Title: Space Weather Tracker Chatbot

Student name	Registration no	section	Roll no
Shreya yadav	12321647	K23HS	RK23HSA25
Priti Kumari	12321504	K23HS	RK23HSA69

GITHUB LINK = https://github.com/y-Shreya21/Space-weather-tracker-

Abstract

The **Space Weather Tracker Chatbot** is a Python-based intelligent desktop application designed to educate and inform users about current space weather conditions, such as solar flares, coronal mass ejections, and geomagnetic storms. With the rise of satellite communication, GPS navigation, and manned space missions, real-time awareness of solar activities has become increasingly important. This project integrates NASA's DONKI (Database Of Notifications, Knowledge, and Information) API to fetch live alerts and data, translating technical information into understandable messages for general users. Built using the Tkinter library, the chatbot offers an interactive graphical user interface where users can type queries in natural language. The system is capable of handling errors gracefully and saving the chat history for future reference. This application not only demonstrates technical proficiency in software development and API integration but also addresses an essential real-world problem—space weather awareness—through an educational tool that is accessible, informative, and user-friendly.

1. Introduction

In recent years, the importance of monitoring space weather has grown significantly due to the increasing dependency on satellite communication, aviation systems, and technologies susceptible to solar activity. Space weather refers to the conditions and phenomena in space caused primarily by solar activities such as solar flares, coronal mass ejections (CMEs), and high-speed solar winds. These phenomena can disrupt satellite operations, GPS navigation, radio communications, and even power grids on Earth.

Despite the critical implications, space weather remains a relatively underrepresented topic among the general public. To address this knowledge gap and make space weather data more accessible and engaging, this project proposes a user-friendly desktop application—**Space Weather Tracker Chatbot**. Developed using Python and Tkinter for the GUI, and powered by NASA's DONKI API, the chatbot serves as an educational and informational tool. It provides real-time alerts, summaries of space weather events, and stores chat history for continued learning.

This project combines the fields of astronomy, computer science, and user experience design to demonstrate how space data can be utilized effectively through conversational interfaces. The chatbot is especially valuable for students, educators, and anyone interested in understanding the current conditions in near-Earth space.

2. Objective

The primary objective of this project is to develop a Python-based desktop chatbot that utilizes NASA's publicly available DONKI (Database Of Notifications, Knowledge, and Information) API to fetch and display real-time space weather alerts and notifications. The chatbot aims to:

- Provide users with up-to-date information about solar events, such as Coronal Mass Ejections (CMEs), Solar Flares, and High-Speed Streams (HSS).
- Deliver this information through an intuitive and interactive chat-based graphical user interface (GUI) built with Tkinter.

- Simulate a conversational experience where users can type natural language queries like "What's the latest space weather?" and receive meaningful, formatted responses.
- Store chat history locally for educational or reference purposes, enabling users to revisit past conversations.
- Ensure error-free communication by implementing proper exception handling for network failures or malformed API responses.
- Serve as an educational tool to raise awareness about the significance of space weather and its impact on Earth-based technologies and human activities.

3. Tools and Technologies Used

- - Programming Language: Python
- - Libraries: requests: For API communication
- - tkinter: To build the GUI
- - json, os: For file management
- - API Used: NASA DONKI (Database of Notifications, Knowledge, and Information)

4. Features Implemented

- **Graphical Interface using Tkinter**: A clean and intuitive user interface is built using Python's Tkinter library, which allows for easy user interaction.
- **Text Input Field**: Users can input queries or commands in a familiar chat-like format.
- **Scrollable Chat Area**: All chatbot conversations are displayed in a scrollable text area, simulating a natural flow of communication.
- **Real-Time Space Weather Alerts**: The chatbot fetches the latest space weather events from NASA's DONKI API and displays up to three of the most recent alerts.
- **Chat History Storage**: All user-bot conversations are saved in a JSON file so users can revisit their previous queries.
- **Chat History Retrieval**: Users can type "history" at any time to view their saved conversations.
- **Error Handling**: Robust handling for invalid input, empty API responses, or connection issues using try-except blocks.
- **Keyword Recognition**: The chatbot can detect specific keywords like "weather", "history", and "exit" to perform different actions.

