Project Title : **SMART TASK ORGANIZER**

Done By : **SWATHI C**

Domain : **DATA ANALYTICS AND DATA SCIENCE**

Date Of Submission : 09/09/2024

1. Aim of the Project:

The main purpose of this project is to create an intuitive goal-tracking system that enables users to effectively manage daily chores and short-term goals while promoting productivity and personal development. The system attempts to assist users by:

a. **Organizing daily tasks** into fitness, learning, and hobby goals.

b. **Tracking progress** on each goal with an intuitive interface.

c. **Facilitating short-term goal management** with deadlines and progress tracking.

d. **Providing real-time feedback** on task completion and celebrating achievements.

e. **Centralizing personal goal management** in a single platform to improve overall time management, motivation, and accomplishment.

This project will serve as a reliable tool for individuals to track, manage, and complete both daily and short-term personal goals, fostering a disciplined and focused approach to personal development.

2. Problem Statement:

In today's fast-paced world, individuals struggle to efficiently manage their personal goals and daily tasks due to overwhelming responsibilities, distractions, and lack of proper organization. Many people set ambitious targets, such as fitness goals, learning new skills, or pursuing hobbies, but often fail to maintain consistent progress or stay motivated over time. This lack of goal tracking and time management leads to frustration, missed opportunities for personal growth, and an inability to balance multiple areas of life.

Key challenges include:

1. **Disorganization:** People often fail to structure their daily tasks, leading to confusion about priorities and inefficient use of time.

2. **Lack of Progress Visibility:** Without a clear method to monitor and measure progress, individuals can feel unmotivated or unaware of how close they are to completing their goals.

3. **Inefficient Task Management:** Traditional paper-based or fragmented digital methods don't provide real-time updates, reminders, or status reports, making it hard to stay on track.

4. **Neglect of Short-Term Goals:** While long-term ambitions are often considered, people tend to ignore or forget short-term goals that are critical to achieving incremental success.

5. **No System for Achievements:** Lack of feedback or recognition for accomplishments can diminish morale and cause people to give up on goals prematurely.

Thus, there is a need for a centralized, easy-to-use system that not only helps users plan and track their daily goals, but also manage short-term goals with deadlines, celebrate achievements, and provide consistent feedback to motivate users. The absence of such a system leads to poor personal management, missed goals, and a lack of consistent growth across important areas of life like fitness, learning, and hobbies.

3. Project Description:

The project involves developing a task managing system using Python, focusing on core functionalities that allow users to create, manage, and track both daily and short-term personal goals. The system aims to enhance user productivity by encouraging daily progress, offering real-time updates on goal completion, and providing a central platform to track personal development in key areas such as fitness, learning, and hobbies. Here are the main functionalities:

- **Daily Schedule Creation**
- **Goal Progress Tracking**
- **Short-Term Goal Management**
- **Priority-Based Goal Management**
- **Achievement System**
- **Centralized Display of All Goal**
- **Error Handling and Validation**

These functionalities will be implemented using Python programming language, with appropriate object-oriented principles. Core data structures like lists and dictionaries will be used to manage goal information and achievements. The system will be modular, with distinct classes representing different types of goals (fitness, learning, hobby, and short-term) and goal management (daily schedule, short-term goal tracking).

4. Functionalities:

- **Daily Schedule Creation**:
  - This functionality allows users to create a personalized daily schedule that includes goals from specific categories such as Fitness, Learning, and Hobby. Users can specify the number of goals they want to accomplish for the day. For each goal, users assign a priority and track progress throughout the day.
  - Why it's important: This ensures that users stay organized and focus on essential tasks, creating a productive routine.
- **Goal Progress Tracking**:
  - Users can update the progress of each goal during the day. The system validates progress between 0-100%, ensuring realistic tracking. Once a goal reaches 100%, it is automatically marked as complete, and the system celebrates the achievement.
  - Why it's important: This real-time progress tracking motivates users to stay on course and provides a clear view of how close they are to completing their tasks.
- **Short-Term Goal Management**:
  - Users can add, track, and update short-term goals with specific deadlines. This feature focuses on goals that extend beyond daily tasks, allowing users to manage larger objectives over a set period. When a goal reaches 100% progress, it is moved to the completed goals section.
  - Why it's important: Short-term goals encourage users to think beyond daily routines and work toward broader personal achievements, fostering long-term growth.
- **Priority-Based Goal Management**:
  - Users assign priorities (1 to 5) to each goal, indicating their importance. Higher-priority goals can be tackled first, helping users manage their time effectively by focusing on crucial tasks.

