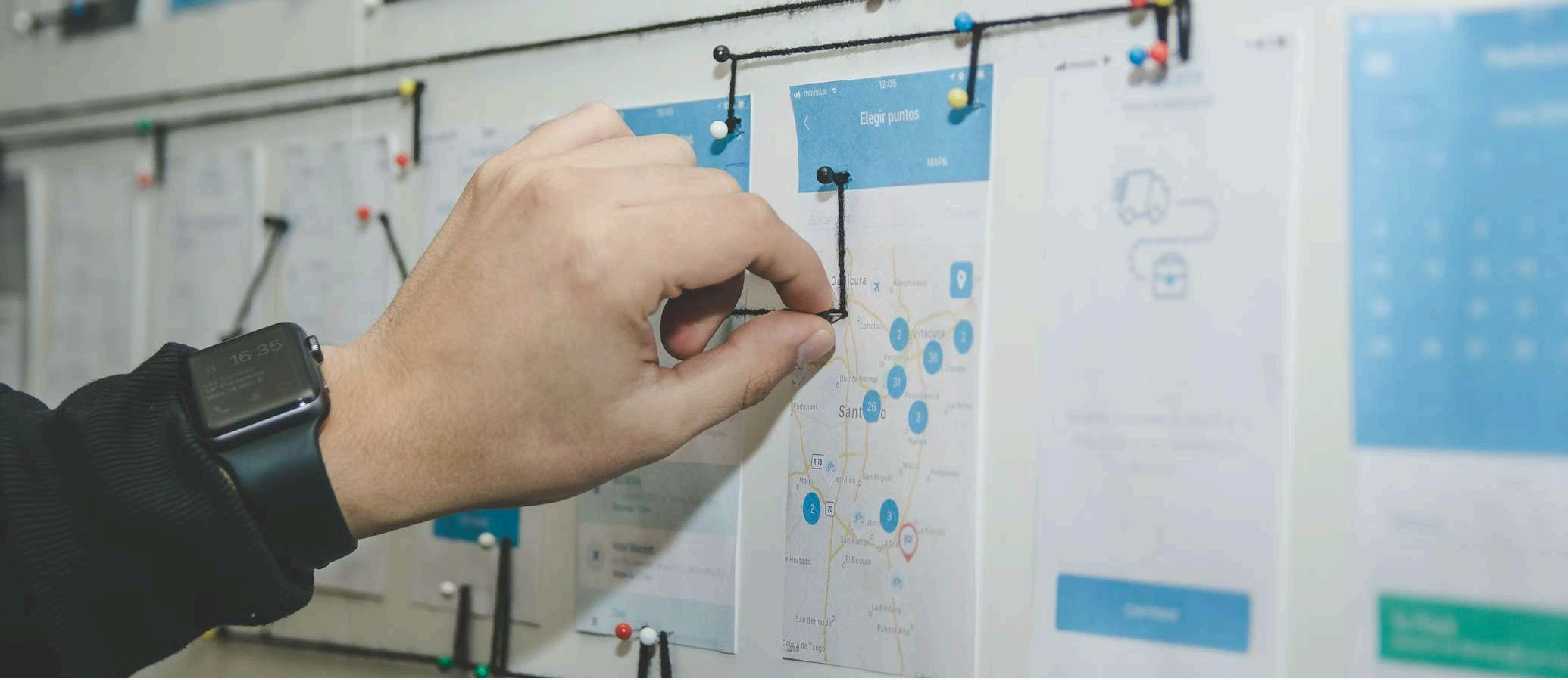
BUILD & SHARE

STREAMLIT

“TURN DATA SCRIPTS IN SHAREABLE WEB APPS”

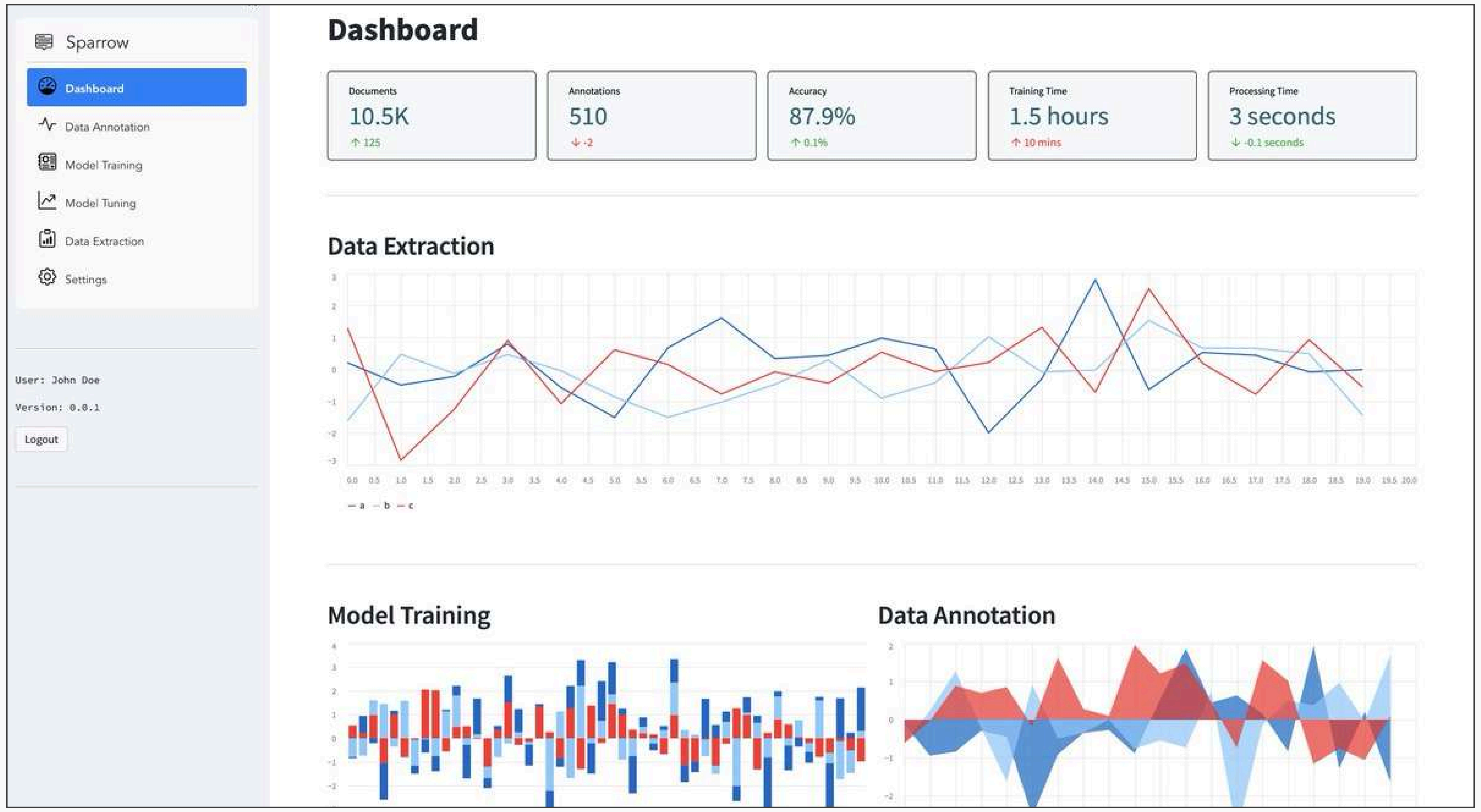
TABLE OF CONTENTS ● ● ●

Build Interactive Apps with Streamlit
Simplifying Data Apps for Everyone



INTRODUCTION

A Python library for creating interactive web apps



User: John Doe
Version: 0.0.1

Dashboard

Documents	Annotations	Accuracy	Training Time	Processing Time
10.5K	510	87.9%	1.5 hours	3 seconds
↑ 125	↓ -2	↑ 0.1%	↑ 10 mins	↓ -0.1 seconds

Data Extraction

Model Training

Data Annotation

WHAT IS STREAMLIT?