5. How It Works

1. When the application is launched, a chatbot window opens and displays a friendly welcome message.
2. The user can enter queries such as “weather” to fetch the latest space weather information.
3. Upon detecting a relevant keyword, the chatbot sends a **GET request** to NASA's DONKI API using the requests library.
4. The API responds with data on recent space weather events, which the chatbot parses and formats into a user-friendly message.
5. If no recent alerts are available, the chatbot informs the user accordingly.

6. Every interaction (user input and chatbot response) is saved into a local JSON file named chat_history.json.
7. Users can type "history" to view their previous conversations or "exit" to close the application gracefully, with all chat logs preserved.

6. NASA API Integration

- **API Endpoint Used**:
  https://api.nasa.gov/DONKI/notifications?api_key=wdfC0b3cpKatbQCZnDPmPLgejUurgZBOUP4p2XpJ

- **Data Retrieval**:
  The chatbot makes a GET request to the DONKI API and retrieves a list of recent alerts. The system selects and displays up to three of the most recent entries.

- **Alert Details Extracted**:

  - messageType: Indicates the type of alert (e.g., Solar Flare, CME).

  - messageBody: Contains a detailed description of the event.

- **Resilience**:
  The chatbot is programmed with proper error handling using try-except to catch:

  - API connection failures

  - Invalid JSON responses

  - Empty or unexpected data formats

- **Security & Key Management**:
  The application uses a personal NASA API key. It is recommended to secure this key in a separate configuration file in production environments.

  Code Overview (Summary)

- **SpaceWeatherChatbot Class**: Handles the entire functionality including GUI layout, event handling, and API interaction.

- **get_space_weather()**: Connects to NASA's DONKI API and retrieves the latest alerts. Parses and formats the JSON response for display.

- **process_input()**: Processes user commands like "weather", "history", and "exit". Also handles fallback messages for unrecognized input.

- **save_chat_history() and load_chat_history()**: Save the conversation to a local JSON file and reload it when the chatbot is restarted. This ensures continuity of user interaction.

8. Challenges Faced

- **Inconsistent or Empty API Responses**: Sometimes the NASA API may not return any data or may return incomplete fields, requiring robust error handling.
- **Long Message Formatting**: Some space weather alerts contain large chunks of text, which had to be broken down and displayed clearly in the chat window.
- **Maintaining GUI Responsiveness**: While making network calls, the Tkinter window could freeze. Using try-except blocks and efficient coding prevented UI hangs during API fetches.
- **Keyword Parsing**: Ensuring the chatbot responds only to meaningful input while ignoring irrelevant or partial user messages.

