SUMMARY OF CODE

🧠 Goal of the Application

This is a **desktop-based task manager** built using:

- **Tkinter** + **ttkbootstrap**: For a modern, user-friendly GUI
- **SQLite**: For storing tasks locally
- **SpaCy NLP**: To automatically assign task priority based on task description
- **Threading**: To run reminders in the background without freezing the GUI

✅ Detailed Explanation Section-by-Section

📦 1. Imports & NLP Setup

```
import tkinter as tk
from tkinter import ttk, messagebox
import sqlite3
import threading
import spacy
from datetime import datetime
import ttkbootstrap as tb
```

- tkinter: Basic GUI components
- ttk: Treeview widget for displaying tasks
- messagebox: Alert pop-ups
- sqlite3: Local database to save tasks
- threading: To run reminders in the background
- spacy: Natural Language Processing to detect task urgency
- datetime: Date comparison and validation
- ttkbootstrap: A prettier version of tkinter widgets

💡 Load NLP Model

```
try:
    nlp = spacy.load("en_core_web_sm")
except OSError:
    messagebox.showerror("Error", ...)
    exit()
```

- Loads the small English model en_core_web_sm.
- If not installed, shows an error and exits.

🧱 2. TaskScheduler Class

🔧 __init__(self, root)

```
def __init__(self, root):
```

- Initializes the main GUI window.
- Sets the theme (darkly).
- Calls:
    - init_database() – creates DB and table.
    - create_gui() – builds the interface.
    - Starts a **background thread** for reminders with check_tasks().

🗃️ init_database()

```
def init_database(self):
```

- Connects to tasks.db.
- Creates a tasks table with fields:
    - id, title, description, due_date, priority, status, ai_reason

---

🤖 ai_prioritize_task(description)

```
def ai_prioritize_task(self, description):
```

- Uses NLP to assign task priority:
    - If text contains words like “urgent”, “critical” → High
    - If it includes “soon”, “priority”, etc. → Medium
    - Else → Low

📅 validate_date(date_str)

def validate_date(self, date_str):

- Validates whether a date is in **YYYY-MM-DD** format.

🖥️ 3. User Interface: create_gui()

def create_gui(self):

- Builds all GUI components.

Components:

1. **Title Label**
    - Big bold heading: “AI Task Scheduler”
2. **Input Fields**
    - Title, Description, Due Date
3. **Action Buttons**
    - **Add Task**
    - **Modify Task**
    - **Mark as Done**
    - **Delete Task**
4. **Task Table (Treeview)**
    - Displays all tasks in rows.
    - Columns: ID, Title, Description, Due Date, Priority, Status
    - Data comes from the database.

➕ add_task()

def add_task(self):

- Gets input values.
- Validates title and due_date.
- Uses NLP to assign priority.
- Saves task into the database.
- Refreshes the task list.

📝 modify_task()

def modify_task(self):

- Selects a task from the Treeview.
- Gets new input values.
- Validates them.
- Updates the task in the database.
- Recalculates priority using NLP.

✅ mark_task_done()

def mark_task_done(self):

- Marks the selected task's status as **Completed**.

❌ delete_task()

def delete_task(self):

- Deletes the selected task from the database.

🔄 refresh_tasks()

def refresh_tasks(self):

- Clears and reloads all tasks in the Treeview.
- Orders tasks:
  - By **priority** (High → Medium → Low)
  - Then by **due date**

🔔 show_notification(message)

def show_notification(self, message):

- Uses after() to display a **popup reminder** from a background thread.
- messagebox.showwarning() shows the message.

🔁 check_tasks() – Reminders with Threading

def check_tasks(self):

What It Does:

- Runs **forever** in the background (while True)
- Every hour:
    1. Connects to the database
    2. Gets all pending tasks
    3. Compares due date with **today**
    4. If due today or overdue → shows a reminder

Thread-Safe Handling:

- Uses a **new SQLite connection** inside the thread (important!)
- Prevents the sqlite3.ProgrammingError by keeping DB usage within the same thread.

🚀 4. Main Program Entry

if __name__ == "__main__":

- Sets up the window using ttkbootstrap's "superhero" theme.
- Launches the TaskScheduler app.

🔚 Final Summary

Feature	**Description**
NLP Task Priority	Assigns High/Medium/Low based on keywords
Full CRUD	Add, Modify, Mark Done, Delete
Treeview Task Display	Shows all tasks sorted by priority and due date
Local Storage	Uses SQLite for persistence
Background Thread Alerts	Warns user if task is due or overdue
Modern UI	Uses ttkbootstrap for styling