OPEN SOURCE PYTHON FRAMEWORK

Open-source library creating directly from **Python scripts**.

DATA SCIENCE DRIVEN

Create intuitive custom **dashboards** and **demos** for your **models**.

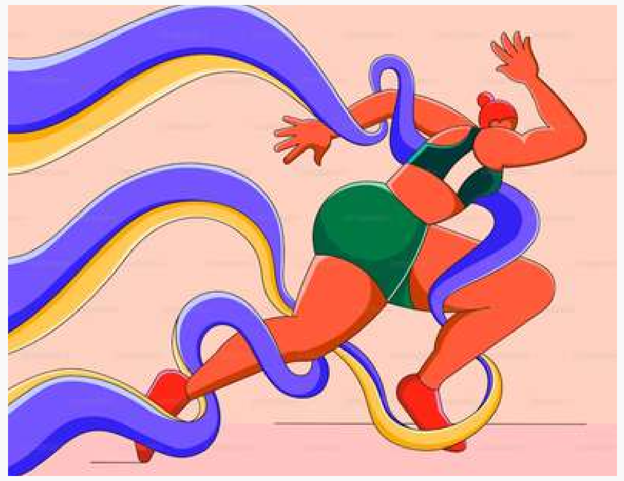
WEB APPS

No need for web development **expertise**.

PROTOTYPING

Build real-time, interactive, data-driven web apps with **minimal effort**.

WHY USE STREAMLIT?



USER-FRIENDLY

Simple and intuitive syntax, allowing you to build interactive applications in **minutes**.

ECOSYSTEM

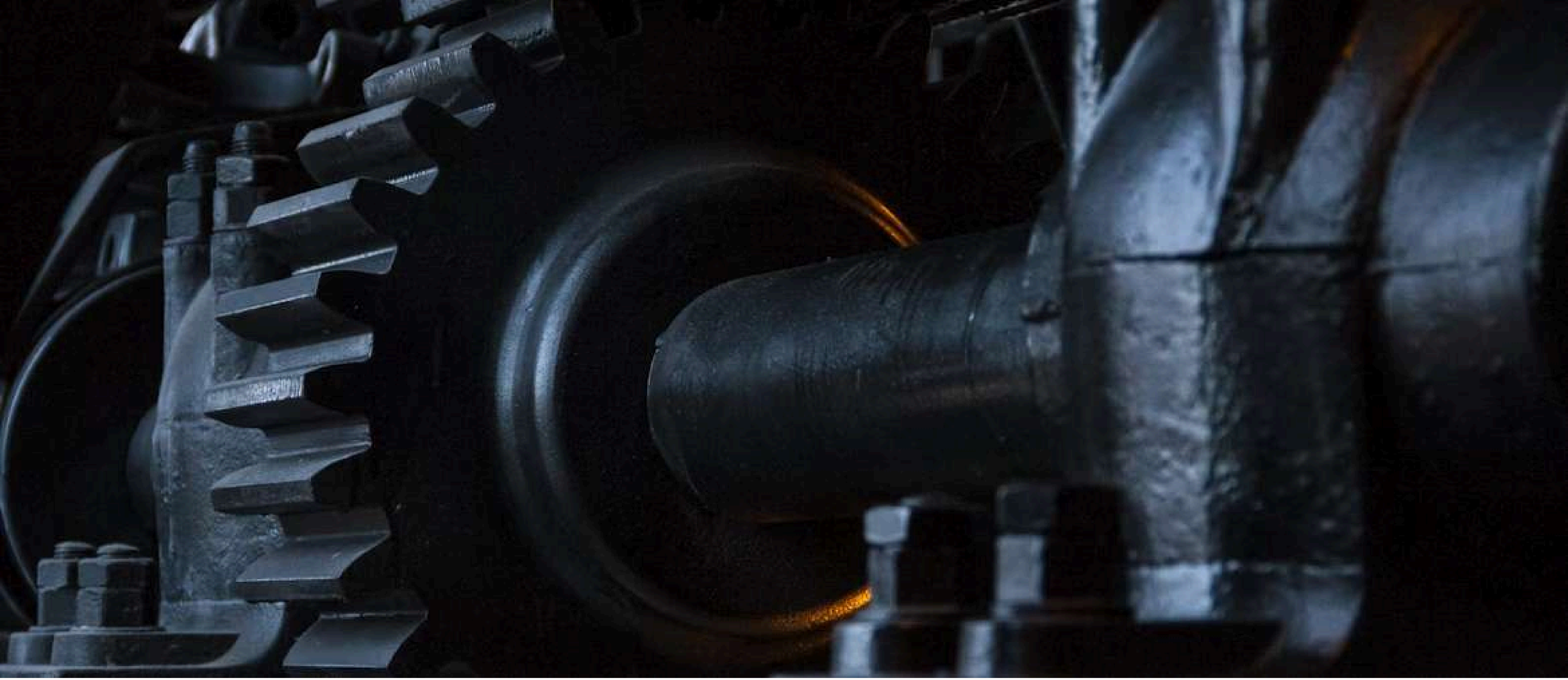
Pre-built components, a gallery of examples, and an active **community**.

REAL-TIME FEEDBACK

Quickly visualize data and test machine learning models with **interactive** inputs.

ADAPTABILITY

Suitable for **everything** from exploratory data analysis to machine learning model inference apps.