- Prioritization helps users focus on the most significant tasks, avoiding time wasted on lower-priority goals, which enhances time management and productivity.

- **Achievement System**:

  - Each completed goal is recorded with a timestamp in the achievement list. Users can revisit their accomplishments, and the system displays messages of encouragement upon goal completion.

  - Celebrating small wins motivates users to keep striving toward new goals and creates a sense of progress and personal development.

- **Centralized Goal Display**:

  - The system offers a centralized interface where users can view both daily goals and short-term goals. Progress updates and achievements are displayed clearly, making it easy to track ongoing goals and view what has been accomplished.

  - A clear view of all active goals in one place helps users stay organized, track their progress at a glance, and maintain focus on both daily and long-term objectives.

- **Completion Monitoring**:

  - The system continuously monitors the completion of all tasks and informs users when all daily goals are achieved. If all tasks are not completed, it offers options for managing short-term goals, ensuring that users are always engaged with either daily or longer-term tasks.

  - This feature keeps users accountable and ensures they remain focused on either finishing their daily tasks or making progress on short-term objectives.

These functionalities combined make the goal-tracking system an effective tool for enhancing personal productivity, keeping users organized, and promoting long-term growth through efficient goal management.

5. Error Handling:

Error Handling in simple words means making sure the system doesn't crash or break when something unexpected happens. It checks the information users give and makes sure it's correct before moving forward.

For example:

- ✓ If a user enters a progress percentage that’s not between 0 and 100, the system will show an error and ask the user to try again.
- ✓ If the deadline for a short-term goal is entered in the wrong format (like 2024/09/09 instead of 2024-09-09), the system will ask the user to fix it.

This way, the program catches mistakes and guides users to correct them, making the system run smoothly and without issues.