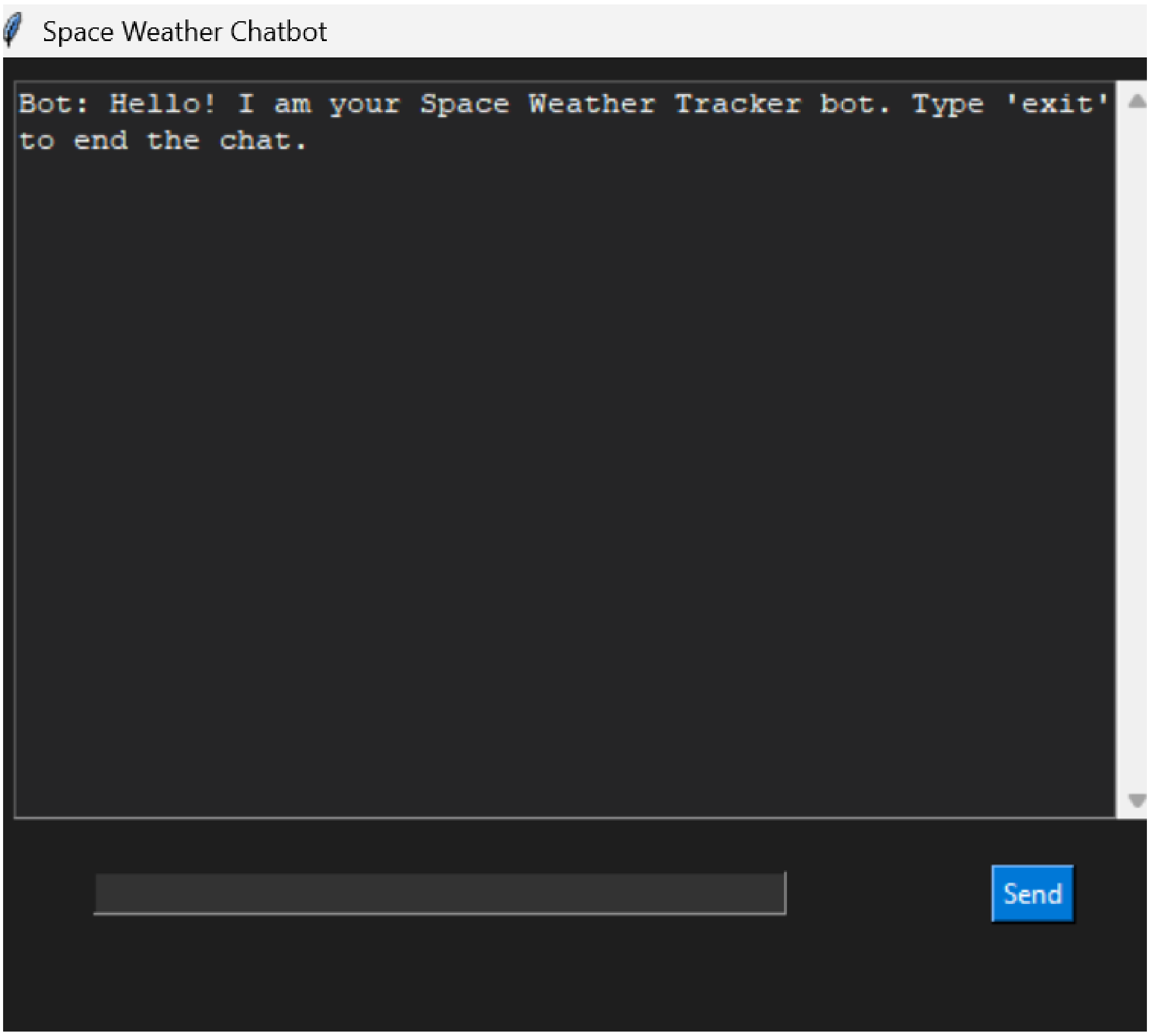
9. Future Enhancements

- **Advanced Filtering**: Add input fields for date ranges, location, or event types for more customized queries.
- **Graphical Data**: Include real-time graphs or visuals for sunspots, CMEs (Coronal Mass Ejections), and solar flares.
- **Voice Interaction**: Integrate speech-to-text and text-to-speech for more natural interaction.
- **Multiple Data Sources**: Combine NASA's data with other APIs like NOAA (National Oceanic and Atmospheric Administration) and ESA (European Space Agency) for more comprehensive results.
- **Mobile Compatibility**: Develop a mobile-friendly or web-based version of the chatbot in the future.

