Mini Project Assignment

Eduwise Solutions

Task Tracker: Web-Based Task Management System

1. Problem Statement

Organizations and individuals require efficient tools to manage tasks, track progress, and maintain productivity. You are required to design and develop a web-based task management application that enables users to perform CRUD (Create, Read, Update, Delete) operations on tasks through an intuitive interface.

2. Project Objectives

- Develop a functional web application using Flask framework
- Implement persistent data storage mechanism
- Design a user-friendly interface for task management
- Apply software engineering principles including modular design and documentation
- Demonstrate proficiency in version control using Git and GitHub
- Deploy the application in a Linux environment

3. Technical Requirements

3.1 Core Functionality (80 marks)

A. Application Features

- Homepage displaying all tasks with their current status
- Add new task functionality with title and description fields
- Update task status (mark as completed/incomplete)
- Delete task functionality with appropriate confirmation
- Task filtering and sorting capabilities (optional enhancement)

B. Data Persistence Layer

Implement one of the following storage mechanisms:

- SQLite Database: Recommended - Provides relational data management
- JSON File Storage: Acceptable - File-based persistent storage
- In-memory Dictionary: Minimal acceptance - Limited persistence

C. RESTful API Endpoints (Bonus - 5 marks)

- GET /api/tasks - Retrieve all tasks in JSON format
- GET /api/task/<id> - Retrieve specific task details
- POST /api/task - Create new task via API
- PUT /api/task/<id> - Update task via API (optional)
- DELETE /api/task/<id> - Delete task via API (optional)

D. User Interface Requirements

- Utilize Jinja2 templating engine for dynamic content rendering
- Implement responsive design using Bootstrap or custom CSS
- Create intuitive navigation and user feedback mechanisms
- Ensure proper form validation and error handling

E. Deployment Configuration (Bonus - 5 marks)

- Deploy application on Linux environment
- Configure WSGI server (Gunicorn recommended)
- Document deployment process with shell scripts
- Implement proper environment variable management

3.2 Architecture & Design Documentation (10 marks)

Submit a comprehensive design document (1-1.5 pages) including:

- **System Architecture Diagram:** Illustrate the application structure, including folder organization, component relationships, and data flow
- **Process Flow Diagram:** Demonstrate the request-response cycle: User Action → Route Handler → Business Logic → Data Layer → Response

- **Design Rationale:** Justify key technical decisions such as:
  - Choice of data storage mechanism
  - Application structure and modularization
  - Template and static file organization
  - Security considerations

3.3 Project Documentation (10 marks)

Create a comprehensive README.md file in the GitHub repository containing:

- Project Overview and Purpose
- Technology Stack (Framework, Database, Libraries)
- System Architecture and Folder Structure
- Installation and Setup Instructions
- Usage Guide with Screenshots
- API Documentation (if implemented)
- Testing Procedures
- Known Issues and Limitations
- Future Enhancement Roadmap
- Contributors and Acknowledgments

4. Required Technical Concepts

- Linux Administration: Virtual environment creation, package management, process management, file permissions
- Python Development: Module structure, package management, dependency management (requirements.txt)
- Flask Framework: Routing, templating with Jinja2, static file serving, request handling, session management
- Version Control: Git workflow, branching strategies, commit conventions, GitHub repository management

- Database Management: Schema design, CRUD operations, query optimization (if using SQLite)
- Web Technologies: HTML5, CSS3, JavaScript (basic), HTTP methods

5. Evaluation Criteria

Criteria	Description	Marks
Core Functionality	Implementation of all required features with proper functionality	40
Data Persistence	Robust data storage and retrieval mechanism	20
Code Quality	Clean, modular, and well-commented code	10
UI/UX Design	Intuitive interface and user experience	10
Architecture Documentation	Comprehensive design document with diagrams	10
Project Documentation	Complete README with setup and usage instructions	10
API Implementation	RESTful API endpoints (Bonus)	5
Deployment	Successful Linux deployment with documentation (Bonus)	5

Total Marks: 100 (including 10 bonus marks)

6. Submission Requirements

Students must submit the following deliverables:

- GitHub Repository Link: Public repository containing complete source code with proper commit history
- Design Document: PDF format or integrated in README with architecture and flow diagrams
- Application Screenshots: Minimum 4-5 screenshots demonstrating all major functionalities
- Setup Instructions: Step-by-step guide for Linux deployment and execution
- Requirements File: requirements.txt with all Python dependencies
- Sample Data: Seed data or SQL scripts for database initialization (if applicable)

7. Development Guidelines

- Follow PEP 8 coding standards for Python code
- Use meaningful variable and function names
- Implement proper error handling and input validation
- Write descriptive commit messages following conventional commit format
- Maintain a clean project structure with separate folders for templates, static files, and modules
- Include. gitignore file to exclude virtual environments and cache files
- Test the application thoroughly before submission
- Ensure the application runs on a fresh Linux installation following your documentation

8. Suggested Timeline

- **Week 1:** Requirements analysis, design documentation, and architecture planning
- **Week 2:** Core functionality implementation and database setup
- **Week 3:** UI development, testing, and bug fixes
- **Week 4:** Documentation, deployment, and final submission

9. Recommended Resources

- **Flask Official Documentation:** https://flask.palletsprojects.com/
- **SQLite Documentation:** https://www.sqlite.org/docs.html
- **Bootstrap Framework:** https://getbootstrap.com/
- **Git Documentation:** https://git-scm.com/doc
- **Python PEP 8 Style Guide**: https://pep8.org/
- **Jinja2 Template Designer Documentation**

Note: Academic integrity is expected. Plagiarism or unauthorized collaboration will result in zero marks. Ensure all external code sources are properly cited.