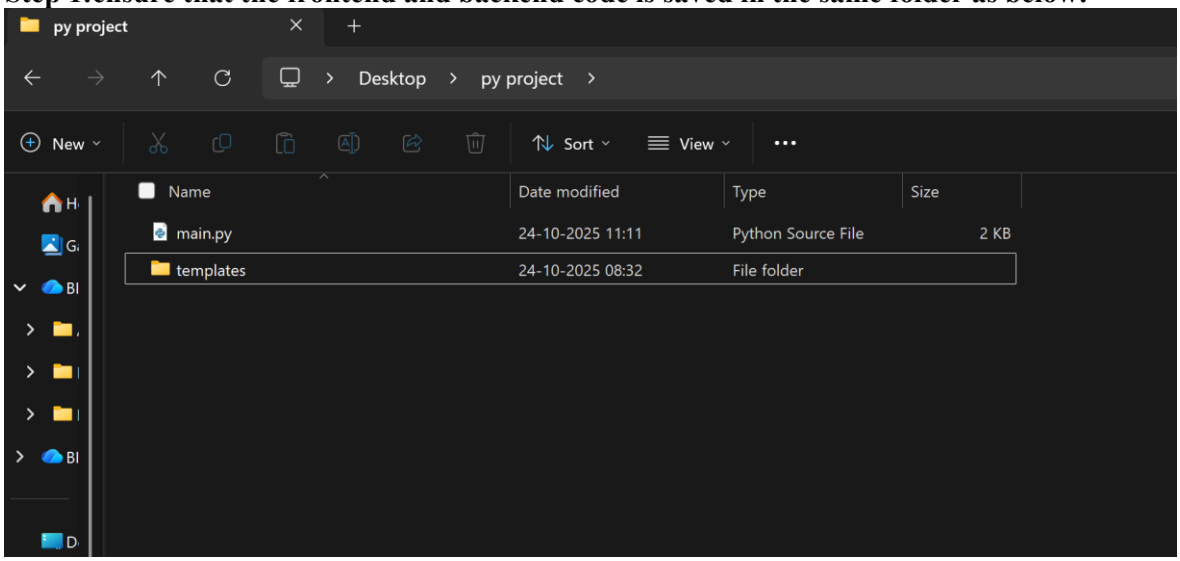
- **Create a virtual environment (recommended):**

```
python -m venv venv
source venv/bin/activate  # on Windows: venv\\Scripts\\activate
```

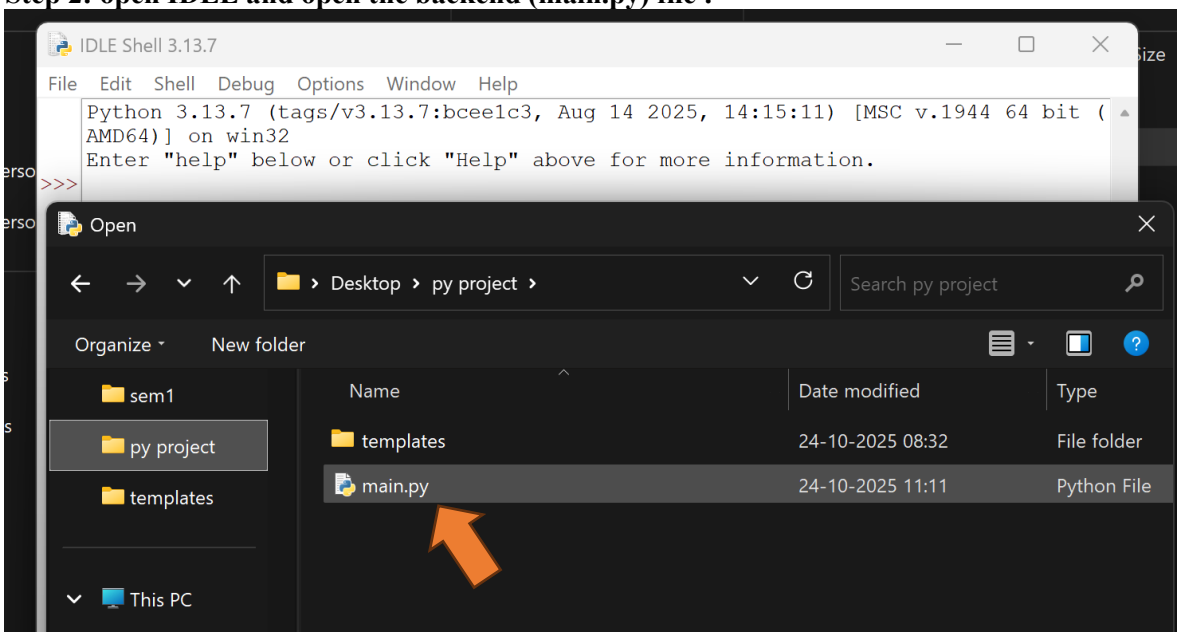
- Install dependencies:

```
pip install -r requirements.txt
```

