TASK NO 6

Digital empowerment network

Objective:

Enhanced To-Do List Application in Python, fulfilling all the specified requirements. The application will be a console-based interactive menu, and it will store tasks in a list of dictionaries. Additionally, we'll implement file operations to save and load tasks.

Step 1: Application Structure

. The main tasks we need to implement are:

- Add task
- View tasks
- Remove task
- Mark task as completed
- Edit task
- Search task by keyword
- Filter tasks (pending/completed)
- Clear all tasks with confirmation
- Sort tasks by ID or status

Step 2: Using a List of Dictionaries

We'll use a list to store tasks, where each task is a dictionary with the following keys:

- `id`: Unique identifier for the task.
- `description`: A brief description of the task.
- `status`: Either `pending` or `completed`.

Step 3: Implementing Each Functionality

Python code to implement the To-Do list:

```
import json

tasks = []

def load_tasks():
    try:
        with open('tasks.json', 'r') as file:
            global tasks
            tasks = json.load(file)
    except FileNotFoundError:
        tasks = []

# Save tasks to file
def save_tasks():
```

```
    with open('tasks.json', 'w') as file:
        json.dump(tasks, file)

# Add a new task
def add_task(description):
    task_id = len(tasks) + 1
    task = {'id': task_id, 'description': description, 'status': 'pending'}
    tasks.append(task)
    print(f"Task '{description}' added with ID: {task_id}")
    save_tasks()

# View all tasks
def view_tasks():
    if tasks:
        print("\nTo-Do List:")
        for task in tasks:
            print(f"ID: {task['id']} | Description: {task['description']} |
Status: {task['status']}")
    else:
        print("\nNo tasks found.")

# Remove task by ID
def remove_task(task_id):
    global tasks
    tasks = [task for task in tasks if task['id'] != task_id]
    print(f"Task ID {task_id} removed.")
    save_tasks()

# Mark task as completed
def mark_completed(task_id):
    for task in tasks:
        if task['id'] == task_id:
            task['status'] = 'completed'
            print(f"Task ID {task_id} marked as completed.")
            save_tasks()
            return
    print(f"Task ID {task_id} not found.")

# Edit task description
def edit_task(task_id, new_description):
    for task in tasks:
        if task['id'] == task_id:
            task['description'] = new_description
            print(f"Task ID {task_id} description updated.")
            save_tasks()
            return
    print(f"Task ID {task_id} not found.")

# Search tasks by keyword
def search_task(keyword):
    found_tasks = [task for task in tasks if keyword.lower() in
task['description'].lower()]
    if found_tasks:
        print("\nSearch Results:")
        for task in found_tasks:
            print(f"ID: {task['id']} | Description: {task['description']} |
Status: {task['status']}")
```

```
    else:
        print(f"No tasks found with keyword: {keyword}")

# Filter tasks by status
def filter_tasks(status):
    filtered_tasks = [task for task in tasks if task['status'] == status]
    if filtered_tasks:
        print(f"\n{status.capitalize()} Tasks:")
        for task in filtered_tasks:
            print(f"ID: {task['id']} | Description: {task['description']} |
Status: {task['status']}")
    else:
        print(f"No {status} tasks found.")

# Clear all tasks with confirmation
def clear_all_tasks():
    confirmation = input("Are you sure you want to clear all tasks? (yes/no):
").lower()
    if confirmation == 'yes':
        global tasks
        tasks = []
        print("All tasks cleared.")
        save_tasks()
    else:
        print("Task clearing cancelled.")

# Sort tasks by ID or status
def sort_tasks(by='id'):
    if by == 'id':
        sorted_tasks = sorted(tasks, key=lambda x: x['id'])
    elif by == 'status':
        sorted_tasks = sorted(tasks, key=lambda x: x['status'])
    print(f"\nTasks sorted by {by}:")
    for task in sorted_tasks:
        print(f"ID: {task['id']} | Description: {task['description']} |
Status: {task['status']}")

# Menu for user interaction
def show_menu():
    print("\n--- To-Do List Menu ---")
    print("1. Add Task")
    print("2. View Tasks")
    print("3. Remove Task")
    print("4. Mark Task as Completed")
    print("5. Edit Task")
    print("6. Search Task")
    print("7. Filter Tasks (pending/completed)")
    print("8. Clear All Tasks")
    print("9. Sort Tasks")
    print("0. Exit")

# Main program loop
def main():
    load_tasks()
    while True:
        show_menu()
        choice = input("\nEnter your choice: ")
```

```
        if choice == '1':
            description = input("Enter task description: ")
            add_task(description)
        elif choice == '2':
            view_tasks()
        elif choice == '3':
            task_id = int(input("Enter task ID to remove: "))
            remove_task(task_id)
        elif choice == '4':
            task_id = int(input("Enter task ID to mark as completed: "))
            mark_completed(task_id)
        elif choice == '5':
            task_id = int(input("Enter task ID to edit: "))
            new_description = input("Enter new description: ")
            edit_task(task_id, new_description)
        elif choice == '6':
            keyword = input("Enter keyword to search: ")
            search_task(keyword)
        elif choice == '7':
            status = input("Enter status to filter (pending/completed):
").lower()
            filter_tasks(status)
        elif choice == '8':
            clear_all_tasks()
        elif choice == '9':
            sort_by = input("Sort by 'id' or 'status': ").lower()
            sort_tasks(sort_by)
        elif choice == '0':
            print("Exiting To-Do List. Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

How the Application Works:

1. **Add Task**: The user enters a task description, and a unique ID is assigned automatically.
2. **View Tasks**: Displays all tasks with their details (ID, description, and status).
3. **Remove Task**: Removes a task based on its unique ID.
4. **Mark as Completed**: Changes the status of a task to 'completed'.
5. **Edit Task**: Allows the user to edit the task's description.
6. **Search Task**: Finds tasks with a specific keyword in the description.
7. **Filter Tasks**: Displays tasks that are either pending or completed.
8. **Clear All Tasks**: Asks for confirmation before clearing the entire task list.
9. **Sort Tasks**: Sorts the task list by either ID or status.

Next Steps:

- We can customize the interface further.
- Add error handling for invalid inputs.