Task Manager System

Objective:

The primary goal of this project is to create a simple, text-based task management system that allows users to register, login, and manage tasks. The tasks can be added, viewed, marked as complete, or deleted, and all of this information is stored persistently in CSV files. The system ensures that users can efficiently manage their tasks, track their progress, and perform basic task-related actions.

Technologies Used:

- **Python**: The programming language used for the entire project.
- **Text Files** (user.db and task.csv): Used for storing user information and task data, ensuring persistence across program runs.

System Overview:

The system involves two main components:

1. **User Registration and Authentication**: Users can create an account and log in to access their tasks.
2. **Task Management**: Users can add, view, mark as complete, and delete tasks. Tasks are stored in a CSV file associated with a user ID.

Key Features:

1. **User Registration**:
    - Users can register with a unique username and password.
    - User information is stored in a user.db file, where each entry contains a unique ID, username, and password.
2. **Login**:
    - Users need to log in with their username and password.
    - The system verifies the credentials by checking the user.db file.
3. **Task Management**:
    - **Add Task**: Users can add new tasks with descriptions, which will be marked as "INCOMPLETE."
    - **View Tasks**: Users can view their list of tasks with details (task ID, description, status).
    - **Mark Task as Complete**: Users can mark tasks as complete by their task ID.
    - **Delete Task**: Users can delete tasks by their task ID.
    - Tasks are stored in a CSV file (task.csv), with each task associated with a user ID.

4. **Persistent Storage**:
    - The system uses text files (user.db and task.csv) to store data persistently.
    - If these files don't exist, the system automatically creates them.

Detailed Flow:

1. User Registration:

When a new user wishes to register, the system asks for a username and password. These details are saved in the user.db file in the format user_id~username~password.

2. Login:

When a user logs in, the system prompts for the username and password. It checks the user.db file for matching credentials. If a match is found, the user is granted access to the task manager system. Otherwise, an error message is displayed.

3. Task Management:

Once logged in, users can interact with their tasks:

- **Adding Tasks**: Users can create tasks with descriptions. New tasks are saved in task.csv with the format user_id~task_description~status. Initially, tasks are set to "INCOMPLETE."
- **Viewing Tasks**: Users can view all their tasks with their status (either INCOMPLETE or COMPLETE).
- **Marking Tasks Complete**: Users can mark a task as completed by entering its task ID.
- **Deleting Tasks**: Users can delete tasks by selecting the task ID.

4. Data Persistence:

- **User Data**: Stored in user.db, where each line contains a unique user ID, username, and password.
- **Task Data**: Stored in task.csv, where each line contains the user ID, task description, and task status.
- The system reads and writes data to these files whenever tasks or user information are updated.

File Structure:

1. **user.db**: Stores user information.

CopyEdit

1~john_doe~password123

2~jane_doe~mypassword

2. **task.csv**: Stores task information associated with user IDs.

CopyEdit

1~Complete project~INCOMPLETE

1~Buy groceries~INCOMPLETE

Functions in the System:

1. **main_menu()**:
    - Displays the main menu, which allows the user to either login or register.
2. **login_prompt()**:
    - Prompts the user to enter a username and password for login.
3. **register_user()**:
    - Registers a new user by appending the user's information to the user.db file.
4. **add_task()**:
    - Prompts the user to add a task to their task list. The task is saved with the status "INCOMPLETE."
5. **view_task()**:
    - Displays all tasks associated with the currently logged-in user.
6. **mark_task()**:
    - Marks a specific task as "COMPLETE" by updating the task.csv file.
7. **delete_task()**:
    - Deletes a specific task based on its task ID.
8. **show_task_manager_menu()**:
    - Displays the menu for managing tasks (add, view, mark complete, delete, or exit).

Code Example:

Register a New User:

python

CopyEdit

```
def register_user(userName, password):
    userdb = open('user.db', 'a')
```

```
    id = find_total_user() + 1  # Increment user ID
    userdb.write(str(id) + "~" + userName + '~' + password + "\n")
    userdb.close()
    print('User Successfully Registered. Please log in...')
```

Add a Task:

python

CopyEdit

```
def add_task():
    task_details = input('Enter the task: ')
    task_db = open('task.csv', 'a')
    task_db.write(current_user[0] + "," + task_details + "," + "INCOMPLETE" + "\n")
    print('Task Added...')
    task_db.close()
```

Future Enhancements:

1. **Improved User Interface**: The current system is text-based. It can be enhanced with a graphical user interface (GUI) for better user interaction.
2. **Password Encryption**: Storing passwords in plain text (as done in the current system) is not secure. A future version could implement password encryption using libraries like bcrypt.
3. **Task Deadlines**: Users could set deadlines for tasks and receive notifications when a task is approaching its deadline.
4. **Data Backup**: The current system does not back up data. Implementing automatic backup mechanisms could prevent data loss.

Conclusion:

This Task Manager project provides a simple yet functional task management system. It demonstrates the use of file handling, user authentication, and basic task management features. While the project is text-based, it lays the foundation for a more advanced application with additional features like deadlines, priorities, and notifications.