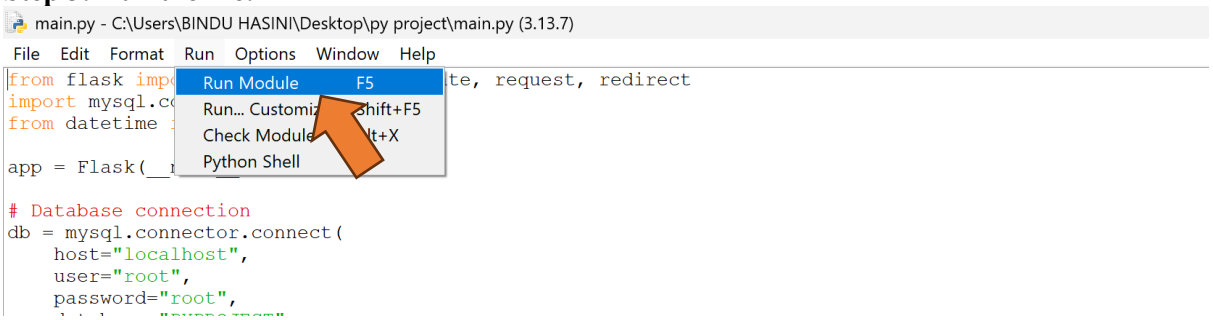
Step 1:ensure that the frontend and backend code is saved in the same folder as below:



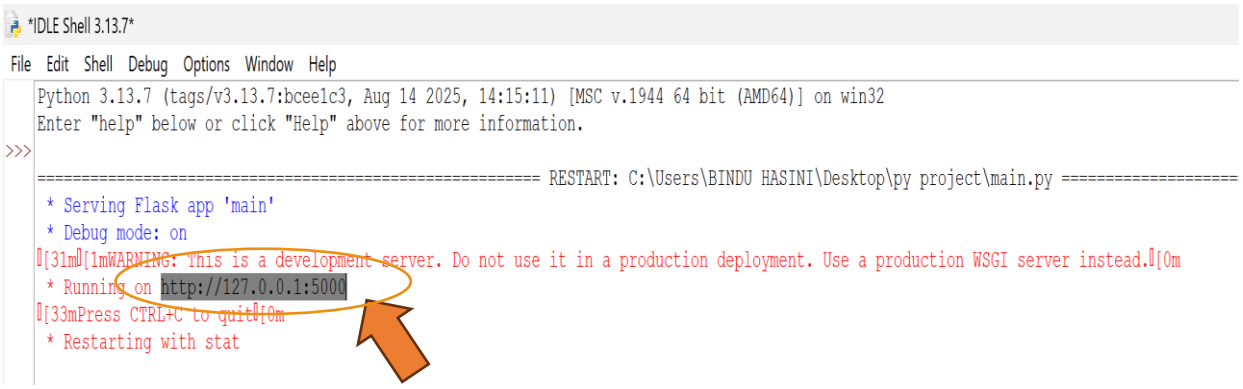
Step 2: open IDLE and open the backend (main.py) file :