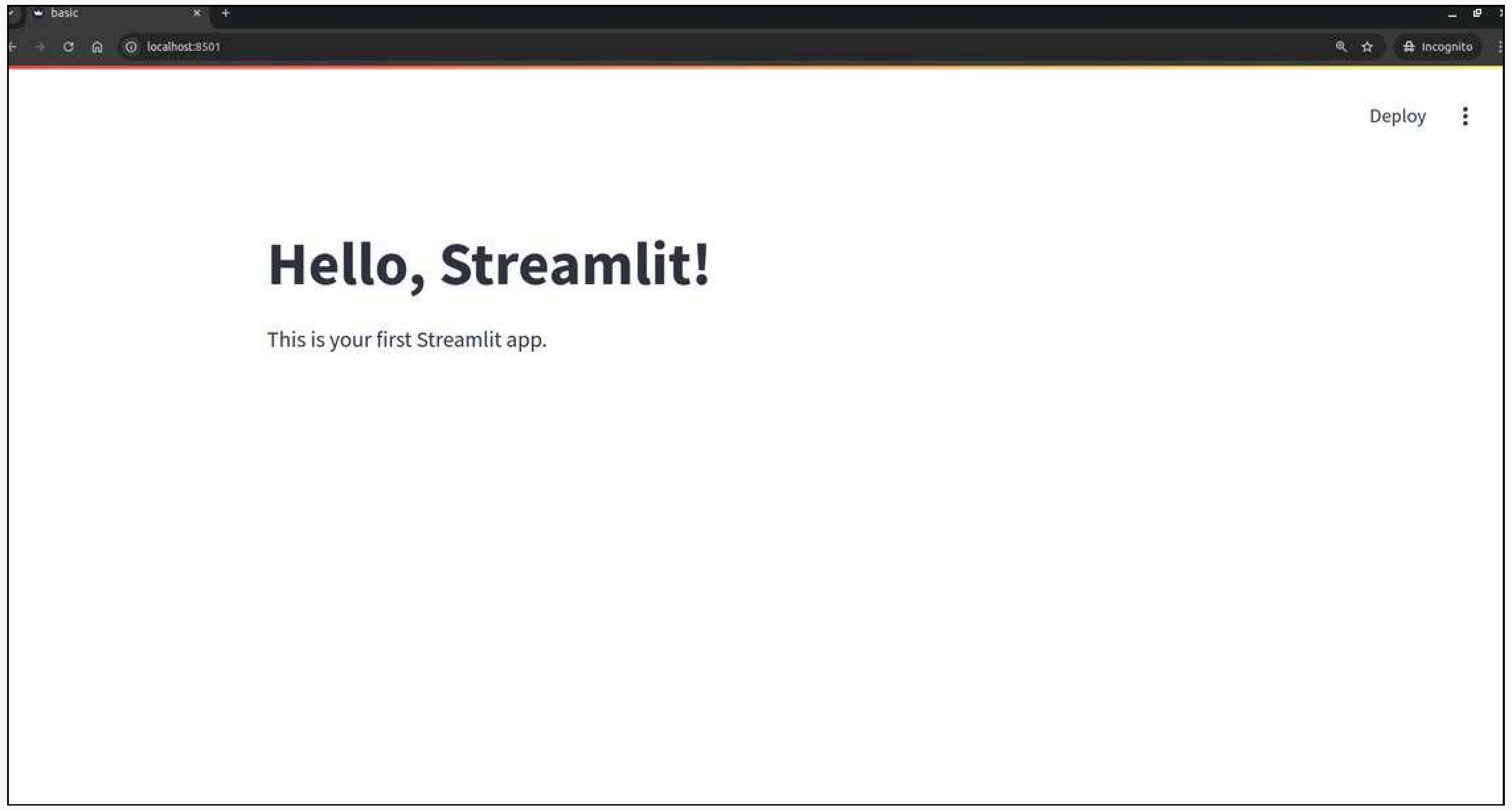


HOW IT WORKS?

Understanding Streamlit's
Reactive Framework

INSTALLATION

The first step to be able to use Streamlit is to install it. To do this we are going to use our favourite Python **package manager**.



Hello, Streamlit!

This is your first Streamlit app.

```
pip install streamlit

# Collecting streamlit
#   Downloading streamli
any.whl.metadata (8.5 kE
  Collecting altair<6,>=
  Downloading altair-5.4
any.whl.metadata (9.4 kE
  Collecting blinker<2,>
    Downloading blinker-
any.whl.metadata (1.6 kE
  Requirement already sa
achetools<6,>=4.0 in /p
```

