1. Project Title:

Task Reminder Bot (Python)

2. Project Overview:

The Task Reminder Bot is a Python-based application designed to remind users about their pending tasks via desktop notifications or messaging platforms such as Telegram. It includes the ability to set reminders, track tasks, and notify users of upcoming tasks.

Main Features:

- Console-based user interface for task management.
- Task reminders via desktop notifications or Telegram.
- Persistent data storage using SQLite (for task history).
- Reports/logs of tasks added, completed, and notifications sent.

3. Technologies Used:

- **Python**: For the application's core logic and functionality.
- **Tkinter (Optional)**: For building the GUI of the task reminder bot.
- **SQLite (Optional)**: For storing task history persistently.
- **Telegram API (Optional)**: For sending task reminders via Telegram.
- **Pandas (Optional)**: For advanced report generation and data management.
- **Logging**: For generating logs and tracking task reminders.

4. System Design:

4.1 User Interface:

- **Console Interface**: A simple command-line interface for interacting with the bot, where users can:
  - Add tasks with due dates.
  - View pending and completed tasks.
  - Set reminders.
- **Optional GUI**: Tkinter-based graphical interface allowing users to manage tasks more intuitively with buttons and forms

4.2 Functionality:

- **Add Task**: Users can input tasks with details (e.g., task description, due date).

- **Task Reminder**: Users can opt to receive a reminder notification at a specified time.
- **Task Status**: Track the status of tasks (pending, completed, overdue).
- **Reminder Mechanism**: Desktop notifications or Telegram bot sends timely reminder
- **Reports**: Generate reports of tasks with history stored in SQLite.
- **Error Handling**: Handle invalid inputs gracefully (e.g., incorrect task format or invalid date).

5. Modules and Classes:

5.1 Main Module:

- **task_reminder.py**: The core script responsible for handling task operations, reminders, and notifications.

5.2 Task Class:

```
class Task:
    def __init__(self, 
task_description,due_date,reminder_time=None):
        self.task_description = task_description
        self.due_date = due_date
        self.reminder_time = reminder_time
        self.status = "Pending"

    def mark_complete(self):
        self.status = "Completed"

    def mark_overdue(self):
        self.status = "Overdue"
```

5.3 Notification Class:

```
class Notification:

def __init__(self, task):

    self.task = task



def send_reminder(self):

    # Code to send desktop notification

    # Or use Telegram API for messaging app

    Pass
```

5.4 SQLite Database Handler:

```
import sqlite3
```

```
class DBHandler:
    def __init__(self, db_name="tasks.db"):
        self.connection = sqlite3.connect(db_name)
        self.cursor = self.connection.cursor()
    def create_table(self):
    self.cursor.execute("CREATE TABLE IF NOT EXISTS tasks (description TEXT, due_date
TEXT,   status TEXT)")

   def save_task(self, task):
      self.cursor.execute("INSERT INTO tasks (description, due_date, status) VALUES (?, ?,
?)",
              (task.task_description, task.due_date, task.status))
    self.connection.commit()
```

6. System Architecture:

6.1 Flowchart:

A flowchart outlining the user interaction with the system:

- **Start → Input Task → Set Reminder (Optional) → Save Task to DB → Send Reminder (Notification/Telegram) → Update Task Status → End**

6.2 Database Design:

- **Table**: tasks
    - **Columns**: task_description, due_date, status

7. Functionality Overview:

7.1 Adding a Task:

- The user can add a task using a simple prompt in the console:

```
task_description = input("Enter the task description: ")
due_date = input("Enter the due date (YYYY-MM-DD): ")
task = Task(task_description, due_date)
```

7.2 Setting a Reminder:

After adding a task, users can specify a reminder time:

```
    reminder_time = input("Enter reminder time (HH:MM): ")
   task.reminder_time = reminder_time
```

7.3 Sending Notifications:

A scheduled reminder sends a desktop notification or a Telegram message using a bot:

```
def send_notification(task):
  if task.reminder_time == current_time:
      Notification(task).send_reminder()
```

7.4 Handling Errors:

The application handles invalid inputs by:

- Checking if the entered date is in the correct format.
- Handling exceptions like division by zero when adding, subtracting, or multiplying numbers in the case of errors.

8. Testing and Edge Cases:

- **Test Case 1**: User enters an invalid date format.
  - Expected Result: The program should prompt for the correct format.
- **Test Case 2**: User enters non-numeric input for task duration.
  - Expected Result: The system should reject the input and ask for numeric input.

9. Optional Features:

9.1 GUI with Tkinter:

The GUI allows users to:

- Add tasks via input fields.
- View tasks in a list box.
- Set reminders through a simple form.

9.2 Telegram Integration:

A Telegram bot sends messages to users to remind them of their tasks. The bot listens for updates using python-telegram-bot library.

9.3 SQLite Database:

Tasks are saved into a local SQLite database for persistent storage. The database can store task history, which is retrieved and displayed to the user when needed.

10. Reports and Logs:

The system generates reports of all the tasks and logs user actions for debugging and tracking:

- **Log File**: A log file stores each action performed by the user (e.g., task added, reminder sent).
- **Report Generation**: Use Pandas to export task data to a CSV or Excel file for reporting

11. Conclusion:

- The Task Reminder Bot offers a robust solution for task management with automatic reminders through desktop notifications or Telegram. This project demonstrates an understanding of core Python concepts, database management, and integration with external services (Telegram).

12. Project Files:

You can find the complete Python project files in the repository attached:

- task_reminder.py
- db_handler.py
- notification.py
- README.md
- requirements.txt

13. Instructions for Running the Application:

1. Clone the repository.

2. Install the required dependencies:

```
pip install -r requirements.txt
```

3. Run the application:

```
python task_reminder.py
```