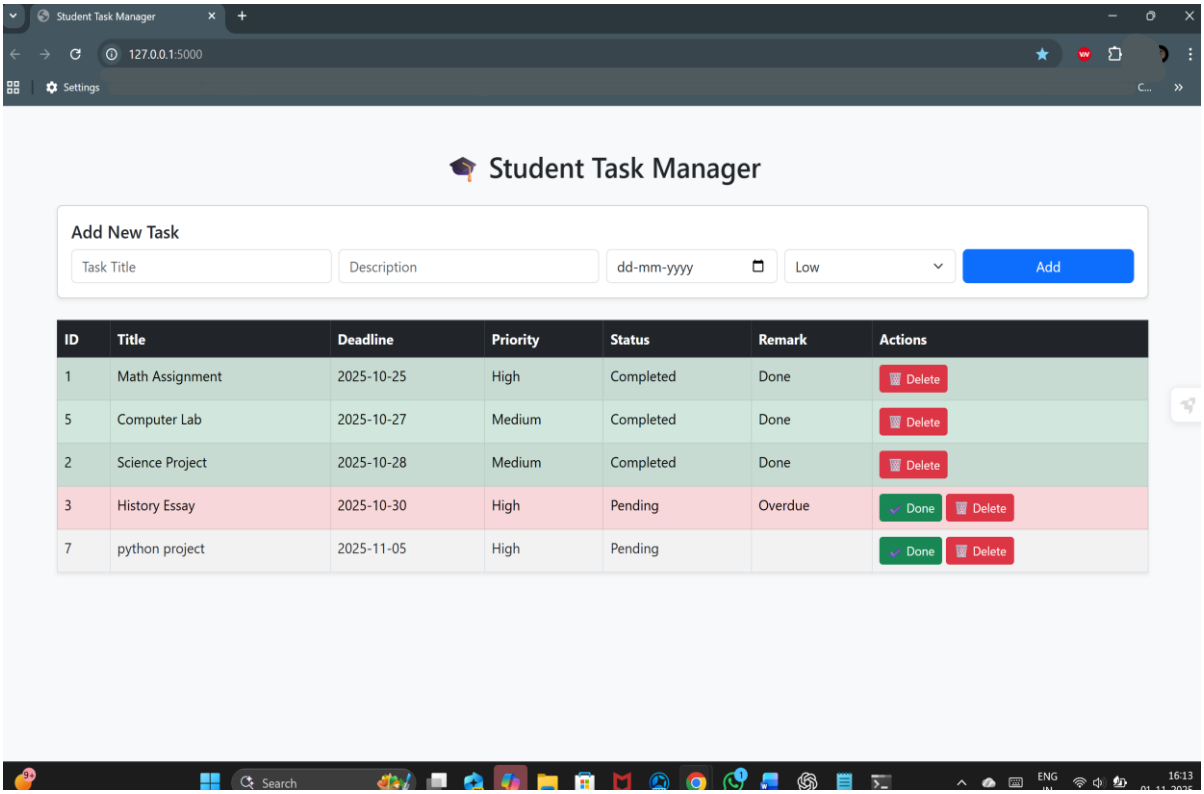
Step 3: Run the file:



Step 4: The link to your application will be given:
Copy and Paste the link in your browser(like edge or chrome to open .