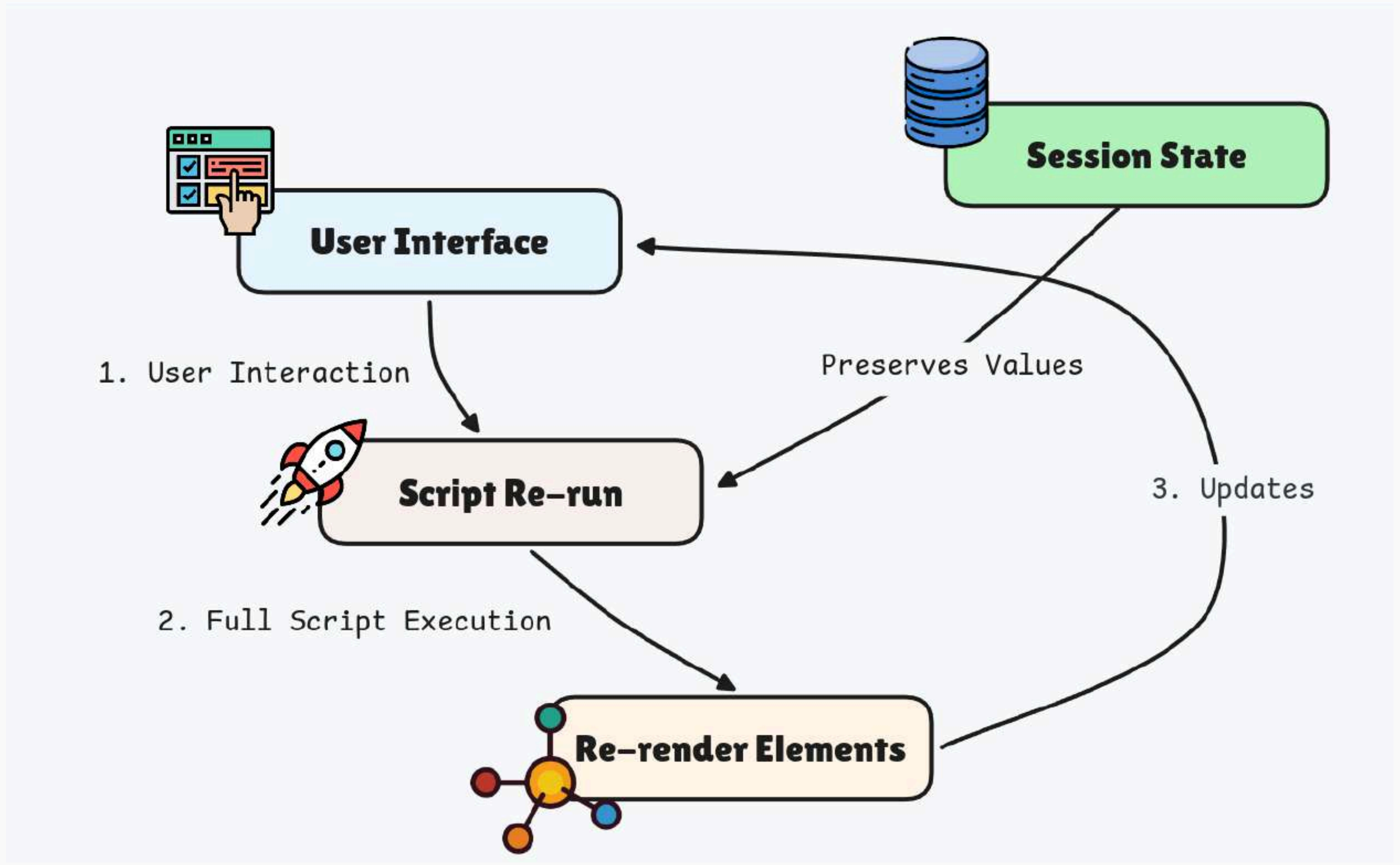
INSTALLATION

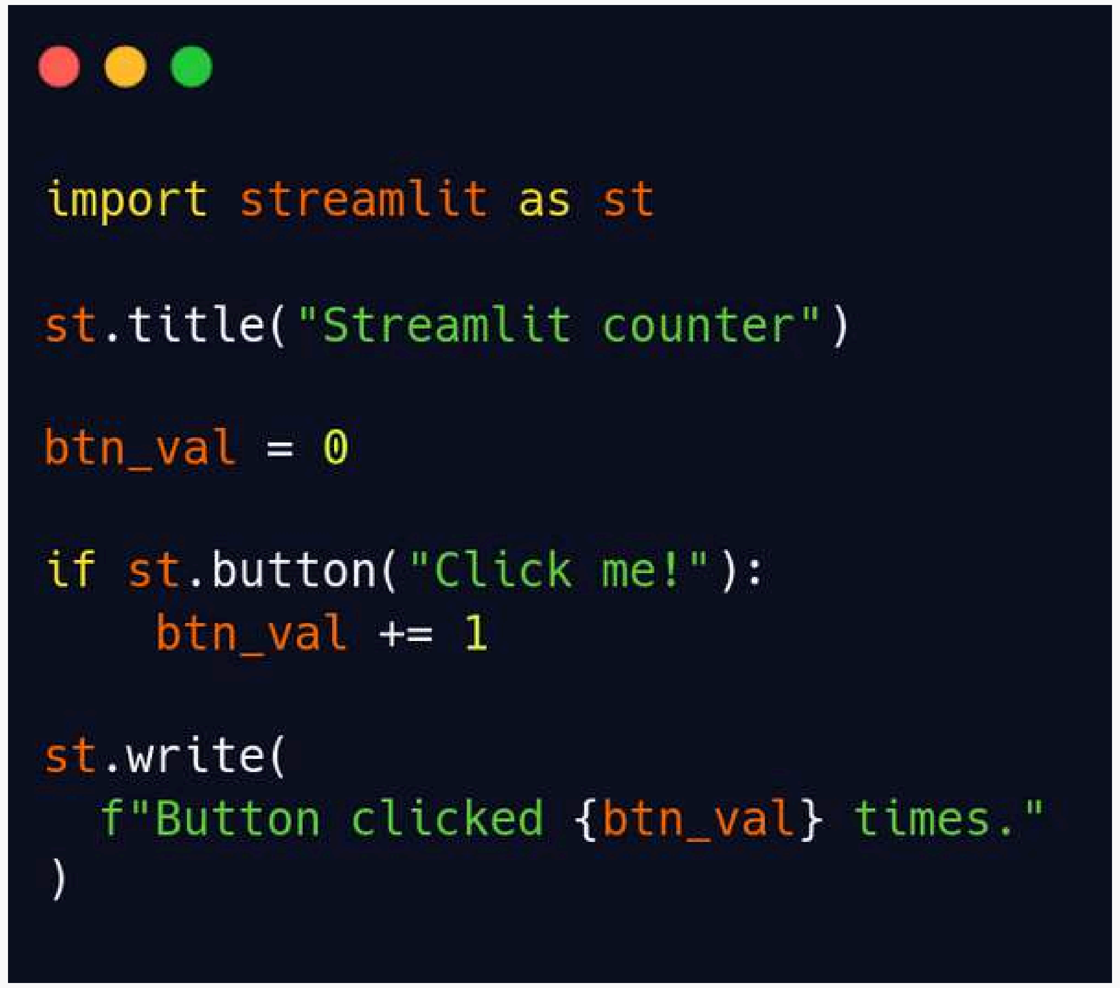
With **pip** or **conda**.

```
# app.py
import streamlit as st

st.title(
  "Hello, Streamlit!"
)
st.write(
  "This is your "
  "first Streamlit app."
)



# streamlit run app.py
```

HELLO WORLD APP

Installation test.



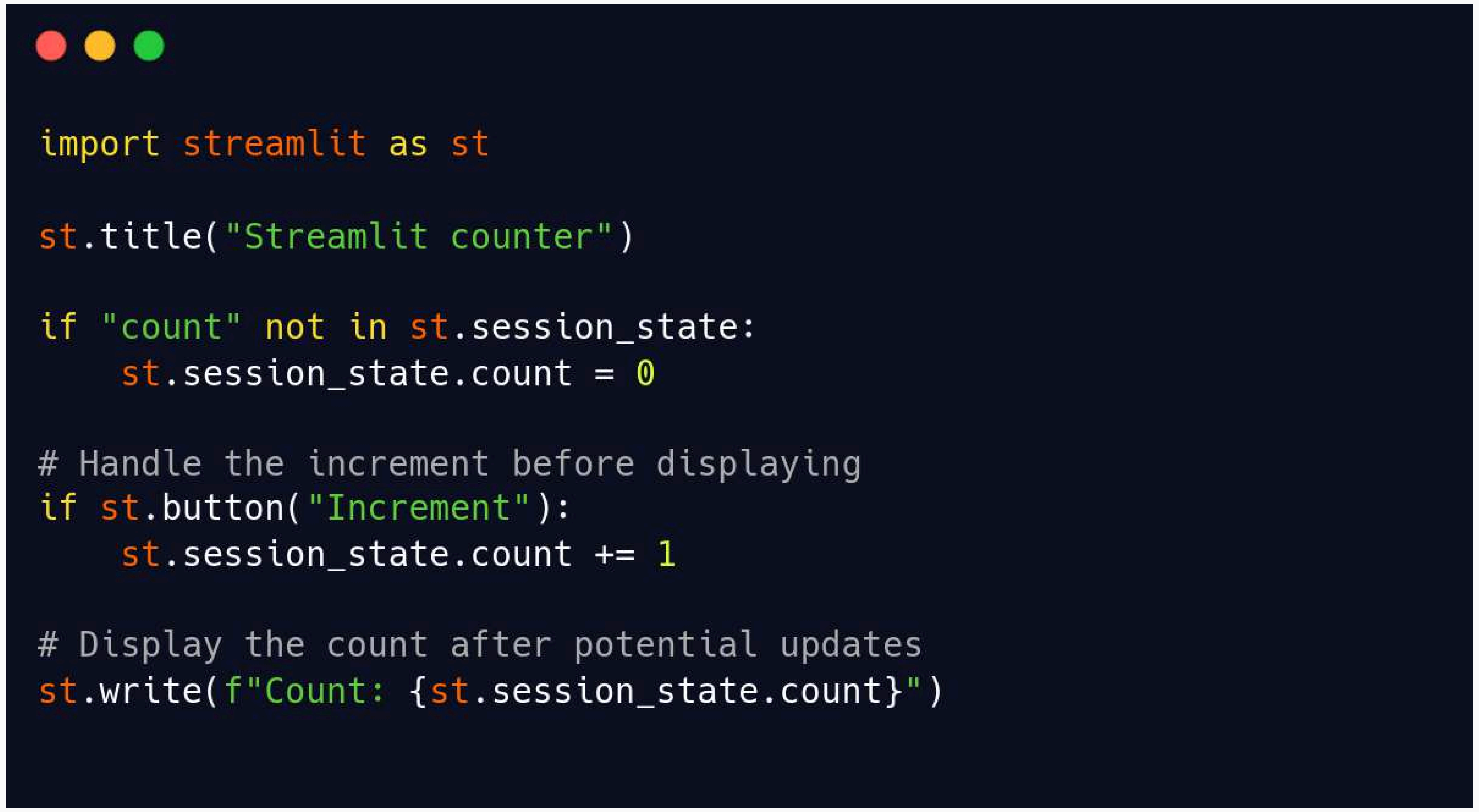


```python
import streamlit as st

st.title("Streamlit counter")

btn_val = 0

if st.button("Click me!"):
    btn_val += 1

st.write(
  f"Button clicked {btn_val} times."
)
```

The counter will never exceed 1

STREAMLIT'S REACTIVE FRAMEWORK

Streamlit is **reactive**—the app reruns from top to bottom whenever the user interacts with the interface. Each rerun starts fresh—**variables don't persist** across runs.



```python
import streamlit as st

st.title("Streamlit counter")

if "count" not in st.session_state:
    st.session_state.count = 0

# Handle the increment before displaying
if st.button("Increment"):
    st.session_state.count += 1

# Display the count after potential updates
st.write(f"Count: {st.session_state.count}")
```

STATE MANAGEMENT IN STREAMLIT

SESSION_STATE

Use *st.session_state* to **persist** data.

USE CASES

Storing counters, user inputs, or temporary results between interactions.

