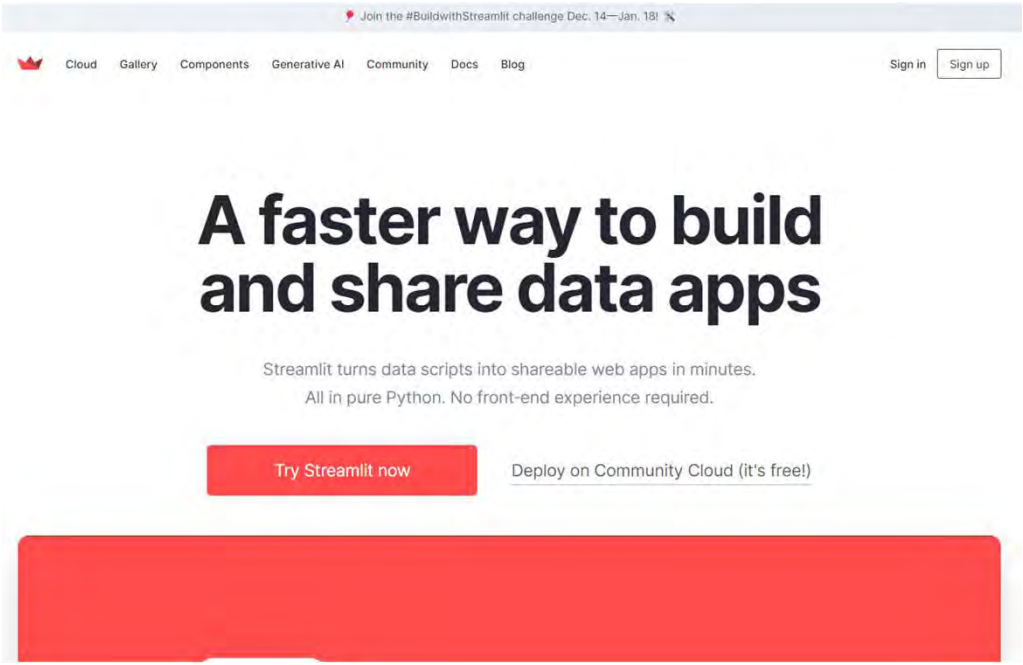
2025

Lesson 14 - Streamlit

Streamlit

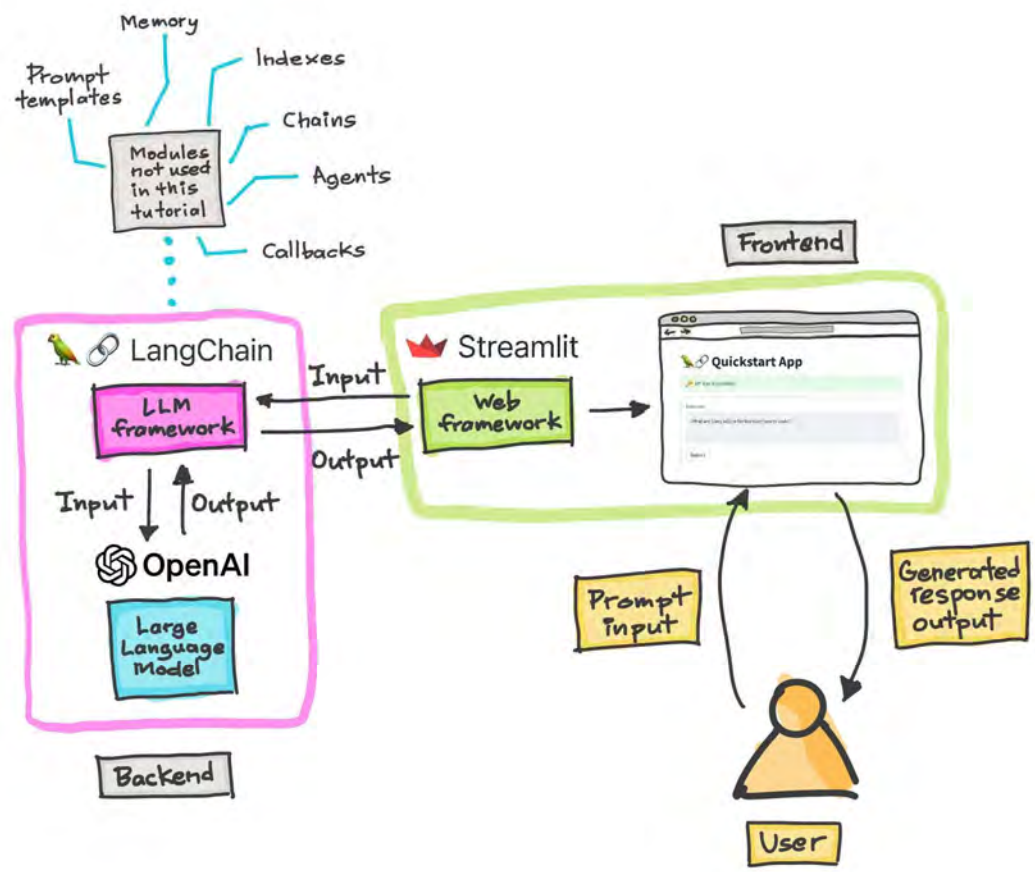
- an open-source Python library that is used to create and share beautiful, custom web apps for machine learning and data science.
- a tool that makes it easy for data scientists and machine learning engineers to turn data scripts into interactive web applications without requiring extensive knowledge of web development.



