Minimal To-Do Application – SDLC & DevOps Report

Author: Your Name

Course: Your Course / Section

Date: October 2025

1. Introduction & Objective

The Minimal To-Do Application was developed to demonstrate the implementation of a small-scale CRUD system that can later be integrated into a DevOps pipeline. The application provides Create, Read, Update, and Delete functionality, persistent data storage using SQLite, and a minimal HTML frontend served by FastAPI. The project emphasizes clean code organization, modularity, and readiness for containerization and CI/CD automation.

2. SDLC Model & Justification

The Agile SDLC model was chosen for this project because it supports incremental delivery and flexibility. Agile enables continuous feedback, shorter development cycles, and fast adaptation—ideal for iterative development and future DevOps pipeline integration. Each iteration produced a working increment: backend CRUD, then frontend, then Docker/CI setup.

3. Requirements & User Stories

User Stories: 1. As a user, I can create a new task with a title. 2. As a user, I can view all existing tasks. 3. As a user, I can mark a task as complete or incomplete. 4. As a user, I can delete tasks I no longer need. Acceptance Criteria: - Tasks are persisted in SQLite. - Validation prevents empty task titles. - API responses follow consistent JSON format.

4. Architecture Overview

The architecture includes a FastAPI router layer, CRUD logic, ORM mapping, and a local SQLite database. A single-page HTML frontend communicates with the API via HTTP fetch requests. This separation supports modular testing and easy DevOps integration. (Architecture diagram should be attached in report.)

5. UML/Class Diagram

A single entity 'Task' represents the core data model with attributes: - id (int, PK) - title (string) - completed (bool) - created_at, updated_at (datetime) Mapped using SQLAlchemy ORM. (UML diagram attached.)

6. Implementation Summary

Backend: FastAPI (Python 3.11) Database: SQLite via SQLAlchemy ORM Frontend: Static HTML + JavaScript fetch API Endpoints: - GET /api/tasks - POST /api/tasks - PUT /api/tasks/{id} - DELETE /api/tasks/{id} - GET /api/health

7. Reflection: DevOps Adaptation

The app was prepared for DevOps scalability by adding a Dockerfile for containerization and a GitHub Actions workflow for continuous integration. The modular FastAPI architecture supports test automation, environment variables for config, and deployment to any container-based environment such as AWS ECS or Azure App Service.

8. Setup Instructions

1. Clone the repository from GitHub. 2. Create a virtual environment: - macOS/Linux: python3 -m venv .venv && source .venv/bin/activate - Windows: python -m venv .venv && .venv\Scripts\Activate.ps1 3. Install dependencies: pip install -r requirements.txt 4. Run locally: uvicorn app.main:app --reload 5. Visit http://127.0.0.1:8000/ for the UI and /docs for API docs.