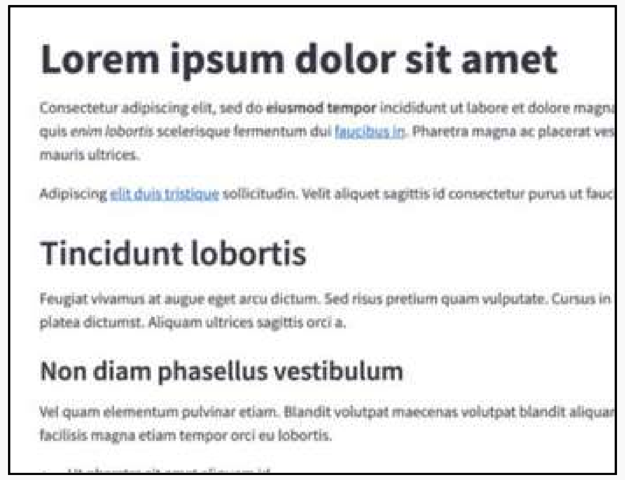
TOP-DOWN FLOW

Take care! Streamlit executes scripts **sequentially** from top to bottom, so code **order matters**.

KEY COMPONENTS

What does
it provide?

WIDGETS OVERVIEW


Lorem ipsum dolor sit amet

Consectetur adipiscing elit, sed do **eiusmod tempor** incididunt ut labore et dolore magn
quis *enim lobortis* scelerisque fermentum dui faucibus in. Pharetra magna ac placerat ves
mauris ultrices.

Adipiscing elit duis tristique sollicitudin. Velit aliquet sagittis id consectetur purus ut fauc

Tincidunt lobortis

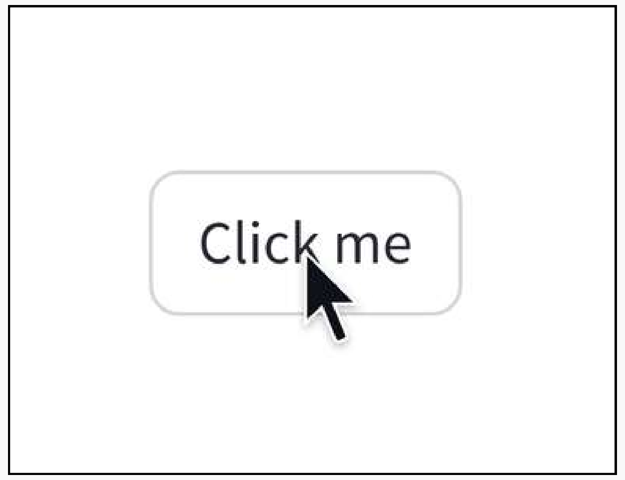
Feugiat vivamus at augue eget arcu dictum. Sed risus pretium quam vulputate. Cursus in
platea dictumst. Aliquam ultrices sagittis orci a.

Non diam phasellus vestibulum

Vel quam elementum pulvinar etiam. Blandit volutpat maecenas volutpat blandit aliqua
facilisis magna etiam tempor orci eu lobortis.

TEXT ELEMENTS

From titles, to headers, subheaders, code blocks, LaTeX and more. Even Markdown!



INPUT WIDGETS

Bake **interactivity** directly into your apps. +25 input widgets: buttons, sliders, text inputs, and more.

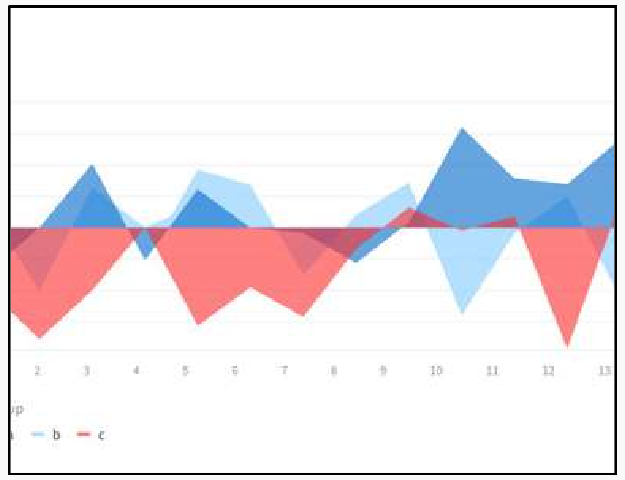


CHART ELEMENTS

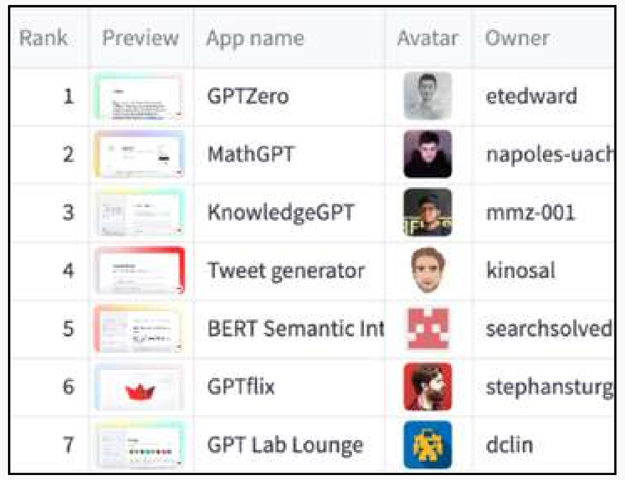
Simple area, bar, line, scatter, map, built-in charts. **Supports** different chart libraries.

MEDIA ELEMENTS

Easily embed images, videos, and audio files, or **third-party components**!

API reference here

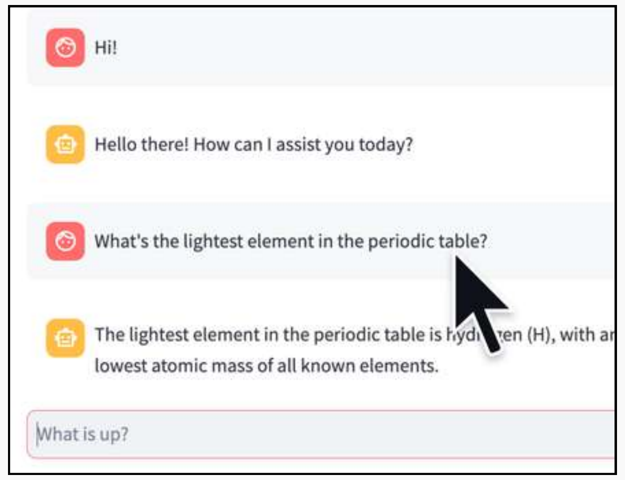
WIDGETS OVERVIEW



Rank	Preview	App name	Avatar	Owner
1		GPTZero		etedward
2		MathGPT		napoles-uach
3		KnowledgeGPT		mmz-001
4		Tweet generator		kinosal
5		BERT Semantic Int		searchsolved
6		GPTflix		stephansturg
7		GPT Lab Lounge		dclin

DATA ELEMENTS

Visualize that data quickly, interactively, and from multiple different angles.



Hi!

Hello there! How can I assist you today?

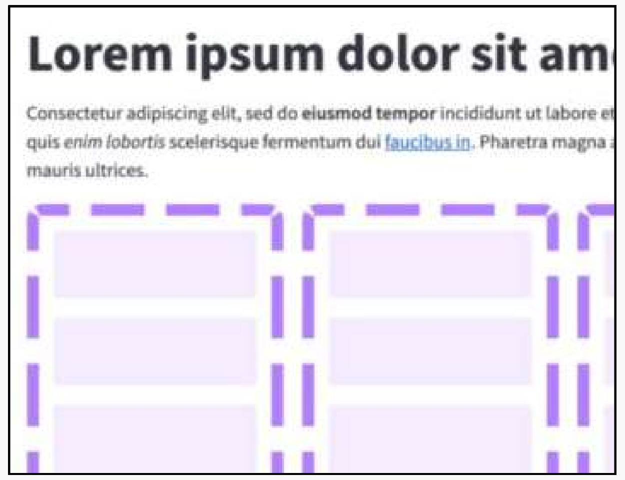
What's the lightest element in the periodic table?