Streamlit

- **Ease of Use**: Streamlit allows you to create a web app with just a few lines of Python code. It's designed to be simple and intuitive, making it accessible even to those who are new to web development.
- **Rapid Prototyping**: With Streamlit, you can quickly turn data scripts into shareable web apps. This enables fast prototyping and iterative development, which is particularly useful in data science projects where results and visualizations need to be presented and modified frequently.
- **Interactive Widgets**: Streamlit includes a range of widgets like sliders, buttons, and text inputs, which can be used to interact with your data and models. This interactivity is key for exploring data and model outputs dynamically.
- **Data Visualization**: It integrates seamlessly with major data visualization libraries like Matplotlib, Plotly, and Altair. This makes it easy to include interactive charts and graphs in your apps.
- **Machine Learning Integration**: Streamlit is often used to showcase machine learning models. You can easily integrate your Python-based ML models and display their outputs, such as predictions and inferences.
- **Customizable Layouts**: While Streamlit's layout capabilities were initially basic, recent updates have added more flexibility in arranging the layout of your app, allowing for a more customized user interface.
- **Community and Ecosystem**: Being open-source, Streamlit has a growing community and ecosystem, which means plenty of resources, tutorials, and community support are available.
- **Deployment**: Streamlit apps can be easily deployed and shared. They can be hosted on various platforms, including Streamlit sharing, which offers a simple way to deploy Streamlit apps for free.

Streamlit + LangChain + LLMs



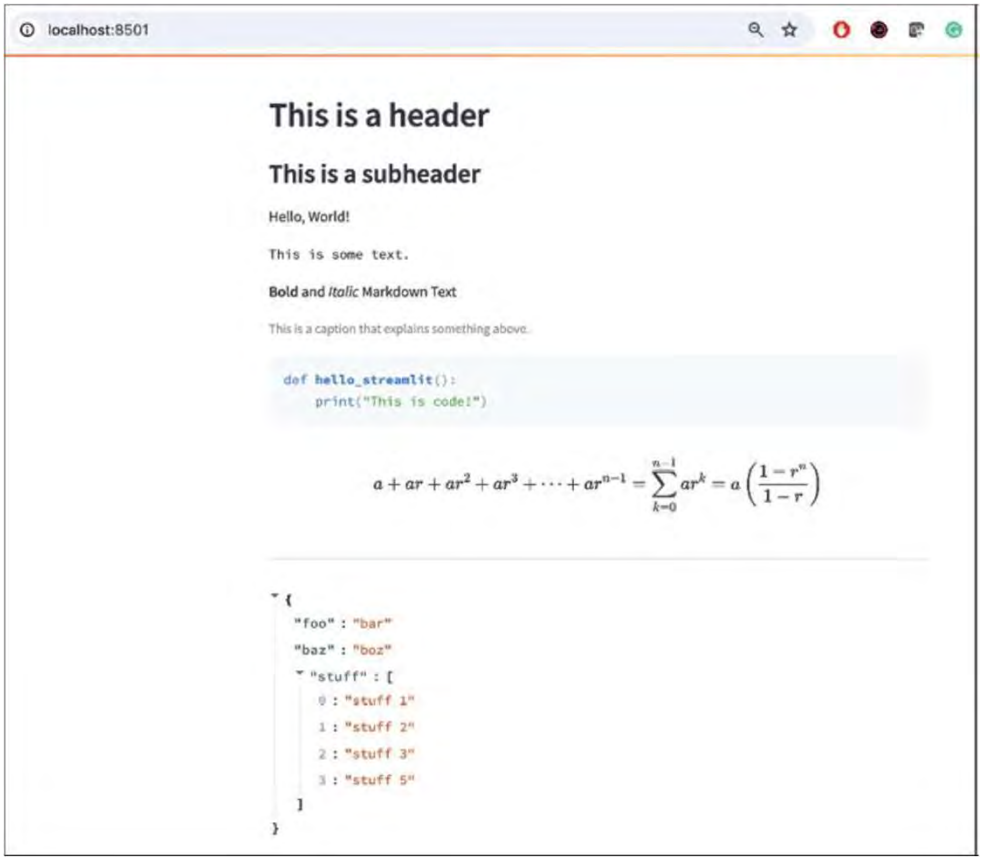
Ref: https://blog.streamlit.io/langchain-tutorial-1-build-an-llm-powered-app-in-18-lines-of-code/

