Task Manager

Final Project Group 3

Diogo Benetti Fae Rodrigues - CS 7319 - On Campus

Nick Wall - CS 7319 - On Campus

## Overview

This project is a Task Management system that enables users to interact with lists of tasks. Tasks have a name, description, completion status, potentially a parent task, and an associated list. Lists have a name, a description, and tasks. This enables tasks to be organized into lists and have a hierarchy. This implementation is done in two applications: an MVC architecture and a Pipe and Filter architecture. Both implementations enable users to create and view a list of tasks, create/edit/delete tasks, edit task hierarchies, and all data is persisted between sessions using SQLite.

## Installation and Execution

This project is implemented in Python using no external runtime dependencies. All dependencies are development dependencies used for code quality, such as linting. This was managed using the poetry package manager. However, to install and run this application, nothing is needed except for a Python installation. This project has only been validated to work with versions over 3.9, but all LTS versions should work.

If you do not have Python installed, you can download it [here](https://www.python.org/downloads/). Download the installer, follow the steps in the wizard to install the Python runtime, and ensure you enable the option to add Python to your path.

With Python installed, you can clone the repository with the following command:

git clone \

<https://github.com/nicholaswall/CS5-7319-Final-Project-Group-03-Nick-Wall_Diogo-Rodrigues.git>

With the repository clone, you can enter the cloned repository by navigating into the directory. On unix-based systems, this is done as follows:

cd CS4-7319-Final-Project-Group-03-Nick-Wall-Diogo-Rodrigues

Inside the cloned repository, there are two projects, the MVC and the Pipe & Filter implementations. They are in the Selected and Unselected folders. To interact with the CLIs, they both follow the same convention.

python Selected/src/main.py -h

python Unselected/src/main.py -h

Running the different CLIs with the standard -h flag convention will open a help menu to display all of the commands and their required arguments. There is no compilation step, no dependency installs, and no unconventional CLI interfaces.

## Architectural Differences

There are many design differences between the two systems, MVC and Pipe & Filter. The MVC design ensures all operations flow through a controller that manages the workflow. This means that no arguments are needed at the CLI execution. The Pipe & Filter expects arguments to be passed to the command upfront. This is a large distinction as the MVC paradigm is much more imperative as the controller orchestrates the state. At the same time, the Pipe & Filter is more declarative and purely applies operations to the input state and returns an output state. Pipe & Filter does this by modeling each operation as a Command, which executes all of the necessary steps, so each operation maps to a single Command. The MVC architecture instead maps different operations to either the Tasks or Lists Controller, which delegates different functions for the operation. The Pipe and filter design, as a result, is highly composable, while MVC hides the implementation of repeated functions. Both implementations share the same logic and utilities for interacting with the database however, the MVC implements models while the Pipe and Filter design uses a functional style. The argument parsers for the CLI are a shared component, but other than that; the components are very different despite implementing similar logic.

## Architecture Changes

We had to change one of our architecture designs from the proposal after getting feedback. Having made the mistake of selecting an invalid architecture choice (Object-Oriented), we decided to do MVC as it is an extremely common architecture that mapped well to the problem and would give us valuable experience in developing it.