The lightest element in the periodic table is hydrogen (H), with an lowest atomic mass of all known elements.

What is up?

CHAT ELEMENTS

Streamlit provides a few commands to help you build **conversational apps**.

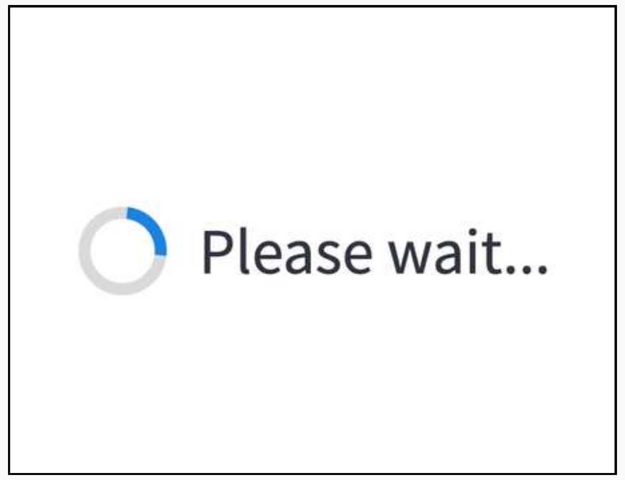


Lorem ipsum dolor sit am

Consectetur adipiscing elit, sed do **eiusmod tempor** incididunt ut labore et quis *enim lobortis* scelerisque fermentum dui faucibus in. Pharetra magna a mauris ultrices.

LAYOUTS AND CONTAINERS

Options for controlling how different elements are laid out on the screen.



Please wait...

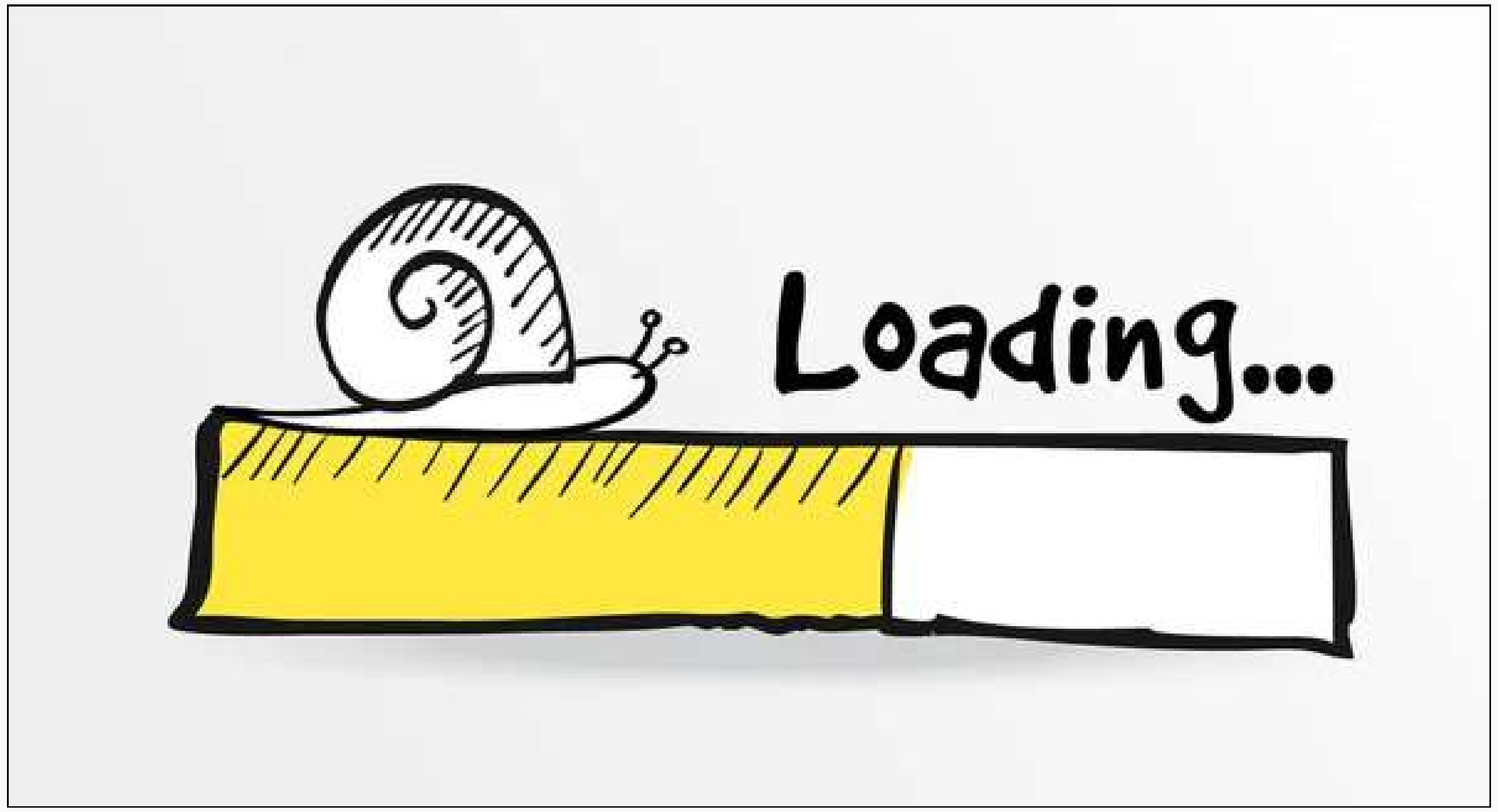
PROGRESS AND STATUS

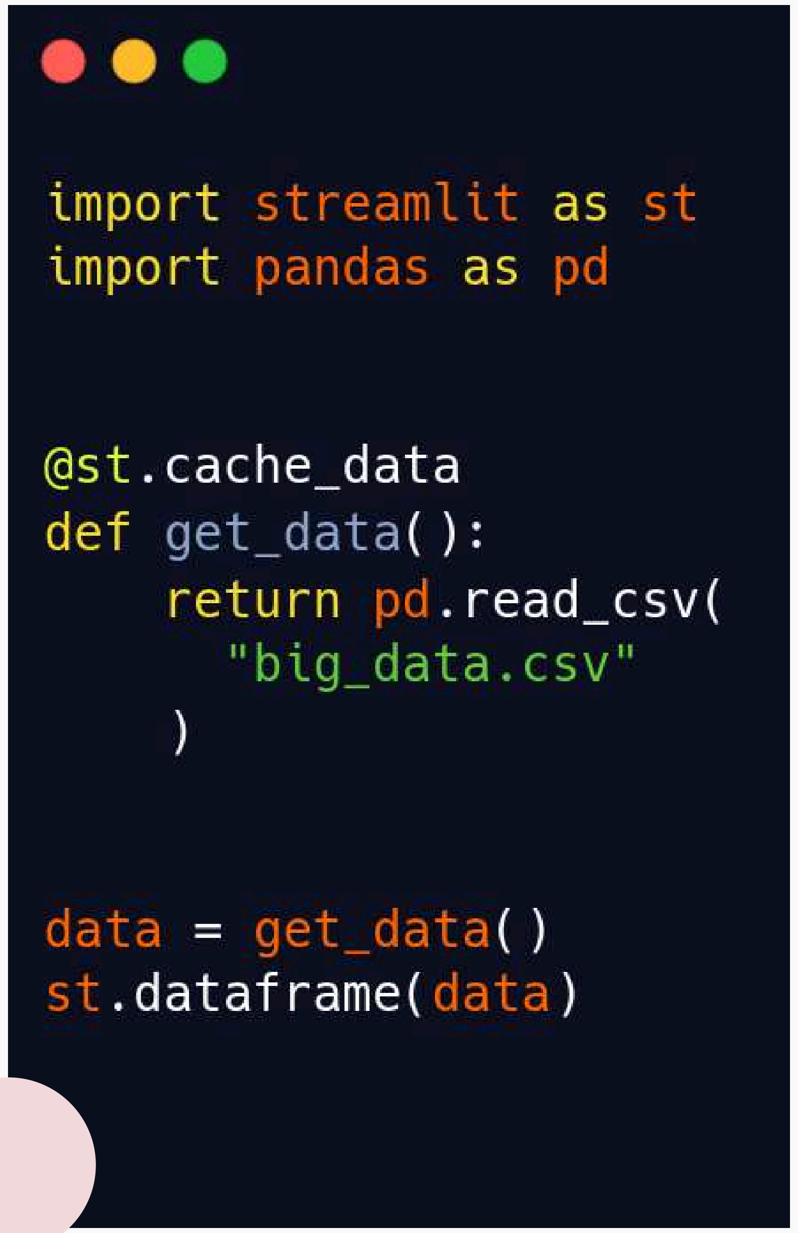
Inform your user with progress bars, status messages (like warnings), and celebratory balloons.