Streamlit core concepts

- Streamlit is a Python framework that runs Python scripts from top to bottom
- As the script is run, Streamlit renders its output live on the browser.
- Every time a webpage is opened, or any interaction on the webpage triggers the Streamlit page to be rerun. An interaction could be a click or widget state change. This is important to note when developing apps that have to maintain the state.
- Streamlit caches the result to avoid reloading and recomputing expensive operations.

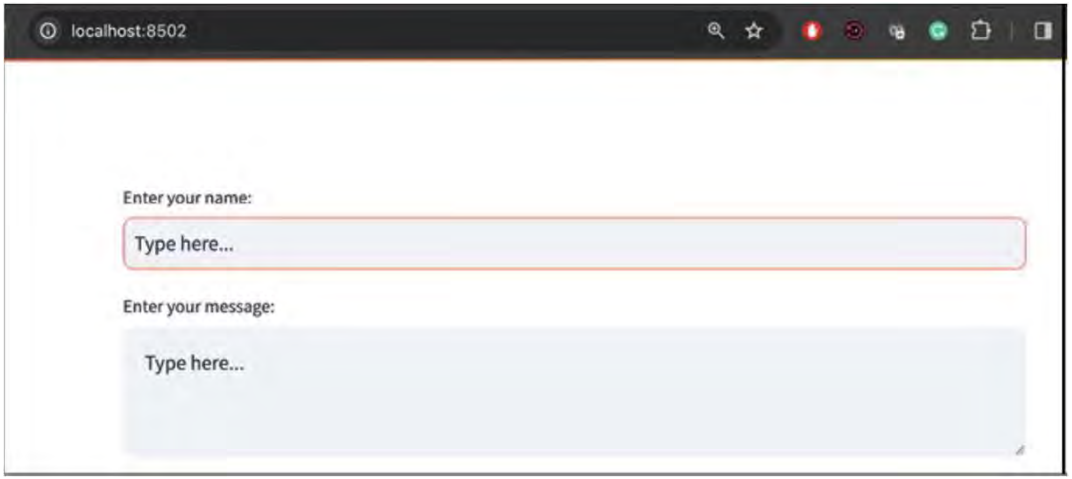
Widgets

- Display Widgets
  - st.**title**: Adds title to your page
  - st.**header**: Adds header in your page
  - st.**subheader**: Adds subheader in the page
  - st.**write**, st.**text**: Adds text in your page
  - st.**markdown**: Adds markdown in the page
  - st.**caption**: Puts the text in a caption
  - st.**code**: Adds code block in the page
  - st.**latex**: Adds complex mathematical equations
  - st.**divider**: Adds a horizontal divider
  - st.**json**: Displays a JSON in the page

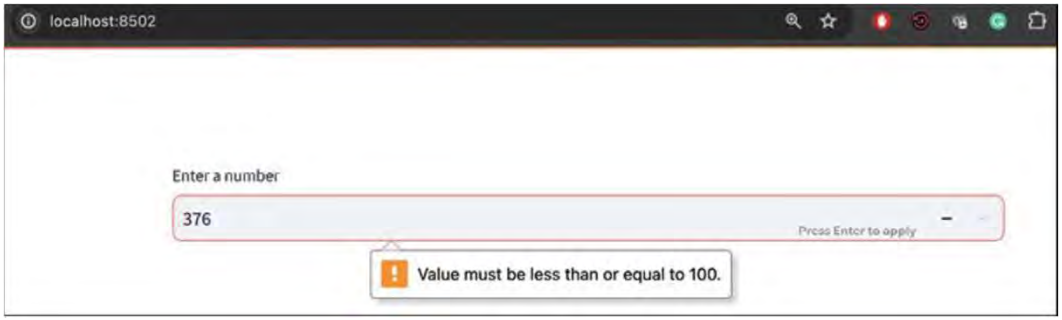


Widgets

- Text input Widgets
  - st.**text_input**: Accept text input in the page.
  - st.**text_area**: Accept large text input in the page