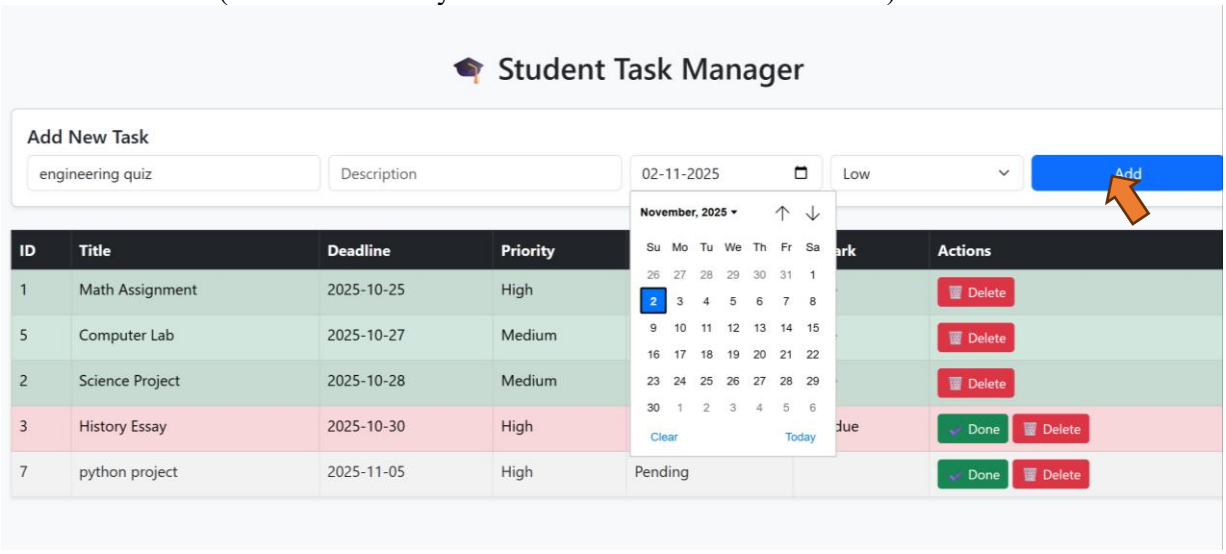


The application now opens:

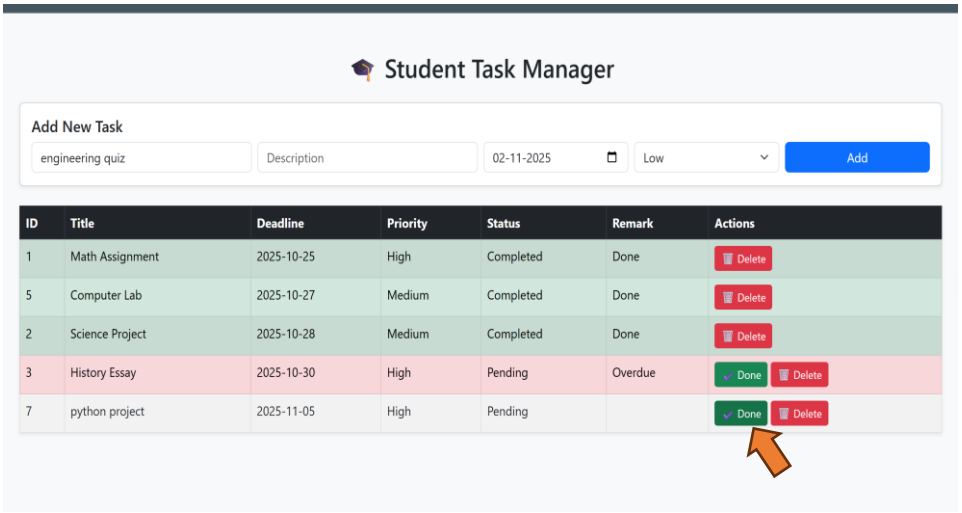


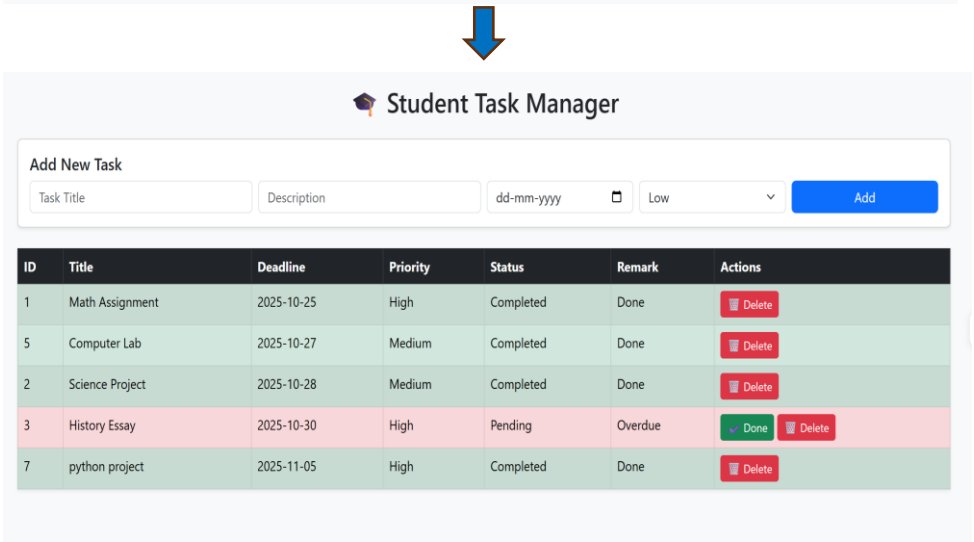
The user can view the tasks with its priority and due dates and also add, check-off or delete the tasks:

ADDING TASK- (enter the necessary details of the task and click on ADD)

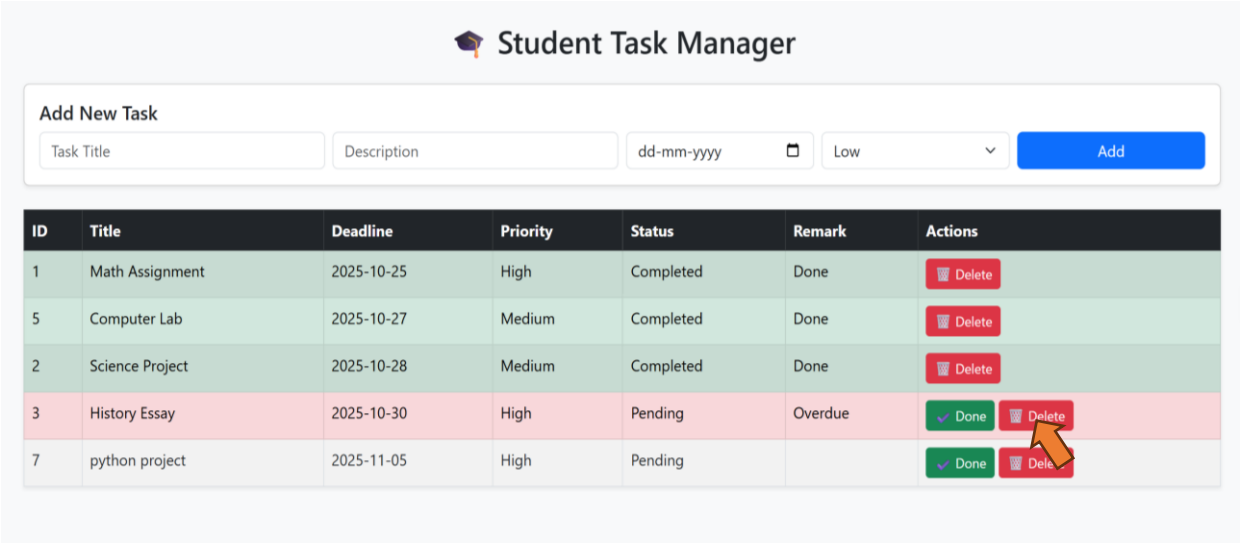


UPDATING TASK STATUS: (click on done to check-off a completed task)





DELETING TASK: click on the DELETE button to delete outdated tasks:



5. Conclusion

The development of the Student Task Manager provided valuable insights into building a complete web-based application from the ground up. Through this project, we learned to integrate Python-based backend logic with a clean and responsive HTML frontend, ensuring seamless communication through REST APIs. Using MySQL as the database strengthened our understanding of data management, relational modelling, and CRUD operations in a real-world setting.

The project successfully achieved its objective of allowing students to manage their academic tasks effectively. The modular design of the backend made it easier to maintain and extend, while the frontend interface offered clarity and simplicity to the end user. The system's flexibility ensures that additional features such as user login, notification alerts for deadlines, or task prioritization can be easily integrated in the future.

Beyond technical implementation, the development of the Student Task Manager offered hands-on experience in full-stack web application design using Flask and MySQL. This project also emphasized teamwork, debugging, and problem-solving under realistic constraints. It strengthened our understanding of route handling, server-side rendering, and database management. Each team member contributed to different layers of the project, leading to a coherent and functional outcome. It helped us appreciate how structured collaboration and clear communication can enhance development outcomes. Overall, the Student Task Manager stands as a practical demonstration of our ability to apply programming concepts to design an efficient, user-centred solution that addresses everyday academic challenges faced by students.