TO_DO_LIST APP

Desktop application

PREPARED BY: Alaa Elfallah & Alaa Alarbi& Saja Mahmoud

Project Report: To-Do List Application Using Data Structures

This documentation provides a detailed explanation of the **To-Do List Application** implemented in Python using the tkinter library. The application allows users to manage tasks by adding, searching, deleting, and displaying tasks sorted by priority. Below, we explain the data structures, functionalities, framework choice, and challenges faced during implementation.

1.Data Structures Used

1.1 HashTable

The HashTable class is used to store tasks efficiently. It uses a **chaining** mechanism to handle collisions. Each task is stored as a Node object in a list, where the index is determined by a hash function.

- **Key Features:**
  a. Hashing Function: Converts the task description into an index using a custom hash function.

  b. Chaining: If a collision occurs, tasks are stored in a linked list at the same index.

  c. Operations:

     i. insert: Adds or updates a task.

     ii. delete: Removes a task.

     iii. get: Retrieves a task by description.

1.2 Binary Search Tree (BST)

a. The TaskManager class implements a **BST** to sort tasks by priority. Each node in the BST contains a Task object.

b. **Key Features:**

   i. **Priority-Based Sorting:** Tasks are inserted based on their priority value (High: 3, Medium: 2, Low: 1).

   ii. **Operations:**

1. insert: Adds a task to the BST.
2. delete: Removes a task from the BST.
3. _inorder_traversal: Retrieves tasks in sorted order (High to Low).

1.3 Linked List

The LinkedList class is used to store tasks in a linear fashion. Each node in the linked list contains a Node object.

1. **Key Features:**
   a. **Linear Storage:** Tasks are stored in a sequence, allowing for easy traversal.
   b. **Operations:**
      i. insert: Adds a task to the linked list.
      ii. delete: Removes a task from the linked list.
      iii. get_all_tasks: Retrieves all tasks in the linked list.

1.4 Task and Node Classes

1. Task Class:
   a. Represents a task with a description and priority.
   b. Provides a method (get_priority_value) to convert priority strings to numerical values.
2. Node Class:
   a. Represents a task in the HashTable and LinkedList with additional attributes like state (Completed/Uncompleted) and a next pointer for chaining.

2. Functionalities

1. 2.1 Adding a Task
2. Steps:
   a. The user enters a task description, selects a priority, and optionally selects a state.
   b. The task is added to both the HashTable and the BST and linkedList
   c. If the task already exists, it is updated with the new priority and state.

3. **Implementation:**
    a. add_task method in the App class handles user input and calls HashTable.insert and TaskManager.insert,and linkedList..

2.2 Searching for a Task

1. **Steps:**
    a. The user enters a task description.
    b. The HashTable.get method retrieves the task.
    c. If found, the task details (description, priority, state) are displayed.
2. **Implementation:**
    a. search_task method in the App class handles the search operation.

2.3 Deleting a Task

1. **Steps:**
    a. The user enters a task description.
    b. The task is removed from both the HashTable and the BST and linkedlist
    c. If the task does not exist, a "Task not found!" message is displayed.
2. **Implementation:**
    a. delete_task method in the App class handles the deletion.
3. **2.4 Displaying Tasks**
4. **Steps:**
    a. The user clicks the "Show Tasks" button.
    b. The LinkedList.get_all_tasks method retrieves all tasks.
    c. Tasks are displayed in a scrollable window.

- **Implementation:**
    - show_tasks method in the App class handles this functionality.

2.5 Displaying Tasks by Priority

1. **Steps:**
   a. The user clicks the "Priority" button.
   b. The TaskManager._inorder_traversal method retrieves tasks sorted by priority.
   c. Tasks are displayed in descending order of priority (High to Low).
2. **Implementation:**
   a. show_tasks_by_priority method in the App class handles this functionality.

3. Framework Choice

Why Tkinter?

1. **Simplicity:** tkinter is a built-in Python library, making it easy to use without additional installations.
2. **Cross-Platform:** Works on Windows, macOS, and Linux.
3. **Customizability:** Allows for the creation of custom GUI elements like buttons, entry fields, and combo boxes.
4. **Lightweight:** Suitable for small to medium-sized applications like this To-Do List.

Why HashTable, BST, and Linked List?