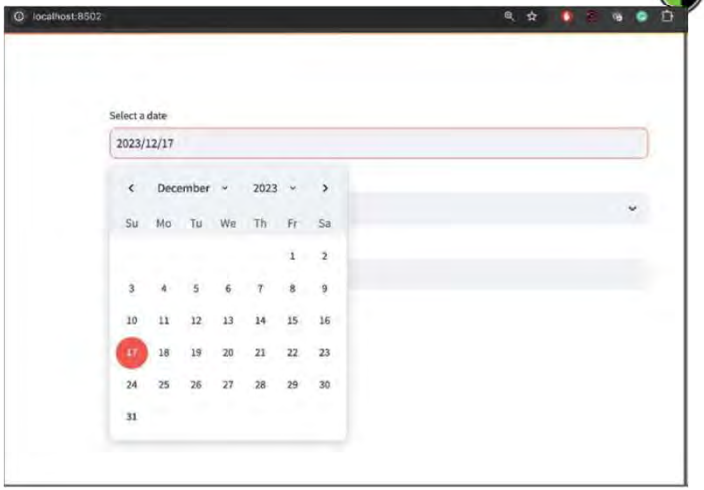


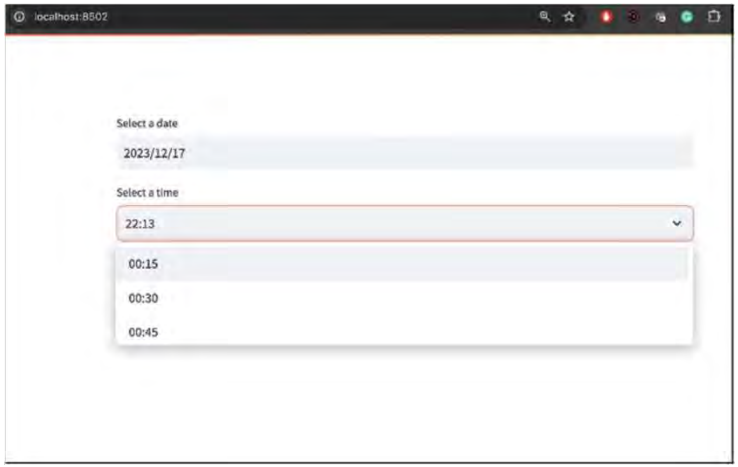
- Numberical input Widgets
  - st.**number_input**: Accept numeric inputs in the page, with capability to set checks.

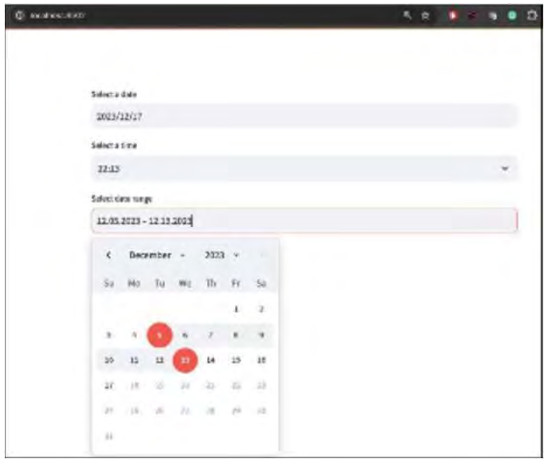


Widgets

- Date and time Widgets
  - **date_input**: Accepts date input in the page.
  - **time_input**: Accepts time input in the page.