6. Code Implementation:

```
# Abstract base class 1
class Goal(ABC):

#Initialize a goal with a name, priority, progress, and achievements.
    def __init__(self, name, priority=1):
        self._name = name # Internal use
        self._progress = 0
        self._priority = self._validate_priority(priority)  # Validating and setting priority
        self._achievements = []  # List to store achievements with timestamps

    @staticmethod
    def _validate_priority(priority): #Ensure priority is between 1 and 5.
        if 1 <= priority <= 5:
            return priority
        else:
            raise ValueError("Priority must be between 1 and 5.")

    @staticmethod
    def _validate_progress(progress): #Ensure progress is between 0 and 100.
        return min(100, max(0, progress))

    @abstractmethod
    def update_progress(self, progress): #Abstract method to update progress; must be implemented in subclasses.
        pass

    @abstractmethod
    def display_status(self):  #Abstract method to display the status; must be implemented in subclasses.
        pass

    def add_achievement(self, achievement): #Add an achievement with a timestamp.
        timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')  # Get current timestamp
        self._achievements.append(f"{timestamp} - {achievement}")  # Append achievement with timestamp

    def __str__(self):  #Return a string representation of the goal's status.
        achievements = ', '.join(self._achievements) if self._achievements else 'None'  # Format achievements
        return (f"** {self._name} **\n"
                f"Progress: {self._progress}%\n"
                f"Priority: {self._priority}\n"
                f"Achievements: {achievements}")
```

```
# Subclass for fitness goals
class FitnessGoal(Goal):
    def update_progress(self, progress): #Update progress ensuring it's between 0 and 100.
        self._progress = self._validate_progress(progress)  # Use base class method to validate progress

    def display_status(self): #Display the status of the fitness goal.
        print(f"\n⛹️ Fitness Goal Status:\n{self}")  # Print goal status


# Subclass for learning goals
class LearningGoal(Goal):
    def update_progress(self, progress): #Update progress ensuring it's between 0 and 100.
        self._progress = self._validate_progress(progress)  # Use base class method to validate progress

    def display_status(self): #Display the status of the learning goal.
        print(f"\n📚 Learning Goal Status:\n{self}")  # Print goal status


# Subclass for hobby goals
class HobbyGoal(Goal):
    def update_progress(self, progress): #Update progress ensuring it's between 0 and 100.
        self._progress = self._validate_progress(progress)  # Use base class method to validate progress

    def display_status(self): #Display the status of the hobby goal.
        print(f"\n🎸 Hobby Goal Status:\n{self}")  # Print goal status


# Subclass for short-term goals with a deadline
class ShortTermGoal(Goal):
    def __init__(self, name, deadline, priority=1):
        super().__init__(name, priority)  # Initialize base class
        self._deadline = self._validate_deadline(deadline)  # Validate and set deadline

    @staticmethod
    def _validate_deadline(deadline):
        #Validate date format YYYY-MM-DD.
        try:
            datetime.strptime(deadline, '%Y-%m-%d')  # Check if date is in the correct format
            return deadline
        except ValueError:
            raise ValueError("Incorrect date format, should be YYYY-MM-DD.")

    def update_progress(self, progress):
```

```
        #Update progress ensuring it's between 0 and 100.
        self._progress = self._validate_progress(progress)  # Use base class method to validate progress

    def display_status(self):
        #Display the status of the short-term goal.
        print(f"\n📅 Short-Term Goal Status:\n{self}")  # Print goal status


# Abstract base class 2
class Updation(ABC):
    @abstractmethod
    def add_goal(self, goal):
        #Abstract method to add a goal; must be implemented in subclasses.
        pass

    @abstractmethod
    def display_goals(self):
        #Abstract method to display goals; must be implemented in subclasses.
        pass


# Class for managing daily schedule and goals
class DailySchedule(Updation):
    def __init__(self):
        self.goals = []  # List to store daily goals

    def add_goal(self, goal):
        #Add a goal to the daily schedule.
        self.goals.append(goal)  # Append goal to the list

    def display_goals(self):
        #Display all goals for the day in a more readable format.
        print("\n*** Today's Tasks ***")
        if not self.goals:
            print("No goals added to today's schedule.")  # Handle empty goal list
        else:
            for idx, goal in enumerate(self.goals, start=1):
                print(f"{idx}. {goal._name} | Priority: {goal._priority} | Progress: {goal._progress}%")

    def mark_task_completed(self, goal_name, progress): #Mark a task as completed and update its progress.
        goal_name = goal_name.lower()  # Convert goal name to lowercase for case-insensitive matching
        goal = next((g for g in self.goals if g._name.lower() == goal_name), None)  # Find goal
```

```
        if goal:
            goal.update_progress(progress)  # Update goal progress
            print(f"Updated progress for '{goal._name}': {goal._progress}%")
            if goal._progress == 100:
                print(f" 🎉 Congratulations! Task {goal._name} is completed successfully :)")
        else:
            print(f"Task '{goal_name}' not found in daily schedule.")  # Handle case where goal is not found

    def all_tasks_completed(self):
        #Check if all daily tasks are completed.
        return all(goal._progress == 100 for goal in self.goals)  # Return True if all goals are complete


# Class for managing short-term goals
class ShortTermGoalManager(Updation):
    def __init__(self):
        self._goals = []  # List to store short-term goals
        self._completed_goals = []  # List to store completed goals

    def add_goal(self, name, deadline):
        """Add a new short-term goal."""
        try:
            goal = ShortTermGoal(name, deadline)  # Create a new ShortTermGoal
            self._goals.append(goal)  # Add goal to the list
            print(f"Added new short-term goal: {name}")
        except ValueError as e:
            print(e)  # Print error message if date format is incorrect

    def display_goals(self):
        #Display all short-term goals and completed goals in a readable format.
        if self._goals:
            print("\n*** Short-Term Goals ***")
            for idx, goal in enumerate(self._goals, start=1):
                print(f"{idx}. {goal._name} | Progress: {goal._progress}% | Deadline: {goal._deadline}")
        else:
            print("\nNo short-term goals.")

        if self._completed_goals:
            print("\n 🎉 Completed Short-Term Goals:")
            for idx, goal in enumerate(self._completed_goals, start=1):
