PYTHON FULL STACK WEB DEVELOPMENT- Task 1

User Management Web Application" using Python (Flask), HTML, CSS & SQLite

Objective:
The objective of this task is to introduce interns to **end-to-end full stack web development using Python.**
You will learn how:

- Frontend (HTML, CSS)
- Backend (Python with Flask)
- Database (SQLite)

work together as one complete system, just like real industry web applications.
This task focuses on **fundamentals, clarity, and confidence**, not complexity.

---

2. Industry Context (Why This Task Matters)

Python Full Stack Developers are in high demand for:

- Web platforms
- SaaS products
- Admin dashboards
- Internal company tools

Most real applications start with **user management**:

- Adding users
- Viewing users
- Storing data in databases

This task mirrors how **actual backend-driven web apps are built** in companies.

---

3. Real-World Problem Statement

"A company needs a simple web application to store and manage user details through a web interface."
You are required to build a **User Management Web Application** where users can:

- Add new user details
- View all users in a table
- Store data permanently in a database

4. What You Will Build (Very Clearly Defined)

A **Full Stack Web Application** with:

Frontend

- HTML form to add user details
- Styled user table to display data

Backend

- Flask server
- Routes to handle requests
- Database operations

Database

- SQLite database to store user records

---

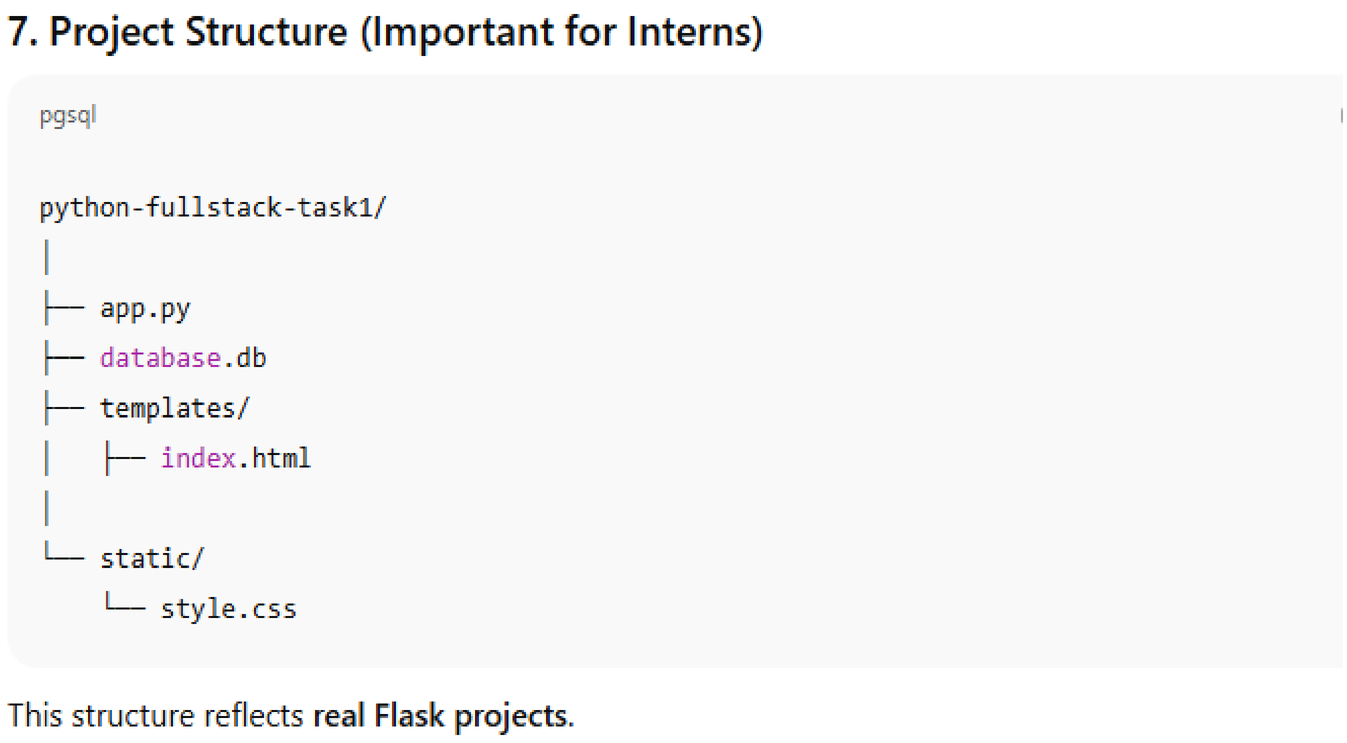
5. Features to Implement

1. Home page with user input form
2. Submit form data to backend
3. Store user details in SQLite database
4. Fetch and display all users
5. Clean UI with basic CSS styling