1. **HashTable:**
   a. Provides fast insertion, deletion, and lookup operations (O(1) average case).
   b. Efficiently handles task management with unique descriptions.
2. **BST:**
   a. Enables efficient sorting and retrieval of tasks by priority (O(log n) average case).
   b. Simplifies the display of tasks in priority order.
3. **Linked List:**
   a. Provides a simple and efficient way to store and retrieve all tasks in a linear fashion.

4. Challenges Faced

4.1 Handling Collisions in HashTable

- **Challenge:** Two tasks with different descriptions could hash to the same index.
- **Solution:** Implemented **chaining** to store multiple tasks at the same index using a linked list.

4.2 Synchronizing Data Structures

- **Challenge:** Ensuring that tasks are consistently added, updated, and deleted in all data structures (HashTable, BST, and LinkedList).
- **Solution:** Added logic to update all data structures in every operation (insert, delete).

4.3 User Input Validation

- **Challenge:** Ensuring that the user provides valid input (e.g., non-empty task descriptions).
- **Solution:** Added checks in the add_task, search_task, and delete_task methods to validate input.

4.4 Displaying Tasks

- **Challenge:** Ensuring that deleted tasks are not displayed in the task views.
- **Solution:** Added checks in the show_tasks and show_tasks_by_priority methods to verify if a task still exists in the HashTable.

5. Future Improvements

- **Rehashing:** Implement dynamic resizing of the HashTable to handle a growing number of tasks.
- **Persistence:** Save tasks to a file (e.g., JSON or CSV) to allow data persistence between sessions.
- **Enhanced GUI:** Add features like task editing, due dates, and categories for better task management.

To-Do List Application - User Guide

Welcome to the **To-Do List Application**! This application helps you manage your tasks efficiently by allowing you to add, search, delete, and view tasks sorted by priority. Below is a step-by-step guide on how to use the application.

Table of Contents

1. Getting Started

1. **Launch the Application:**
   - Run the Python script to start the application.
   - The application window will open, displaying the **To-Do List** interface.
2. **Interface Overview:**
   - **Task Entry Field:** Enter your task description here.
   - **Priority Dropdown:** Select the priority of the task (High, Medium, Low).
   - **State Dropdown:** Select the state of the task (Completed, Uncompleted).
   - **Buttons:**
     - **Add:** Add a new task.
     - **Search:** Search for an existing task.
     - **Delete:** Delete a task.
     - **Priority:** View tasks sorted by priority.
     - **Show Tasks:** View all tasks in a list.

2. Adding a Task

1. **Enter Task Description:**
   - Click inside the "**Enter your task:**" field and type the task description.
     - Example: Complete project report
2. **Select Priority:**
   - Click the "**Select Priority**" dropdown menu and choose a priority:
     - **High**
     - **Medium**
     - **Low**
     - If no priority is selected, the default is **Low**.
3. **Select State (Optional):**
   - Click the "**Select State**" dropdown menu and choose a state:
     - **Completed**
     - **Uncompleted**
     - If no state is selected, the default is **Uncompleted**.
4. **Add the Task:**
   - Click the "**Add**" button.
   - A confirmation message will appear, showing the task details.

---

3. Searching for a Task

1. **Enter Task Description:**
   - Click inside the "**Enter your task:**" field and type the task description you want to search for.
     - Example: Complete project report
2. **Search the Task:**

   - Click the "**Search**" button.
   - If the task is found, its details (description, priority, state) will be displayed.
   - If the task is not found, a message will indicate that the task does not exist.

---

4. Deleting a Task

1. **Enter Task Description:**
   - Click inside the "**Enter your task:**" field and type the task description you want to delete.
     - Example: Complete project report
2. **Delete the Task:**
   - Click the "**Delete**" button.
   - If the task is found, it will be deleted, and a confirmation message will appear.
   - If the task is not found, a message will indicate that the task does not exist.

---

5. Viewing All Tasks

1. **View Tasks:**
   - Click the "**Show Tasks**" button.
   - A new window will open, displaying all tasks in a scrollable list.

---

6. Viewing Tasks by Priority

1. **View Tasks by Priority:**
   - Click the "**Priority**" button.
   - A new window will open, displaying tasks sorted by priority (High to Low).
   - Each task will show its description and priority.

Team Contributions:

- **Alaa Elfallah**: Implemented the HashTable and integrated it with the GUI .
- **Alaa Alarbi**: Implemented the BST and integrated it with the GUI .
- **Saja Mahmoud**: Implemented the LinkedList, integrated it with the GUI.