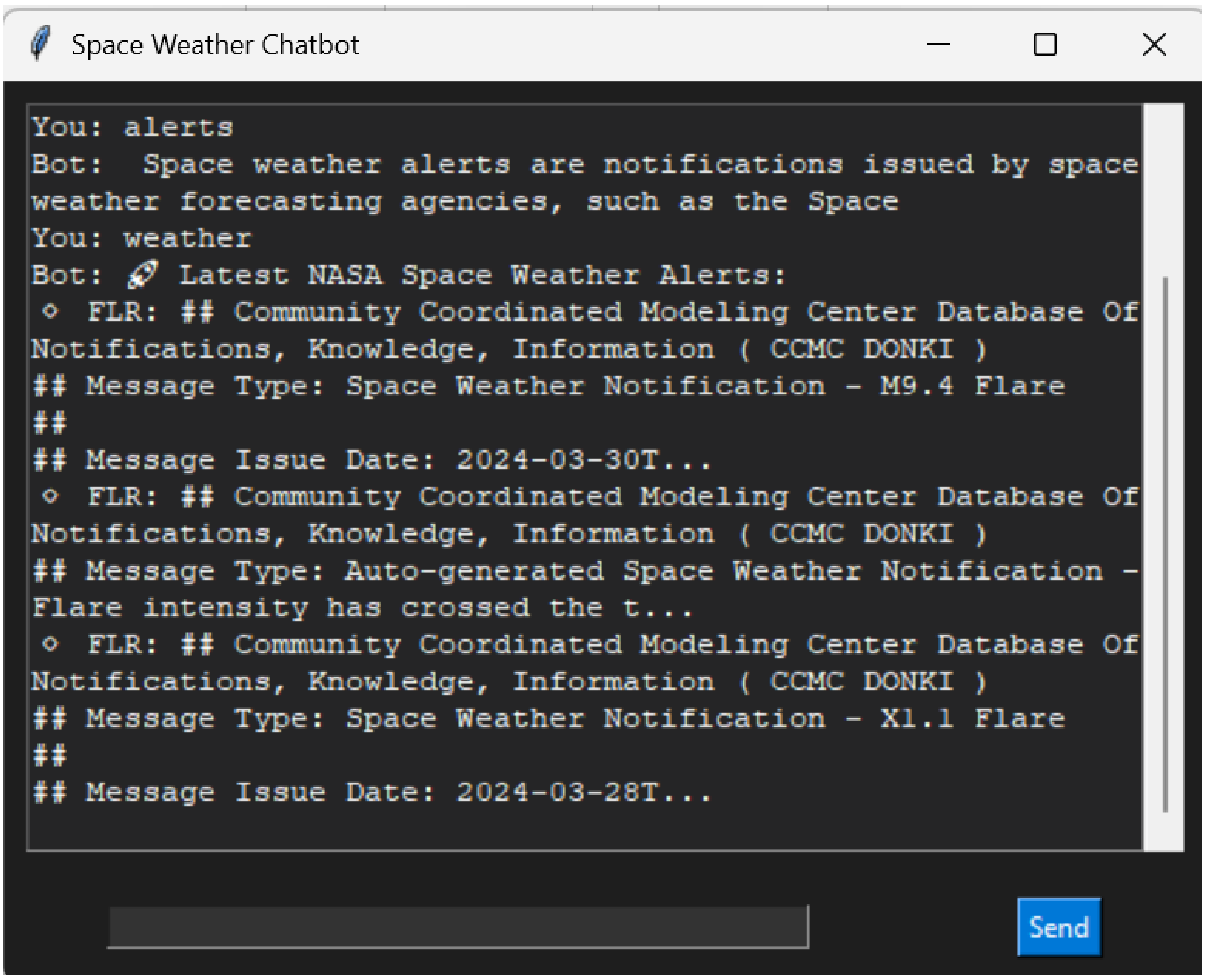
10. Conclusion

This project showcases the practical integration of real-time scientific data into a Python-based desktop application. It demonstrates how public APIs like NASA's DONKI can be used to educate users about space weather phenomena. The chatbot not only provides instant alerts but also serves as an educational tool for students and space enthusiasts. Its modular structure and clean UI make it adaptable for future development, whether for academic, scientific, or hobbyist use.

11. Screenshots



```
Bot: Hello! I am your Space Weather Tracker bot. Type 'exit'
to end the chat.
```



```
You: alerts
Bot:  Space weather alerts are notifications issued by space
weather forecasting agencies, such as the Space
You: weather
Bot: 🚀 Latest NASA Space Weather Alerts:
 ◦ FLR: ## Community Coordinated Modeling Center Database Of
Notifications, Knowledge, Information ( CCMC DONKI )
## Message Type: Space Weather Notification - M9.4 Flare
##
## Message Issue Date: 2024-03-30T...
 ◦ FLR: ## Community Coordinated Modeling Center Database Of
Notifications, Knowledge, Information ( CCMC DONKI )
## Message Type: Auto-generated Space Weather Notification -
Flare intensity has crossed the t...
 ◦ FLR: ## Community Coordinated Modeling Center Database Of
Notifications, Knowledge, Information ( CCMC DONKI )
## Message Type: Space Weather Notification - X1.1 Flare
##
## Message Issue Date: 2024-03-28T...
```

12. References

- NASA API Documentation: https://api.nasa.gov
- Tkinter GUI Library: https://docs.python.org/3/library/tkinter.html
- Python Requests Library: https://docs.python-requests.org
- NASA DONKI Overview: https://ccmc.gsfc.nasa.gov/donki/