API reference here

CACHING

Improve application performance by saving the results of expensive calculations or **resource initialisations**. Avoid repeating the same operations.

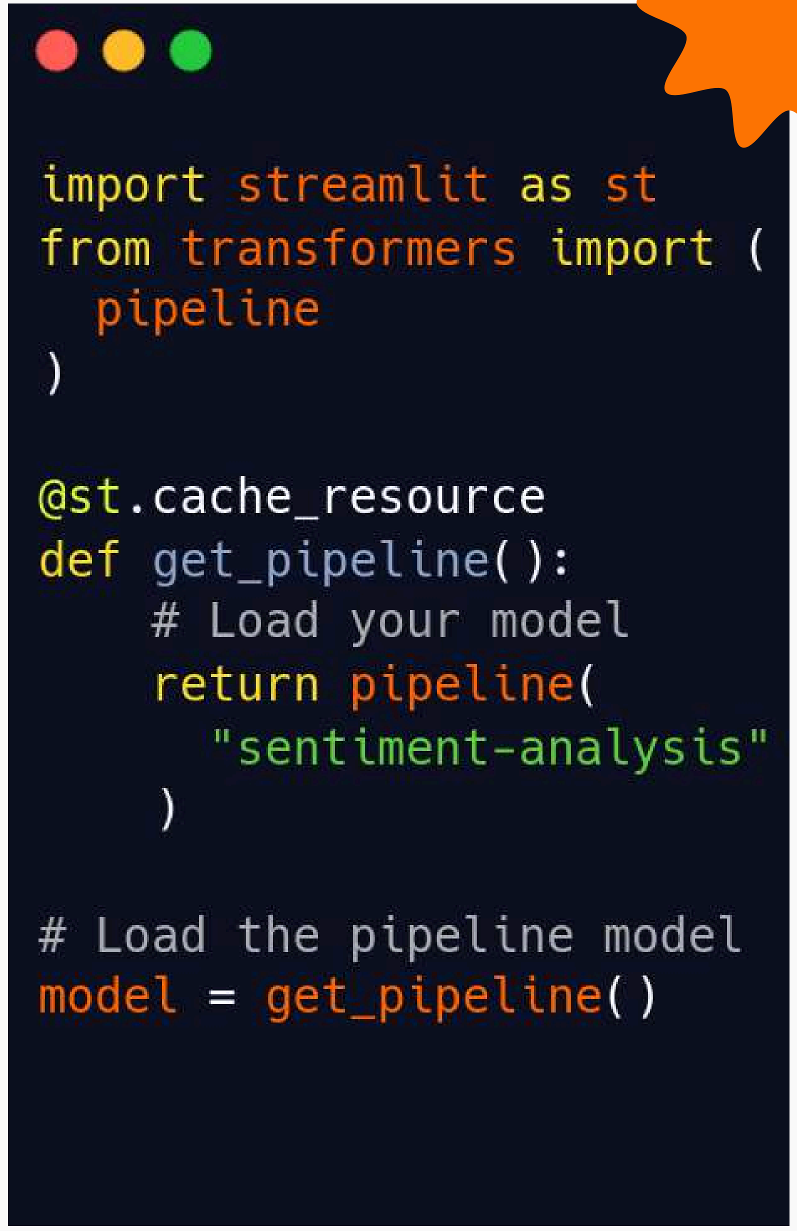




```python
import streamlit as st
import pandas as pd


@st.cache_data
def get_data():
    return pd.read_csv(
      "big_data.csv"
    )


data = get_data()
st.dataframe(data)
```

CACHE DATA

Computations or data that can be recalculated.



```python
import streamlit as st
from transformers import (
  pipeline
)

@st.cache_resource
def get_pipeline():
    # Load your model
    return pipeline(
      "sentiment-analysis"
    )

# Load the pipeline model
model = get_pipeline()
```

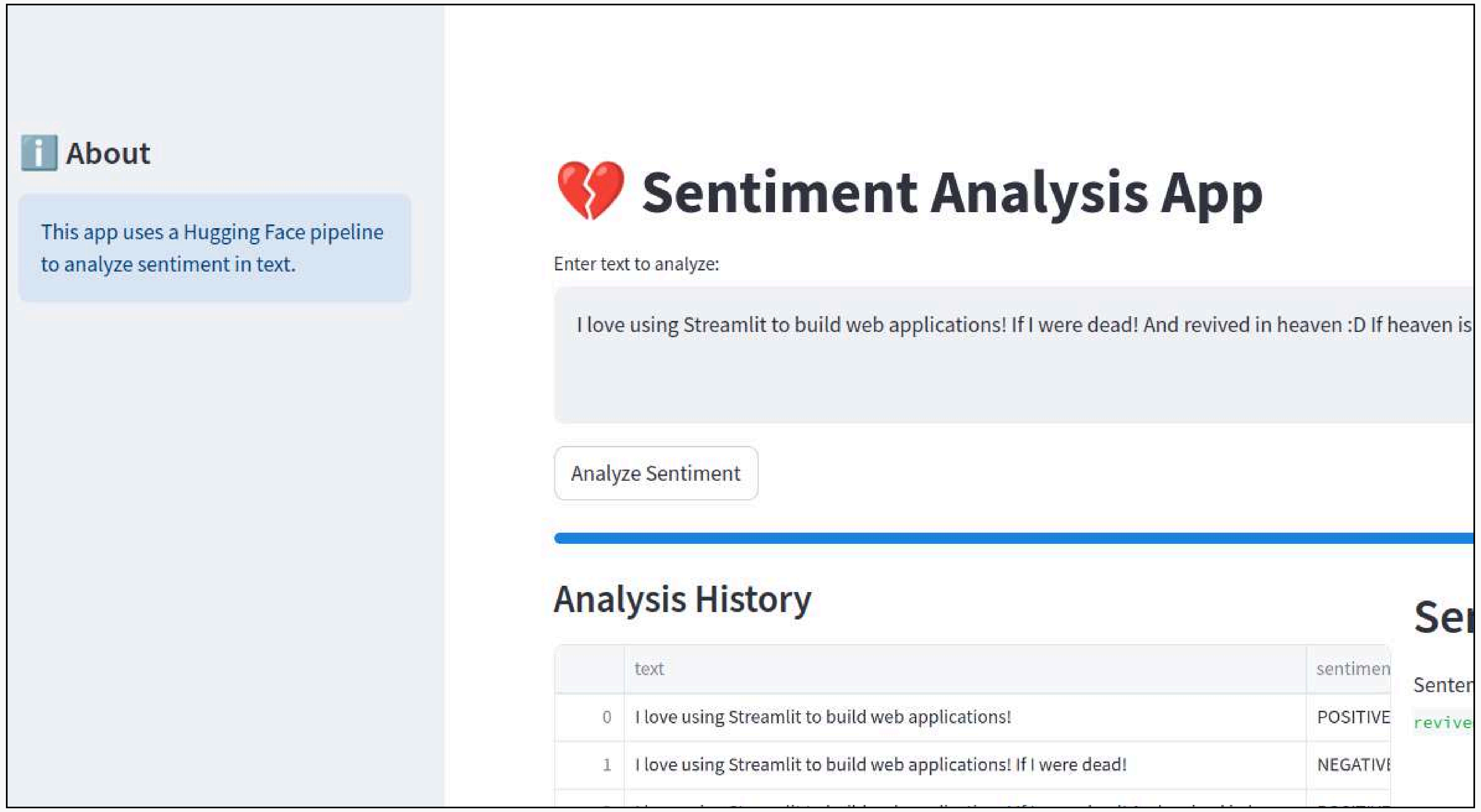
CACHE RESOURCE

objects like ML models that need to persist in memory.

