Building Interactive Apps Quickly with Streamlit

Sander Van Aken (FlixBus)

Making an Impact at EURO 2024 (MC-46)

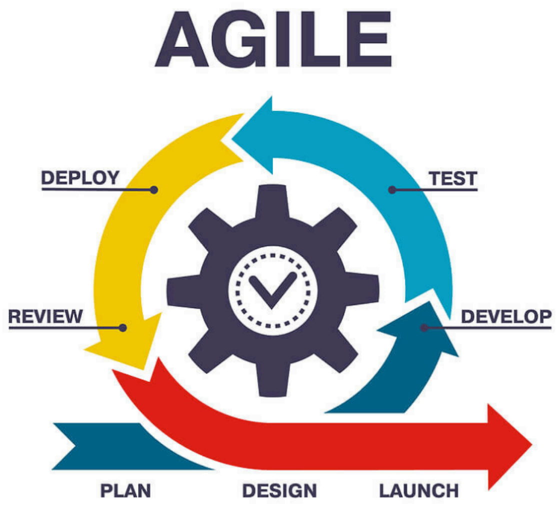
July 1st, 2024

Why consider using Streamlit for your next project?

Sharing analyses, results and models with end-users is crucial

Use-cases for interactive web-apps:

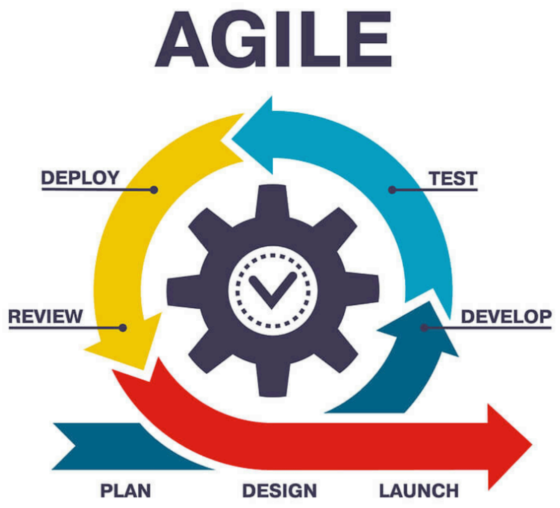
1. Discovery - when users see something, what else do they need?
2. Testing - early-stage validation of a model or algorithm
3. Deploy
    - Share results of an ad-hoc analysis with stakeholders
    - Early delivery of functionality



Sharing analyses, results and models with end-users is crucial

Use-cases for interactive web-apps:

1. Discovery - when users see something, what else do they need?
2. Testing - early-stage validation of a model or algorithm
3. Deploy
    - Share results of an ad-hoc analysis with stakeholders
    - Early delivery of functionality



Standard tools we revert to …

… might not always be the best choice for the use-case we have

What is `streamlit`?

Python-based framework to quickly develop and share web applications

https://streamlit.io/

- Simple and powerful way to iteratively develop
- Pure Python: easy and intuitive coding, no frontend experience needed
- Integrates with a standard libraries like `pandas`, `plotly`, `matplotlib` etc.
- Active developer community: lots of resources, support and extensions available

Getting started ...

Installation, Hello World, and basic concepts

From installation to your first app

1. Installation is easy and straightforward, by running the following command in your terminal:

   ```
   pip install streamlit
   ```

From installation to your first app

1. Installation is easy and straightforward, by running the following command in your terminal:

```
pip install streamlit
```

2. Create a `hello_world.py` file inside your project directory and add the following code:

```python
import streamlit as st

st.title('Hello, World!')
st.markdown('''Streamlit also supports <span style="color:red">Markdown</span>''')
```

From installation to your first app

1. Installation is easy and straightforward, by running the following command in your terminal:

```
pip install streamlit
```

2. Create a `hello_world.py` file inside your project directory and add the following code:

```python
import streamlit as st

st.title('Hello, World!')
st.markdown('''Streamlit also supports <span style="color:red">Markdown</span>''')
```

3. In your project directory, run the following command in your terminal:

```
streamlit run hello_world.py
```

4. The app will open in a new browser window.

Some basic, yet useful and important concepts

- While developing your app, you can get **immediate feedback** about the changes you make by refresh the tab, or clicking the "Rerun" button
- Capturing user input and interactions with the app is straightforward for various widgets

```
user_input = st.text_input('Please enter your name')
st.write(user_input)
```

Some basic, yet useful and important concepts

- While developing your app, you can get **immediate feedback** about the changes you make by refresh the tab, or clicking the "Rerun" button

- Capturing user input and interactions with the app is straightforward for various widgets

```
user_input = st.text_input('Please enter your name')
st.write(user_input)
```

- *Reorganizing* the app is easy and straightforward using different containers and widgets like `columns`, `expander` and `tabs`.

```
expander_container = st.expander('Click to expand')

# Option 1
with expander_container:
      st.write('Everything within the `with` block will be inside the expander ...')

# Option 2
expander_container.write('... or you can directly write to the expander')
```

Streamlit re-runs on each user-interaction

In it's essence, this is great as it allows for an *easy-to-implement interactive and dynamic user experience*.