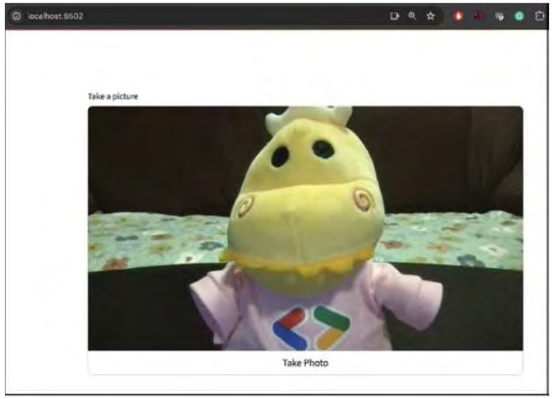






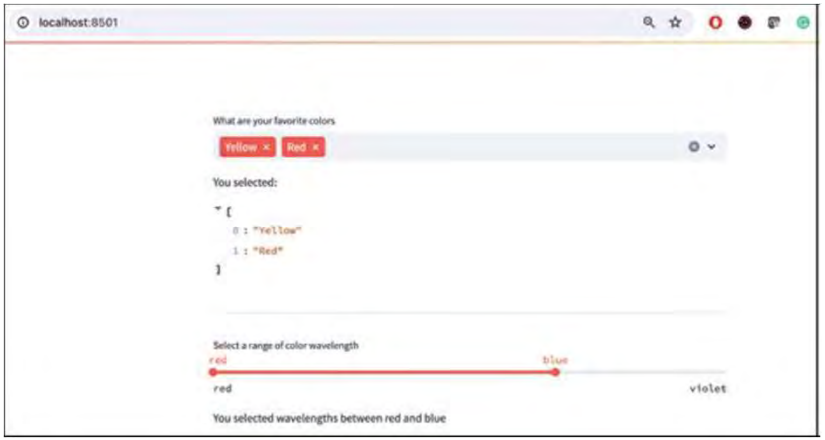
Widgets

- Other input Widgets
  - st.**camera_input**: Accept input from camera
  - st.**color_picker**: Accept color code input
  - st.**file_uploader**: Upload file in the app



Widgets

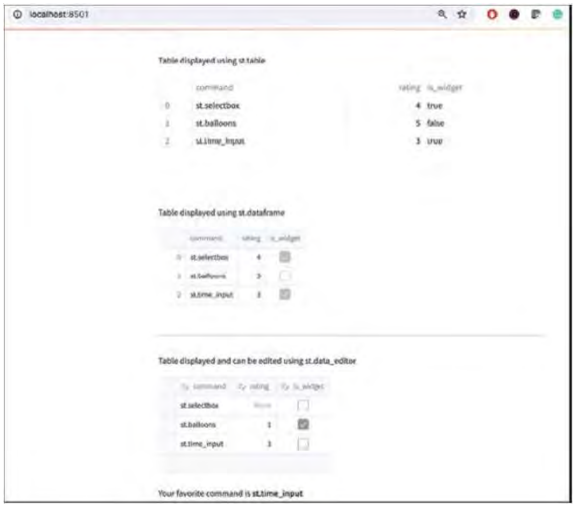
- Multiple choice Widgets
  - st.**multiselect**: Selects multiple options from the dropdown.
  - st.**select_slider**: Accepts a range input from the slider.

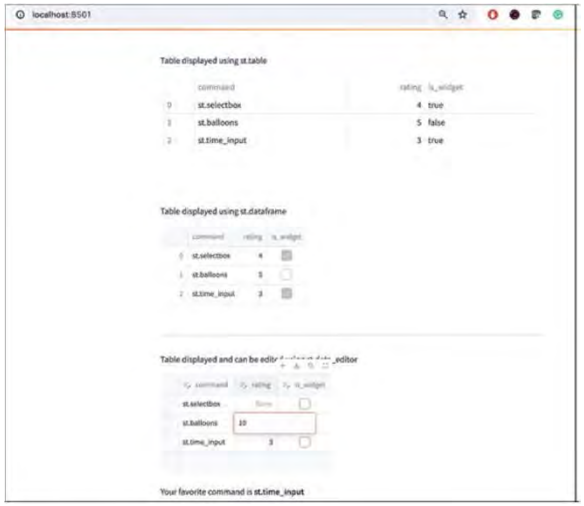


- Button Widgets
  - st.**button**: Button widget to click and perform actions like callback and
  - submission.
  - st.**download_button**: Button widget with function to download.
  - st.**link_button**: Button widget tied to a hyperlink

Widgets

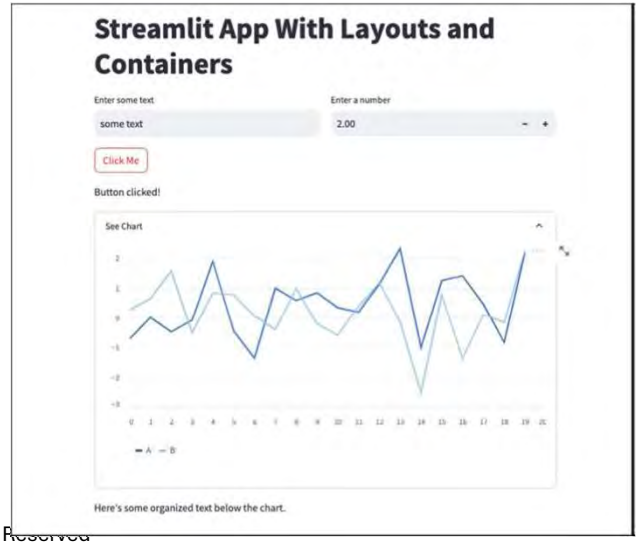
- Data Widgets
  - st.**dataframe**: Displays dataframe in the app
  - st.**table**: Displays table in the app
  - st.**data_editor**: Displays a dataframe that can be edited from the app