ESTIMATED 50,000 CAR-SHARERS IN AUSTRALIA
CONVENIENCE
PASS BOX
PROTOTYPE IN MELB. USES ULTRASOUND REPORTS ON CYCLIST SAFETY TO AUTHORITIES
LOCATION
BAAS DOCKLESS SHAREBIKES
ALL APP-RUN KIND-UNLOCK - RIDE LETS YOU PICK UP ANY BIKE THAT IS AVAILABLE & LEAVE IT THERE
Bikeshare has never been this easy.
SMRTGRIPS
SAFETY
NAVIGATION
PRICING
$3 seems like a good deal.
HANGZHOU
60,000 BIKES FOUND EVERY 100 METRES
DENSITY
BLAZE LASERLIGHT
NOW ON 250 CITIBIKES IN NEW YORK AS A TRIAL
BLUBELL

EXAMPLE APPS

From theory to practice



ℹ️ About

This app uses a Hugging Face pipeline to analyze sentiment in text.

💔 Sentiment Analysis App

Enter text to analyze:

I love using Streamlit to build web applications! If I were dead! And revived in heaven :D If heaven is

Analyze Sentiment

Analysis History

	text	sentimen
0	I love using Streamlit to build web applications!	POSITIVE
1	I love using Streamlit to build web applications! If I were dead!	NEGATIVE

Se

Senter

revive

SENTIMENT ANALYSIS APP: MODEL AND CHARTS

LOAD A MODEL

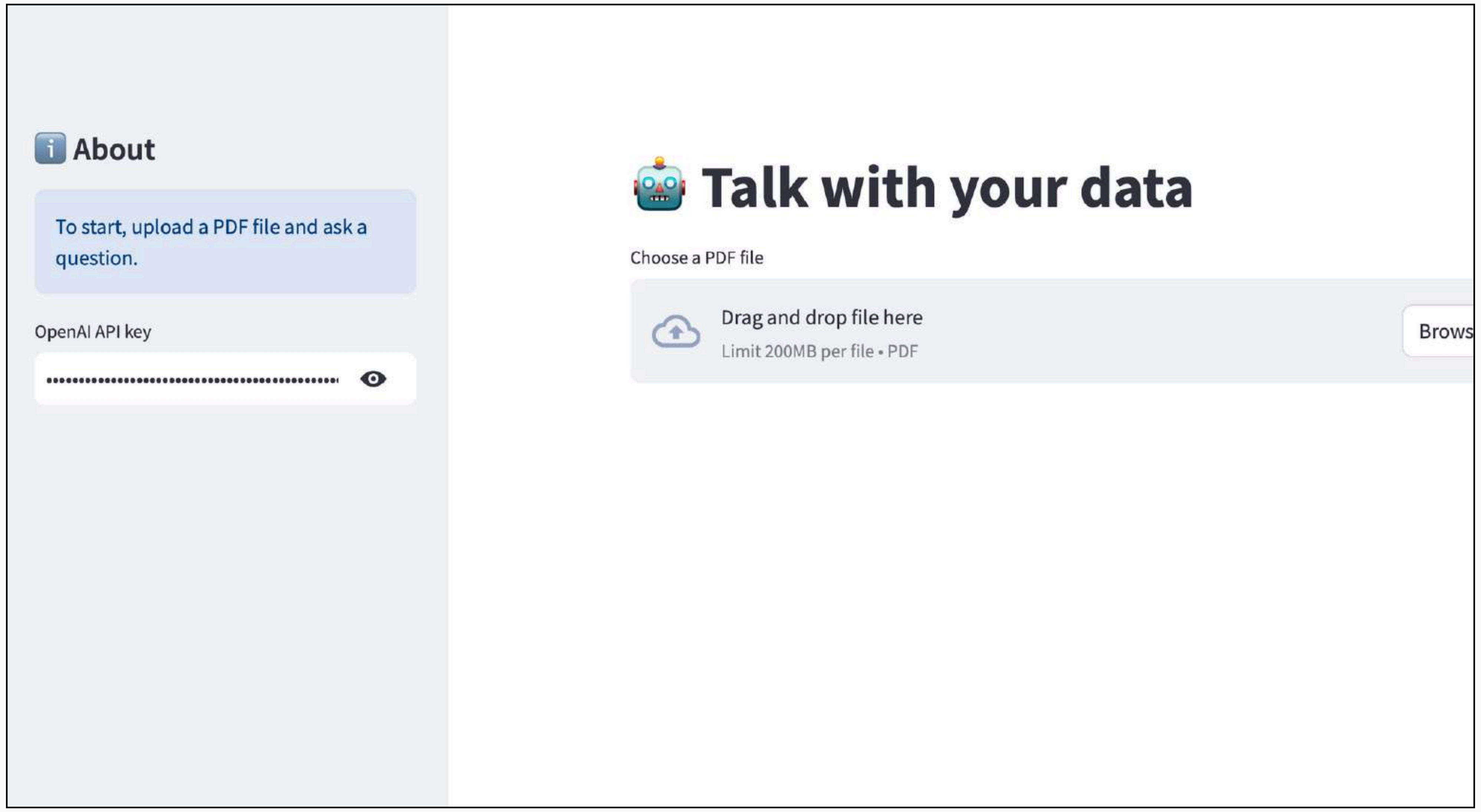
As in a **real problem** or use case, we will load a Hugging Face model and run it.

A BUNCH OF COMPONENTS

From text, sidebars, charts, dataframes, progress bars, and more.

FACING PROBLEMS

Let's look at a problematic situation we might encounter with the **state** and how to **solve it**.


About

To start, upload a PDF file and ask a question.

OpenAI API key

Talk with your data

Choose a PDF file

TALK WITH YOUR DATA APP: CHAT COMPONENT ●●●

PDF TEXT EXTRACTION

Extract and process text from **uploaded PDFs**.

INTERACTIVE CHATBOT

Interact with the content through natural language queries.

CONVERSATION HISTORY

Retain **conversation context**, providing answers that consider both previous interactions and the document's content.

CONCLUSION

What we
can keep

RECAP

Streamlit’s strengths

PROTOTYPING

SPEED

Quickly turn Python scripts into functional apps, perfect for showcasing models or data insights with minimal effort.

SHARING

SIMPLICITY

Easily share Streamlit apps via simple URLs using Streamlit Cloud or other platforms, enabling seamless collaboration.

INTERACTIVITY

ENGAGEMENT

Add widgets like sliders and dropdowns to create dynamic, interactive apps for real-time data exploration and visualization.

THANKS

mario.parreno@newfireglobal.com

maparla.es

aidventure.es

AIdventures/masterclass_streamlit