However, it requires some care when designing the app

1. *Caching* might be needed to avoid performance issues

```
@st.cache_data
def load_data() -> pd.DataFrame:
    ...
```

2. *State management* can be crucial

    - to capture user interactions between reruns and persist information
    - to link different functionalities in a proper, e.g. multiple `multiselects` widget

```
# --> This will evaluate to None the first time
st.write(st.session_state.get("text_input_value"))

st.text_input("Enter some text", key="text_input_value")

# --> This will evaluate to an empty string right away
st.write(st.session_state.get("text_input_value"))
```

Let's take it a step further ...

... and build an conference talk browsing app

Step-by-step development of the conference programme explorer

Check out this GitHub repository and the commits on the PRs linked below for a step-by-step walkthrough

- Hello World and basic app development
  - Hello World example
  - Collecting user input through widgets like `text_input` and `number_input`
  - Show-case of `streamlit` rerunning the app on each user interaction
  - Ease of using `session_state` to share variables between reruns
  - **Structuring the app** using `tabs`, `columns` and `expanders`
- Interactive programme explorer
  - Multiselect, and text-based filters to create **basic filtering functionality**
  - Effectively controlling in the interaction of filters with each other
  - **Making dataframes user-selectable**
  - Dynamic `expander` generation based on select rows
- Integration of optimization models and custom components
  - Optimization model creation using `pulp` to **maximize utility of attending conference sessions**
  - Integration of the basic model in a separate tab
  - Functionality to allow **user-guided optimization**
  - `streamlit-calendar` integration

Integrating OR models and algorithms

Putting models / algorithms at the fingertips of our stakeholders

Why?

- Users often know more about the problem domain than we do

  Both constraints as well as preferences can be captured

- Validate that we are building the right thing, fail early
- We can deliver algorithms earlier, and iterate on them more quickly

Putting models / algorithms at the fingertips of our stakeholders

Why?

- Users often know more about the problem domain than we do

  Both constraints as well as preferences can be captured

- Validate that we are building the right thing, fail early
- We can deliver algorithms earlier, and iterate on them more quickly

How?

1. Implement the model or algorithm in Python
2. Create a Streamlit app which embeds the model and allows users to provide input
3. Run the model or algorithm on the input and potential additional settings

Golden open-source combo for OR?

Deploying and sharing the app with stakeholders

Different alternatives for deploying and sharing apps with stakeholders

Alternative 1. Streamlit Community Cloud supports integration with GitHub

- Make your repository public
- Define dependencies as `requirements.txt` or `pyproject.toml` file
- Login to Streamlit Cloud using your GitHub account
- Create a new app directly from your **public** GitHub repository

Notes:

- *all apps have 1 GB of RAM available + you can have at most one private app, others are public*
- *for `PuLP`: make sure not to print solver messages (set `msg=0`)*

Different alternatives for deploying and sharing apps with stakeholders

Alternative 1. Streamlit Community Cloud supports integration with GitHub

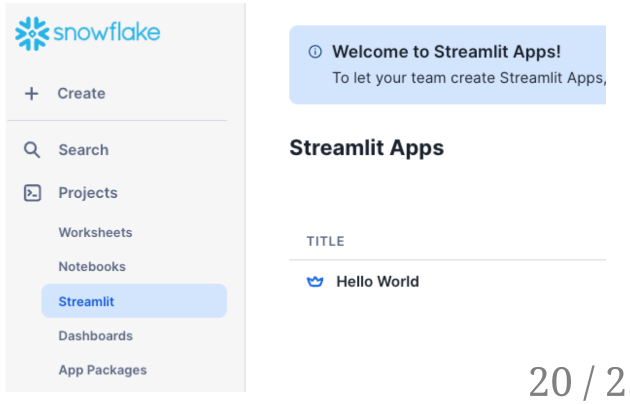
- Make your repository public
- Define dependencies as `requirements.txt` or `pyproject.toml` file
- Login to Streamlit Cloud using your GitHub account
- Create a new app directly from your **public** GitHub repository

Notes:

- *all apps have 1 GB of RAM available + you can have at most one private app, others are public*
- *for `PuLP`: make sure not to print solver messages (set `msg=0`)*

Alternative 2. Build on top of Snowflake with direct access to your data

Alternative 3. Build a Docker image and deploy it on a cloud provider like AWS, Azure or Google Cloud



Welcome to Streamlit Apps!
To let your team create Streamlit Apps,

Streamlit Apps

TITLE
Hello World

Thank you for your attention!

Slideshow created using remark

Want to try it out yourself?

Conference programme browser

tinyurl.com/EURO2024-streamlit-demo

GitHub repository

github.com/SanderVA92

Some resources

Streamlit

- Documentation
- Cheat sheet
- `streamlit-calendar`