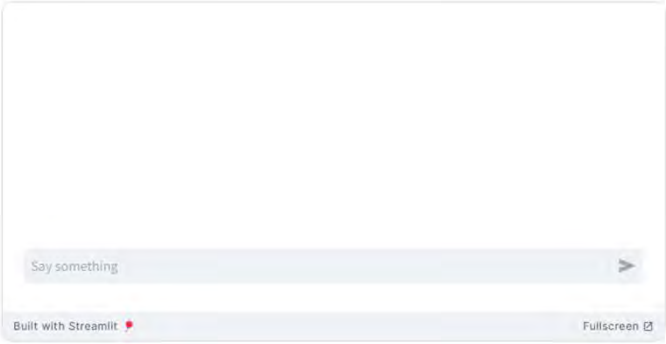
Styling and Layout

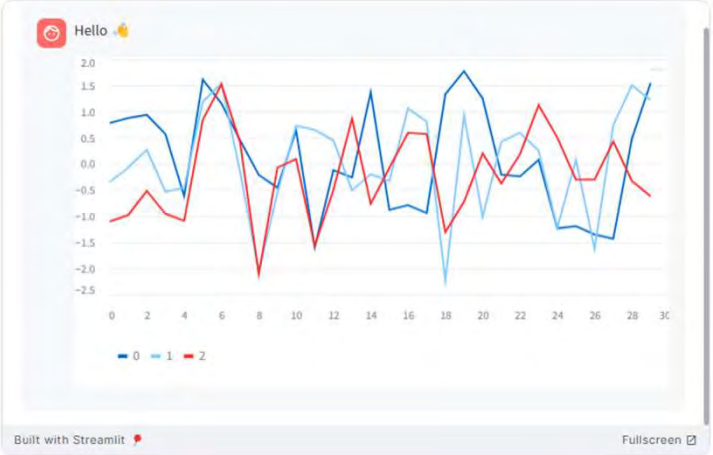
- st.**side_bar**: Help to organize elements in the sidebar of the page.
- st.**columns**: Splits the page into columns.
- st.**tabs**: Splits the content into tabs inside the page.
- st.**expander**: Adds the content in the expandable section.
- st.**empty**: Creates an empty container, useful when a placeholder is required in the layout which can be later populated by dynamic actions.



Chat elements

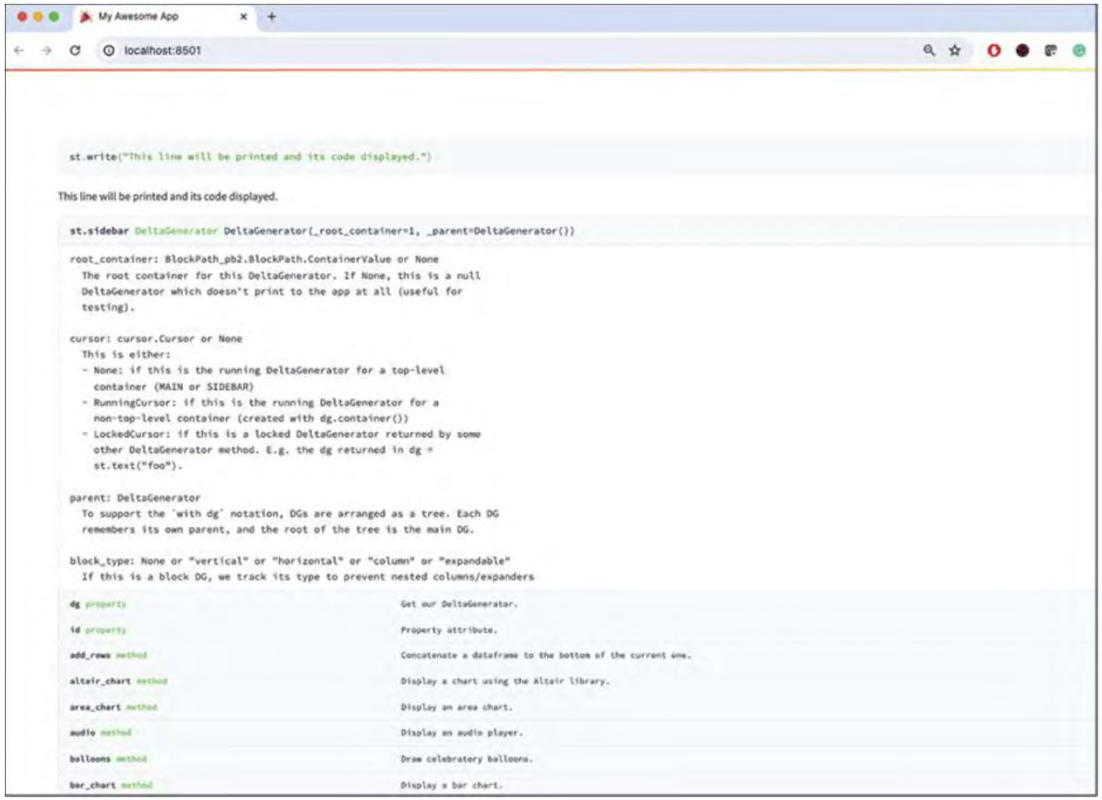
- st.**chat_input**: display a chat input widget
- st.**chat_message**: insert a chat message container
- st.**status**: display output of long-running tasks in a container
- st.**write_stream**: Write generators or streams to the app with a typewriter effect.





