
ThroneTalks – Game of Thrones Summarizer

Use cases: User interactions with the web application

User 1: Game of Thrones Enthusiasts/General Audience
Needs : Rewatch a particular Game of Thrones episode and summary of the previous as a recap

User2 : Discontinued the show for a couple of weeks and now wants to get a recap of all the previous seasons.
Needs : Quick summarization pf season or episodes to quickly understand or catch up on episodes

User 3: Screen writer/Content Creators
Needs : to research on the show to maybe create a similar show/ prequel or sequel using the episode summary generated.
Interaction method: Web application

User Action: The user selects the "season level summary" option and chooses a specific season from a dropdown list or similar input control.
System Response: The system retrieves and displays a summary of the selected season, including an overview of the main events, character developments, and any significant changes.

Python libraries – Gradio, Streamlit

Python Package Choices

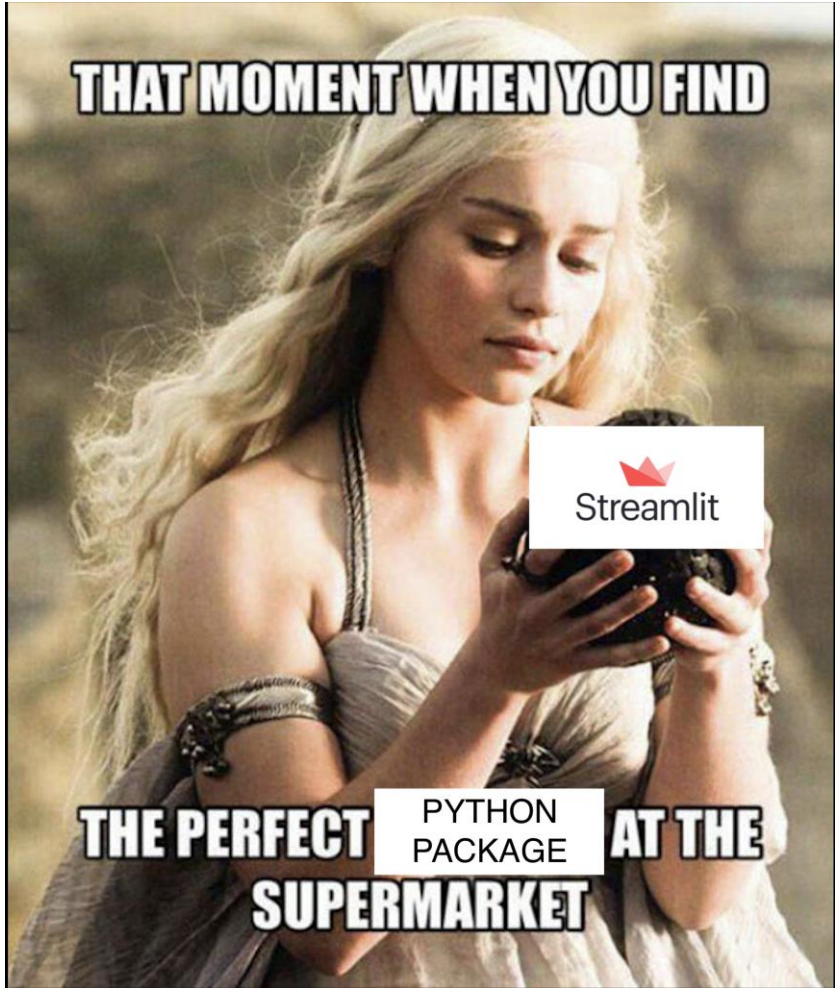
Gradio (MIT grad students) - It is an open-source Python package that allows users to quickly build a demo or web application for your machine learning model, API, or any arbitrary Python function.

Streamlit (Snowflake Inc.) - Streamlit enables quick transformation of Python scripts into interactive web apps, suitable for building dashboards, generating reports, or creating chat apps, with easy deployment via Community Cloud. It is favored for its simplicity, fast prototyping, live editing, and open-source nature, fostering a vibrant community.

Flask (Armin Ronacher) - Flask is a lightweight and flexible Python web application framework that enables the rapid development of simple to complex web applications. It provides developers with tools, libraries, and technologies to build web applications in an easy and efficient manner.
Note: Used for full scale production web applications, but this adds a lot of complexity.

Package Comparison

Parameters	Gradio	Streamlit
Community and Documentation	Growing community Not very extensive documentation as Streamlit.	Widely adopted in the Python community Extensive documentation
Customizations	Gradio offers a set of pre-built components for common use cases, but it is limited as compared to Streamlit.	Streamlit provides more extensive customization options for building highly tailored user interfaces.
Deployment Flexibility	Gradio provides deployment flexibility by allowing users to deploy web applications as either standalone web servers or REST APIs.	Streamlit Community Cloud can be used to deploy the web applications seamlessly and with less effort.
Backend Integration	Gradio's focus is primarily on integrating with machine learning libraries and visualizing model outputs.	Streamlit provides built-in support for common data science libraries such as Pandas, Matplotlib, Seaborn, Plotly, Altair, and many others.


THAT MOMENT WHEN YOU FIND
Streamlit
THE PERFECT
PYTHON PACKAGE
AT THE SUPERMARKET

Our Choice

We chose **Streamlit** for the interface

- We require an intuitive and simple interface for text summarization
- The primary reason is that being first time users of these library, we wanted **good documentation**.
- Streamlit has a **bigger community** than Gradio which will help us in getting solutions to issues we will face much easier.
- Gradio is primarily a demo tool whereas Stream Lit can be **productionized well for small web-applications**.
- Easy **deployment** and **integrates** well with GitHub.

Drawbacks/Remaining Concerns

Machine learning focus:

- Streamlit isn't specifically designed for machine learning models like Gradio is.
- Setting up interfaces for complex models might require more work.

Customization complexity:

- While offering more customization, it can also be overwhelming for beginners due to its complexity.
- This will be our primary concern.

Memory Concerns:

- Streamlit community cloud only gives us only 1gb memory for the application to run. If we were to run the model in community cloud we might run out of memory.
  - We plan to use Azure cloud and use APIs to integrate with the App.

Package Demonstration