---

6. Tech Stack (Beginner-Friendly)

- **Backend:** Python + Flask
- **Frontend:** HTML, CSS
- **Database:** SQLite
- **Tools:** VS Code, Browser



7. Project Structure (Important for Interns)

pgsql

```
python-fullstack-task1/
│
├── app.py
├── database.db
├── templates/
│   ├── index.html
│
└── static/
    └── style.css
```

This structure reflects **real Flask projects.**

8. Step-by-Step Development Guide

Step 1: Environment Setup

- Install Python (3.10+)
- Install Flask using pip

```bash
pip install flask
```

Step 2: Create Backend (Flask App)

`app.py` (Starter Backend Code)

```python
from flask import Flask, render_template, request, redirect
import sqlite3

app = Flask(__name__)

def get_db_connection():
    conn = sqlite3.connect('database.db')
    conn.row_factory = sqlite3.Row
    return conn

@app.route('/', methods=['GET', 'POST'])
def index():
    conn = get_db_connection()

    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']

        conn.execute(
            'INSERT INTO users (name, email) VALUES (?, ?)',
            (name, email)
        )
        conn.commit()
        return redirect('/')

    users = conn.execute('SELECT * FROM users').fetchall()
    conn.close()
    return render_template('index.html', users=users)

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

Step 3: Create Database (One-Time)

Run this once in Python shell:

```python
import sqlite3

conn = sqlite3.connect('database.db')
conn.execute('CREATE TABLE users (id INTEGER PRIMARY KEY, name TEXT, email TEXT)')
conn.close()
```

Step 4: Frontend – HTML Template

`templates/index.html`

```html
<!DOCTYPE html>
<html>
<head>
    <title>User Management</title>
    <link rel="stylesheet" href="/static/style.css">
</head>
<body>

<h2>User Management System</h2>

<form method="POST">
    <input type="text" name="name" placeholder="Enter Name" required>
    <input type="email" name="email" placeholder="Enter Email" required>
    <button type="submit">Add User</button>
</form>

<table>
    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Email</th>
    </tr>
    {% for user in users %}
    <tr>
        <td>{{ user.id }}</td>
        <td>{{ user.name }}</td>
        <td>{{ user.email }}</td>
    </tr>
    {% endfor %}
</table>

</body>
</html>
```

Step 5: CSS Styling

`static/style.css`

```css
body {
    font-family: Arial;
    background: #f4f6f8;
    padding: 40px;
}

h2 {
    color: #1e293b;
}

form input, button {
    padding: 8px;
    margin: 5px;
}

table {
    width: 60%;
    margin-top: 20px;
    border-collapse: collapse;
}

table, th, td {
    border: 1px solid #ccc;
}

th {
    background: #1e40af;
    color: white;
}
```

9. How to Run the Project

```bash
python app.py
```

Open browser:

```cpp
http://127.0.0.1:5000/
```

10. Deliverables

Each intern must submit:

1. Project folder (code files)
2. Screenshots of the running application
3. Database file (database.db)
4. README explaining:
   - Project flow
   - Technologies used

11. Learning Outcome

After completing this task, interns will:

- Understand Python full stack architecture
- Know how frontend, backend, and database connect
- Be ready for advanced topics like authentication, REST APIs, and deployment

---

10. Free YouTube Learning Support (Highly Recommended)

Python Basics

- Python Full Course (Beginner)
  https://www.youtube.com/watch?v=rfscVS0vtbw

Flask Backend

- Flask Full Tutorial (Beginner Friendly)
  https://www.youtube.com/watch?v=Z1RJmh_OqeA
- Flask Project Tutorial (CRUD App)
  https://www.youtube.com/watch?v=3mwFC4SHY-Y

HTML & CSS Refresher

- HTML & CSS Crash Course
  https://www.youtube.com/watch?v=hu-q2zYwEYs

11. Free Documentation & Study Material

- Flask Official Docs
  https://flask.palletsprojects.com
- SQLite Documentation
  https://www.sqlite.org/docs.html
- Python Official Docs
  https://docs.python.org/3/