Project overview of Task Manager app

K M Shihab Jamil

January 2026

Contents

1 Introduction **2**
1.1 Overview of the Project 2
1.2 System Development Life Cycle (SDLC) Perspective 2
1.3 Problem Statement 2
1.4 Objectives of the Project 3
1.5 Scope of the Project 3
1.6 SDLC Phase Mapping for the Task Manager System 3
1.6.1 Planning and Feasibility 3
1.6.2 Requirement Analysis 4
1.6.3 System Design 4
1.6.4 Implementation 4
1.6.5 Testing and Maintenance 5
1.7 Organization of the Full Report 5

Chapter 1

Introduction

1.1 Overview of the Project

This chapter describes the foundation of the complete project report and explains how the proposed *Task Manager Web Application* fits within the System Development Life Cycle (SDLC). The purpose of this chapter is not to implement the system, but to define how the project is understood, justified, and structured before development begins. This chapter sets the direction for the remaining chapters of the report.

The Task Manager Web Application is a simple web-based system designed to allow users to create accounts, manage sessions, and perform basic Create, Read, Update, and Delete (CRUD) operations on tasks. The system is intentionally kept minimal to clearly demonstrate core software engineering principles rather than complex business logic.

1.2 System Development Life Cycle (SDLC) Perspective

The System Development Life Cycle (SDLC) provides a structured framework for developing the Task Manager system. SDLC is chosen because it divides system development into clear and manageable phases, ensuring that the system is built systematically and logically.

In this project, SDLC is used as a guiding model to explain how the system progresses from idea to deployment. Each phase of SDLC is mapped conceptually to the Task Manager project and elaborated in later chapters of the report.

1.3 Problem Statement

Many individuals struggle to organize daily tasks efficiently using manual methods such as notebooks or unstructured digital notes. These approaches lack accessibility, persistence, and proper management features. There is a need for a simple, lightweight, and accessible task management system that allows users to manage tasks securely from any device.

The Task Manager Web Application aims to solve this problem by providing a centralized, user-authenticated platform for task organization.

1.4 Objectives of the Project

The objectives of the project, as defined in this introductory chapter, are as follows:

- To demonstrate the practical application of the SDLC model
- To design a basic yet complete web-based system
- To implement authentication and session handling
- To demonstrate database interaction and CRUD operations
- To ensure simplicity, usability, and clarity in system design

1.5 Scope of the Project

This project is limited to core functionalities required to demonstrate SDLC concepts. The scope includes:

- User registration and login
- Secure session handling
- Task creation, modification, deletion, and viewing
- Database-based data storage

Features such as payment processing, third-party API integration, and advanced analytics are intentionally excluded to maintain focus on fundamental system development concepts.

1.6 SDLC Phase Mapping for the Task Manager System

This chapter conceptually maps the Task Manager system to the SDLC phases:

1.6.1 Planning and Feasibility

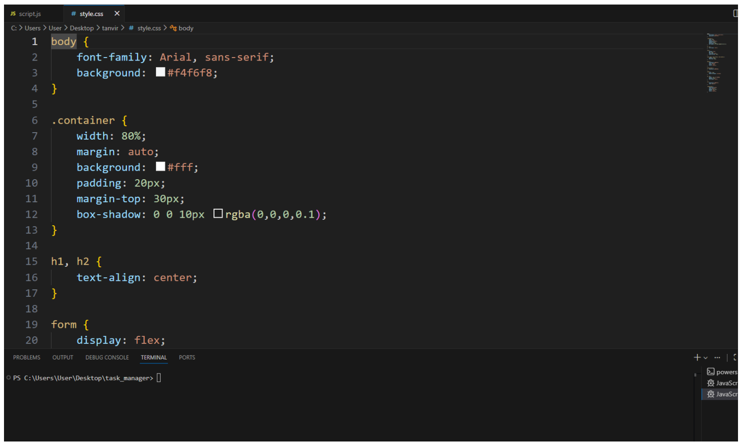
The planning phase identifies the need for a task management system and evaluates its feasibility in terms of technology, cost, and operation. Since the project uses free tools and simple technologies, it is considered feasible.

1.6.2 Requirement Analysis

User requirements such as authentication, task management, and data persistence are identified. These requirements form the basis for the Software Requirement Specification (SRS), which is discussed in the next chapter.

1.6.3 System Design

The system is planned using a basic three-tier architecture consisting of frontend, backend, and database layers. Design decisions are documented before implementation.