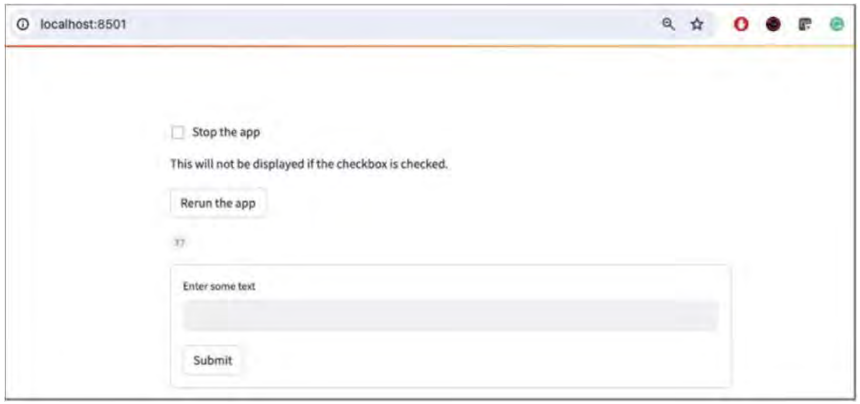
Utility Widgets

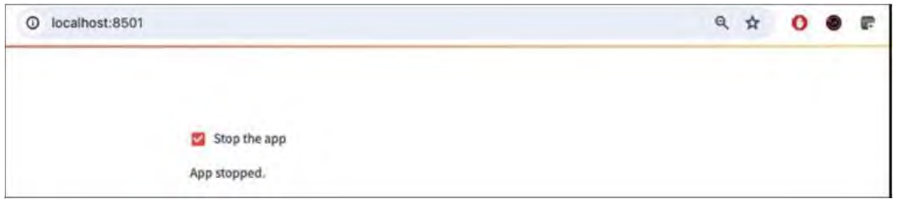
- st.**set_page_config**: Sets the page level settings, like page name, to be displayed on the browser tab.
- st.**echo**: Displays text as a code block.
- st.**help**: Prints help results for any object.



Control flow Widgets

- st.**stop**: This is used to stop the execution of the Streamlit script.
- st.**form** and st.**form_submit_button**: These widgets allow grouping a set of input widgets into a form. This is helpful when complex input needs to be submitted collectively.
- st.**rerun**: This is used to rerun a Streamlit app programmatically.





Streatlit core chart elements

- st.**line_chart** - Line charts: Ideal for showcasing trends and progressions over time.
- st.**bar_chart** - Bar charts: Effective for comparing quantities across different categories.
- st.**area_chart** - Area charts: Useful for illustrating cumulative totals and understanding part-to-whole relationships.
- st.**scatter_chart** - Scatter charts: These are useful to show the relationship between two variables. They are powerful for visualizing patterns, concentrations, and outliers in data
- st.**map_chart** - Map charts: Maps are used to visualize geospatial data. They help in visualizing locations, geographic distributions, or patterns related to physical locations.

State Management

- Streamlit provide **cookies-similar** functionality through **session state** management.
- Streamlit's session state is a special memory space where your app can **remember things**. For example, if a user types something or clicks a button, Session State can be used to store these values.
- It can store any kind of Python object, like integers, floating-point numbers, complex numbers and Booleans, data frames, and even lambdas returned by functions.
- In some cases, especially when you are running your app in different environments, like on different computers or servers, you need to be sure that everything you put in this memory can be safely stored and recovered