                print(f"{idx}. {goal._name} | Completed on {goal._achievements[-1].split(' - ')[0]}")
```

```
        self._goals = [goal for goal in self._goals if goal._progress < 100]  # Remove completed goals from the list

    def update_progress(self, name, progress):
        #Update progress for a short-term goal.
        goal_name = name.lower()  # Convert goal name to lowercase for case-insensitive matching
        goal = next((g for g in self._goals if g._name.lower() == goal_name), None)  # Find goal
        if goal:
            goal.update_progress(progress)  # Update goal progress
            print(f"Updated progress for '{goal._name}': {goal._progress}%")
            if goal._progress == 100:
                goal.add_achievement("Task completed successfully")  # Add achievement on completion
                print(f"Congratulations! Task {goal._name} is completed successfully :)")
        else:
            print(f"Task '{goal_name}' not found in short-term goals.")  # Handle case where goal is not found


def get_validated_input(prompt, validation_func):
    #Get validated input from the user.
    while True:
        try:
            value = validation_func(input(prompt))  # Get input and validate
            return value
        except ValueError as e:
            print(e)  # Print error message if validation fails

def main():
    daily_schedule = DailySchedule()  # Create daily schedule instance
    short_term_goal_manager = ShortTermGoalManager()  # Create short-term goal manager instance

    while True:
        print("\nMAIN MENU:")
        print("1. Create Daily Schedule")
        print("2. Manage Short-Term Goals")
        print("3. Display All Goals")
        print("4. Exit")

        option = input("Select an option: ").strip()  # Get option from user

        if option == "1":
            # Create daily schedule
            goal_count = int(input("How many goals do you want to schedule for today? "))
            for _ in range(goal_count):
```

```
            print("\nSelect Goal Type:")
            print("1. Fitness Goal")
            print("2. Learning Goal")
            print("3. Hobby Goal")
            goal_type = input("Enter choice: ").strip()

            name = input("Enter the goal name: ")
            priority = get_validated_input("Enter priority (1-5): ", lambda x: int(x) if 1 <= int(x) <= 5 else ValueError("Priority must be between 1 and 5."))

            if goal_type == "1":
                daily_schedule.add_goal(FitnessGoal(name, priority))  # Add fitness goal
            elif goal_type == "2":
                daily_schedule.add_goal(LearningGoal(name, priority))  # Add learning goal
            elif goal_type == "3":
                daily_schedule.add_goal(HobbyGoal(name, priority))  # Add hobby goal
            else:
                print("Invalid choice.")

        # Sort goals by priority before asking for progress
        daily_schedule.goals.sort(key=lambda g: g._priority)

        # After adding goals, prompt to update their progress in priority order
        for goal in daily_schedule.goals:
            progress = get_validated_input(f"Enter progress for '{goal._name}' (Priority: {goal._priority}) (0-100): ", lambda x: int(x) if 0 <= int(x) <= 100 else ValueError("Progress must be between 0 and 100."))
            daily_schedule.mark_task_completed(goal._name, progress)

        # Check if all tasks are completed
        if not daily_schedule.all_tasks_completed():
            # If not, remind the user and ask if they want to manage short-term goals
            while True:
                choice = input("Not all tasks are done. Do you want to manage short-term goals anyway? (yes/no): ").strip().lower()
                if choice == 'yes':
                    # Move to short-term goals management
                    break
                elif choice == 'no':
                    # If no, redirect to complete pending tasks
                    print("Please complete all pending tasks first.")
                    for goal in daily_schedule.goals:
                        if goal._progress < 100:
                            progress = get_validated_input(f"Enter progress for '{goal._name}' (Priority: {goal._priority}) (0-100): ", lambda x: int(x) if 0 <= int(x) <= 100 else ValueError("Progress must be between 0 and 100."))
```

```
                    daily_schedule.mark_task_completed(goal._name, progress)
                # Recheck if all tasks are completed after the second update
                if daily_schedule.all_tasks_completed():
                    print("All tasks completed. Returning to the main menu...")
                break
            else:
                print("Invalid input. Please enter 'yes' or 'no'.")

    elif option == "2":
        # Manage short-term goals
        while True:
            print("\nSHORT-TERM GOALS MENU:")
            print("1. Add a New Short-Term Goal")
            print("2. Update Progress for a Short-Term Goal")
            print("3. Display Short-Term Goals")
            print("4. Exit to Main Menu")

            sub_option = input("Select an option: ").strip()

            if sub_option == "1":
                name = input("Enter the goal name: ")
                deadline = input("Enter the deadline (YYYY-MM-DD): ")
                short_term_goal_manager.add_goal(name, deadline)  # Add new short-term goal
            elif sub_option == "2":
                name = input("Enter the name of the goal: ")
                progress = get_validated_input("Enter progress (0-100): ", lambda x: int(x) if 0 <=
int(x) <= 100 else ValueError("Progress must be between 0 and 100."))
                short_term_goal_manager.update_progress(name, progress)  # Update progress for
short-term goal
            elif sub_option == "3":
                short_term_goal_manager.display_goals()  # Display short-term goals
            elif sub_option == "4":
                break
            else:
                print("Invalid option. Please choose again.")

    elif option == "3":
        # Display all goals
        print("\nDisplaying all goals...")
        daily_schedule.display_goals()
        short_term_goal_manager.display_goals()

    elif option == "4":
        print("Exiting the program. Have a nice Day :)")
        break
    else:
```

```
            print("Invalid option. Please choose again.")


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

7. Results and Outcomes:

Through the implementation of the goal-tracking system, we successfully streamlined the process of managing daily tasks and short-term goals. Users experienced a more structured approach to tracking their fitness, learning, and hobby goals, resulting in better personal discipline and time management. The system provided clear progress updates, ensuring tasks were completed efficiently, and goals were achieved within set priorities. Additionally, short-term goals were managed effectively, leading to an organized workflow for larger objectives. This ultimately improved user productivity and accountability while offering a sense of accomplishment when tasks were completed.