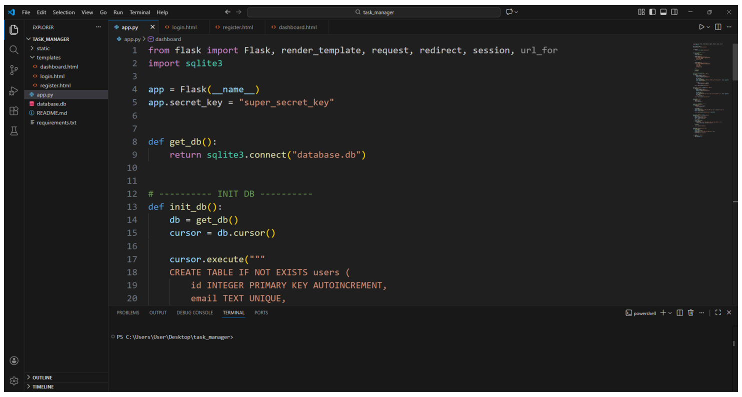
```css
body {
    font-family: Arial, sans-serif;
    background: #f4f6f8;
}

.container {
    width: 80%;
    margin: auto;
    background: #fff;
    padding: 20px;
    margin-top: 30px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
}

h1, h2 {
    text-align: center;
}

form {
    display: flex;
```

```
PS C:\Users\User\Desktop\task_manager>
```

Figure 1.1: Style.css

1.6.4 Implementation

During implementation, the system design is converted into executable code. Each module is developed individually and later integrated.



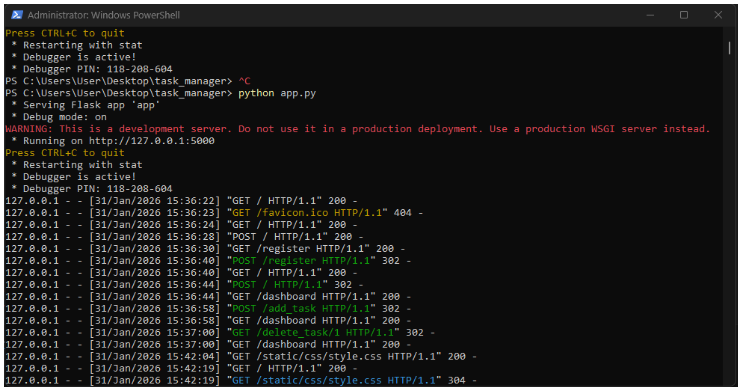
```python
from flask import Flask, render_template, request, redirect, session, url_for
import sqlite3

app = Flask(__name__)
app.secret_key = "super_secret_key"


def get_db():
    return sqlite3.connect("database.db")


# ---------- INIT DB ----------
def init_db():
    db = get_db()
    cursor = db.cursor()

    cursor.execute("""
    CREATE TABLE IF NOT EXISTS users (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        email TEXT UNIQUE,
```

```
PS C:\Users\User\Desktop\task_manager>
```

Figure 1.2: Application code

1.6.5 Testing and Maintenance

Testing ensures that the system meets all specified requirements. Maintenance includes fixing bugs and improving features after deployment.



```
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 118-208-604
PS C:\Users\User\Desktop\task_manager> ^C
PS C:\Users\User\Desktop\task_manager> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 118-208-604
127.0.0.1 - - [31/Jan/2026 15:36:22] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2026 15:36:23] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [31/Jan/2026 15:36:24] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2026 15:36:28] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2026 15:36:30] "GET /register HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2026 15:36:40] "POST /register HTTP/1.1" 302 -
127.0.0.1 - - [31/Jan/2026 15:36:40] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2026 15:36:44] "POST / HTTP/1.1" 302 -
127.0.0.1 - - [31/Jan/2026 15:36:44] "GET /dashboard HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2026 15:36:58] "POST /add_task HTTP/1.1" 302 -
127.0.0.1 - - [31/Jan/2026 15:36:58] "GET /dashboard HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2026 15:37:00] "GET /delete_task/1 HTTP/1.1" 302 -
127.0.0.1 - - [31/Jan/2026 15:37:00] "GET /dashboard HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2026 15:42:04] "GET /static/css/style.css HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2026 15:42:19] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2026 15:42:19] "GET /static/css/style.css HTTP/1.1" 304 -
```

Figure 1.3: Deployment locally

1.7 Organization of the Full Report

The remainder of the report is structured as follows:

- Chapter 2: Requirement Analysis and Feasibility Study
- Chapter 3: System Design
- Chapter 4: Implementation and Testing
- Chapter 5: Conclusion and Future Enhancements

This chapter provides the conceptual foundation of the project and explains how the Task Manager Web Application fits within the SDLC framework.