Activity – Build a basic LLM chat app

01 Introduction
Chat_message and chat_input ….

02 Build a bot
Build a bot that mirrors your input to get a feel for the chat element and how they work

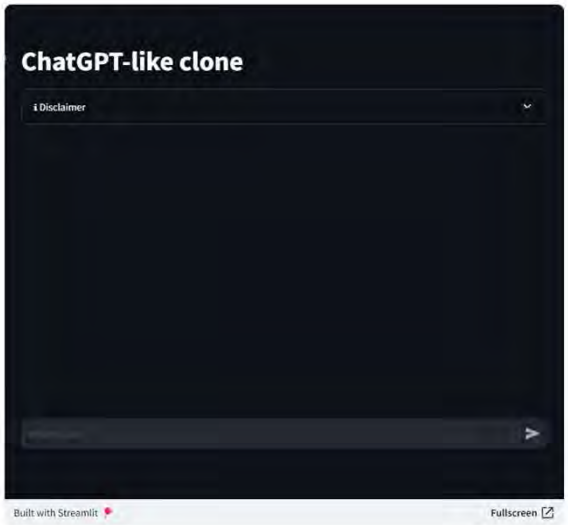
03 Session State
Introduce session state and how it can be used to store the chat history.

04 Streaming chatbot GUI
Build a simple chatbot GUI with streaming..

05 ChatGPT like app
Build a ChatGPT-like app that leverages session state to remember conversational context,

30 mins

L14-00-streamlit-chat.ipynb
L14-01-streamlit-input.ipynb
L14-02-streamlit-chat-history.ipynb
L14-03-streamlit-simplat-chat.ipynb
L14-04-streamlit-openai-clone.ipynb



Streamlit offers several commands to help you build conversational apps. These chat elements are designed to be used in conjunction with each other, but you can also use them separately.

- **st.chat_message** lets you insert a chat message container into the app so you can display messages from the user or the app. Chat containers can contain other Streamlit elements, including charts, tables, text, and more.
- **st.chat_input** lets you display a chat input widget so the user can type in a message.
- Use session state to store the chat history so we can display it in the chat message container.

Ref: https://docs.streamlit.io/library/api-reference

Target to finish by xx:xx

Activity – Build a LLM app using LangChain

REPUBLIC POLYTECHNIC

01 Introduction
Build a Streamlit LLM app that can generate text from a user-provided prompt

02 OpenAI Key
Get an OpenAI Key from the end user and validate the key

03 User Input
Get a text prompt from the user

04 Authenticate
Authenticate OpenAI with the user's key

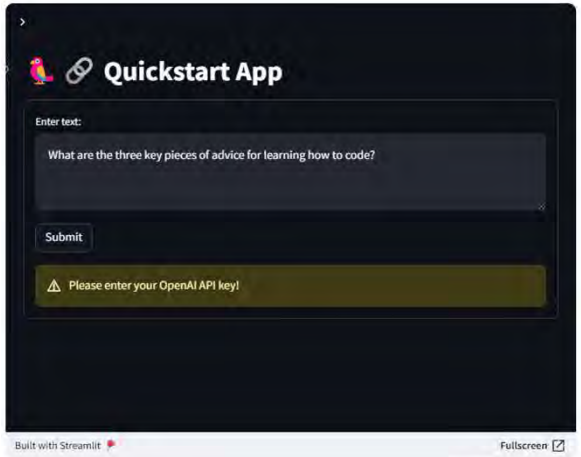
05 Send prompt
Send the user's prompt to OpenAI's API

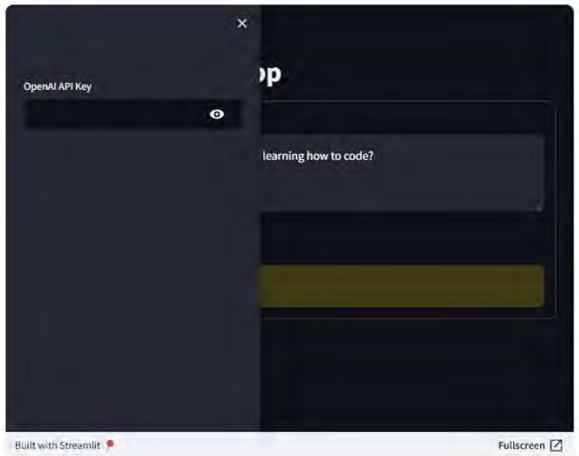
06 Display Response
Get a response and display it

Exercise [Optional]: Build a text file Q&A with OpenAI or Llama v2
Hint: use st.file_uploader()
Share a screen capture of you app on padlet.

15 mins

Scenario: …





1. **st.sidebar** display items in a side bar. Refers to [https://docs.streamlit.io/library/api-reference/layout] for more complex layouts
2. **st.form** -> Create a form that batches elements together with a "Submit" button.

* This tutorial is adapted from a blog post by Chanin Nantesanamat: LangChain tutorial #1: Build an LLM-powered app in 18 lines of code.

L14-05-streamlit-langchain.ipynb

Target to finish by xx:xx

Thank you!