CSC 249 Final Project Document
Student Name: Eyob Ollivierre
Course: CSC 249 – Data Structures & Algorithms
Project Title: Office Task Manager
Date: April 2025

1. Project Overview

The Office Task Manager is a console-based application designed to help users manage their daily responsibilities such as submitting assignments, attending meetings, or completing follow-up tasks. It allows users to:

- Add new tasks with due dates and priorities
- Assign auto-generated task IDs
- Mark tasks as complete
- View all tasks or only completed ones
- Retrieve the most urgent incomplete task based on priority and due date

The system is entirely interactive, with error handling for date and priority input. It's built to be intuitive and efficient for office environments, students, or anyone managing multiple tasks.

2. Key Capabilities

The application meets the following functional requirements:

- Add new tasks by inputting a title, due date, and priority
- Automatically generate task IDs (e.g., T1, T2)
- Accept multiple date formats and validate inputs
- Prioritize tasks using a min-heap (priority queue)
- Mark tasks as complete using their task ID

- Display all tasks with ID, due date, priority, and status
- Display completed tasks in the order they were finished
- Use clear status indicators (✓ for complete, X for incomplete)

3. Data Structure Analysis

This project uses three key data structures:

A. Min-Heap (via heapq)

- Used to retrieve the most urgent task efficiently
- Big O Notation:
    - Insert: O(log n)
    - Retrieve Min: O(1)
    - Remove Min: O(log n)
- Justification: Keeps the highest-priority, earliest-due task on top automatically

B. Hash Map (via Python dict)

- Used to store and retrieve tasks by their unique ID
- Big O Notation:
    - Insert: O(1)
    - Lookup: O(1)
- Justification: Provides constant-time access to any task by ID

C. Linked List (via Python list)

- Used to keep track of completed tasks in order

- Big O Notation:
    - Append: O(1)
    - Iterate: O(n)
- Justification: Simple and effective for tracking the order in which tasks are completed

4. Comparison with Alternative Structure

Alternative Considered: Binary Search Tree (BST)

- Could be used to manage tasks by priority and due date
- However, a BST increases the risk of being unbalanced, leading to O(n) performance in the worst case
- A min-heap is simpler and purpose-built for managing urgency efficiently
- Final decision: Min-heap is more appropriate due to performance and simplicity

5. Use of Generative AI

Generative AI (ChatGPT) was used in the following ways:

- Brainstorming and narrowing the project concept to a task manager
- Helping organize and structure class design and functionality
- Suggesting refinements in input handling and overall readability
- Providing guidance on writing meaningful docstrings and inline comments for clarity

6. Deliverables

- task_structures.py: Contains the Task and TaskManager classes

- office_task_app.py: Interactive console interface using the above data structures
- Video presentation (submitted separately via Yuja)