```
PS C:\Users\swathi\OneDrive\Desktop\smart task organizer> python main.py

MAIN MENU:
1. Create Daily Schedule
2. Manage Short-Term Goals
3. Display All Goals
4. Exit
Select an option: 1
How many goals do you want to schedule for today? 2

Select Goal Type:
1. Fitness Goal
2. Learning Goal
3. Hobby Goal
Enter choice: 1
Enter the goal name: exercise
Enter priority (1-5): 2

Select Goal Type:
1. Fitness Goal
2. Learning Goal
3. Hobby Goal
Enter choice: 2
Enter the goal name: oops
Enter priority (1-5): 1
Enter progress for 'oops' (Priority: 1) (0-100): 100

Updated progress for 'oops': 100%


🎉Congratulations! Task oops is completed successfully :)

Enter progress for 'exercise' (Priority: 2) (0-100): 50

Updated progress for 'exercise': 50%

Not all tasks are done. Do you want to manage short-term goals anyway? (yes/no): no
Please complete all pending tasks first.
Enter progress for 'exercise' (Priority: 2) (0-100): 100
```

8. Conclusion:

In conclusion, the goal-tracking system developed offers a comprehensive and effective solution for individuals to manage their daily tasks and short-term goals. With its intuitive interface and customizable goal types, users can easily monitor their progress across various areas such as fitness, learning, and hobbies. The system promotes better time management and productivity, helping users stay organized and achieve their goals with enhanced accountability. Future enhancements may include features like goal recommendations based on past performance, progress analytics, and integration with third-party apps for seamless tracking and notifications.