Streamlit

MIST 5730/6380 - Package Expo

Group 36: Sydney Reynolds & Chris Conrad

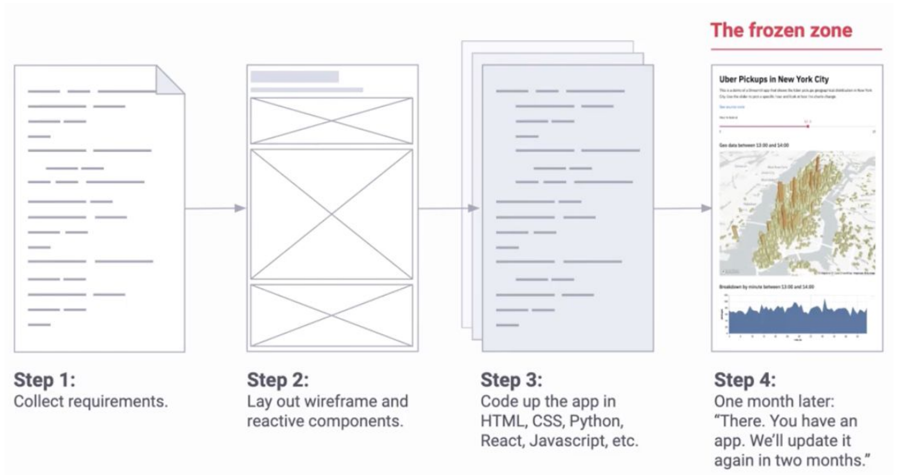
Agenda

1. Overview
2. Technical Details
3. Uses
4. Code Demo

Streamlit Background

History of Streamlit

- Streamlit was created to address a specific problem: **a lot of machine engineers' time is spent creating apps**
- While there are internal teams in many companies that are subject matter experts in this: they move on to other assignments quickly, leaving machine engineers in the "frozen zone"


The frozen zone
Uber Pickups in New York City
Step 1:
Collect requirements.
Step 2:
Lay out wireframe and reactive components.
Step 3:
Code up the app in HTML, CSS, Python, React, Javascript, etc.
Step 4:
One month later: "There. You have an app. We'll update it again in two months."

Streamlit is an app framework for machine learning engineers and data scientists specifically. We asked ourselves: what if make building apps as easy for machine learning engineers as writing python scripts?

-- Adrien Treuille, Streamlit CEO

Streamlit Package Details

- Replace the usual app building process: starting with a layout and building an event model, **with a standard python script process: iterative execution from top to bottom**
- Streamlit is free and open source, and runs locally on your computer (no cloud)

Streamlit's Basic Principles

1

Embrace Python Script

2

Treat Widgets as Variables

3

Re-Use Data and Computations

Streamlit Technical Details

Installation & Execution of Streamlit

Installing/Importing Streamlit

`pip install Streamlit` - installs Streamlit

```
import streamlit as st - imports Streamlit
```

Running Streamlit

```
streamlit run file_name.py - app opens in new browser
```

Displaying and Styling Data

- ➔ `st.write('text')` - Streamlit's "Swiss Army Knife"
  - ◆ You can pass text, data, Matplotlib figures, Altair charts, and more as a parameter
  - ◆ Displays parameter in app
- ➔ Other ways to display data:
  - ◆ `st.table()`
  - ◆ `st.dataframe()`
- ➔ Magic Commands
  - ◆ Any time that Streamlit sees a variable or a literal value on its own line, it automatically writes that to your app using st.write()
- ➔ `st.title('title')` - adds a title to the web app

Caching Data

➔

```
@st.cache

def function
```

➔ Streamlit cache allows your app to execute quickly even when loading data from the web, manipulating large datasets, or performing expensive computations.

➔ When you mark a function with the @st.cache, it tells Streamlit that whenever the function is called it needs to check:
    - ◆ The input parameters that you called the function with
    - ◆ The value of any external variable used in the function
    - ◆ The body of the function
    - ◆ The body of any function used inside the cached function

➔ If this is the first time Streamlit has seen these four components with these exact values and in this exact combination and order, it runs the function and stores the result in a local cache

➔ Next time the cached function is called, if none of these components changed, Streamlit will skip executing the function altogether and, instead, return the output previously stored in the cache.

Widgets and Graphs

- ➔ Easy to implement widgets by treating them as variables
  - ◆ `st.slider()`
  - ◆ `st.button()`
  - ◆ `st.selectbox()`
  - ◆ `st.file_uploader()`
- ➔ Streamlit supports may graph functions:
  - ◆ `st.line_chart(chart_data)`
  - ◆ `st.map(map_data)`

Layout

- ➔ In order to place your widgets and graphs in any area of your app, Strealit offers several functions to customize your layout:
  - ◆ `st.sidebar`
  - ◆ `st.right_column`
  - ◆ `st.beta_columns` - places widgets side by side
  - ◆ Stack these functions with widget functions

Streamlit Uses

Example Use Case Scenarios

- ➔ Displaying the business value of technical consulting work to clients in an organized,interactive, and clean way
- ➔ Machine Learning Engineers can build dashboards to evaluate their progress
- ➔ Building prototypes
- ➔ Demoing code

Streamlit Code Demo