Project Prototyping Workshop

BIO 512

Digital Epidemiology

Project Prototyping Workshop 1
Objectives

- Learn the basics of Streamlit 🚀

Project Prototyping Workshop 1
Objectives

- Learn the basics of Streamlit 🚀
  - Streamlit components.

Project Prototyping Workshop 1
Objectives

- Learn the basics of Streamlit 🚀
  - Streamlit components.
  - Organise your Streamlit project.

Project Prototyping Workshop 1
Objectives

- Learn the basics of Streamlit 🚀
  - Streamlit components.
  - Organise your Streamlit project.
  - Integrate an LLM-powered chatbot in a Streamlit project.

Project Prototyping Workshop 1
Objectives

- Learn the basics of Streamlit 🚀
  - Streamlit components.
  - Organise your Streamlit project.
  - Integrate an LLM-powered chatbot in a Streamlit project.

- Watch a demo of a project prototype 

Project Prototyping Workshop 1
Objectives

- Learn the basics of Streamlit 🚀
  - Streamlit components.
  - Organise your Streamlit project.
  - Integrate an LLM-powered chatbot in a Streamlit project.

- Watch a demo of a project prototype 

- **Your turn !** ⭐

Streamlit
Short introduction

- Free, open-source, Python library.
- Build user-friendly apps with pure Python.
- Doesn't require extensive web development knowledge.

Documentation : https://docs.streamlit.io/.

Streamlit
Fundamentals

- Write your Streamlit application in a Python file (e.g. `app.py`).

Streamlit
Fundamentals

- Write your Streamlit application in a Python file (e.g. **`app.py`**).
- Run the application using the terminal command:

```
streamlit run app.py
```

Streamlit
Fundamentals

- Write your Streamlit application in a Python file (e.g. **`app.py`**).
- Run the application using the terminal command:
  `streamlit run app.py`
- Streamlit starts a local server to open the app in a browser page.

Streamlit
Fundamentals

- Write your Streamlit application in a Python file (e.g. **`app.py`**).
- Run the application using the terminal command:
  `streamlit run app.py`
- Streamlit starts a local server to open the app in a browser page.
- Shut down the application using the terminal command:
  `[Ctrl] + C`

Streamlit
Reactive programming

- Initial step : the script is executed from **top to bottom**.

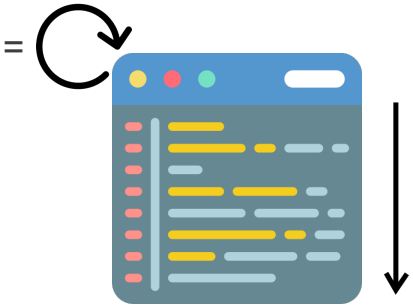
Streamlit
Reactive programming

- Initial step : the script is executed from **top to bottom**.
- Streamlit **re-runs** the entire script each time the user **interacts** with the application.

User interaction =

Streamlit
Reactive programming

🚨 **Implication 1:** Variables don't persist

User interaction = 

⚠️ **Removes variables!**

Streamlit
Reactive programming

🚨 **Implication 1:** Variables don’t persist

➡️ We will store variables in the **Session State**

Streamlit
Reactive programming

🚨 **Implication 1:** Variables don't persist

➡️ We will store variables in the **Session State**

- *Session State* : Streamlit object in which we store variables across application reruns and pages.

```python
import streamlit as st

# Dictionary-like syntax
if 'key' not in st.session_state:
    st.session_state['key'] = 'value'

# Attribute-based syntax
if 'key' not in st.session_state:
    st.session_state.key = 'value'
```

Streamlit
Reactive programming

🚨 **Implication 1:** Variables don't persist

➡️ We will store variables in the **Session State**

- *Session State* : Streamlit object in which we store variables across application reruns and pages.
- Streamlit removes the Session State whenever it is shut down.

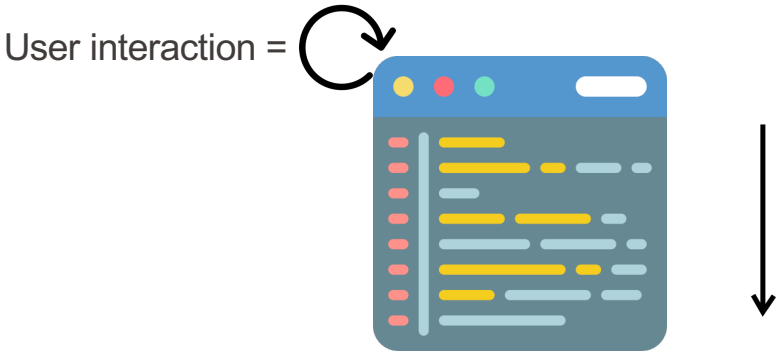
```python
import streamlit as st

# Dictionary-like syntax
if 'key' not in st.session_state:
    st.session_state['key'] = 'value'

# Attribute-based syntax
if 'key' not in st.session_state:
    st.session_state.key = 'value'
```

Streamlit
Reactive programming

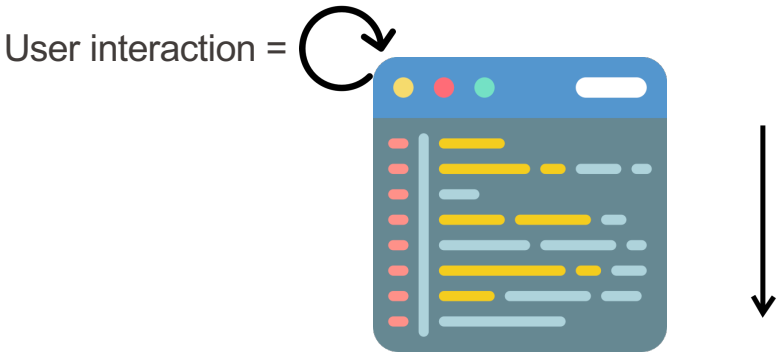
🚨 **Implication 2:** Code can be interrupted by user interactions…



Streamlit
Reactive programming

🚨 **Implication 2:** Code can be interrupted by user interactions…

➡️ but not **functions!**

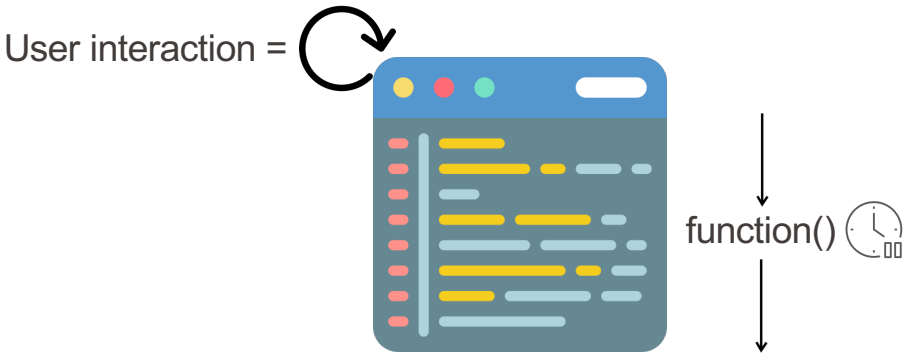


Streamlit
Reactive programming

🚨 **Implication 2:** Code can be interrupted by user interactions…

➡️ but not **functions!**

- Long-running functions will **block** the app from reacting to the user inputs.

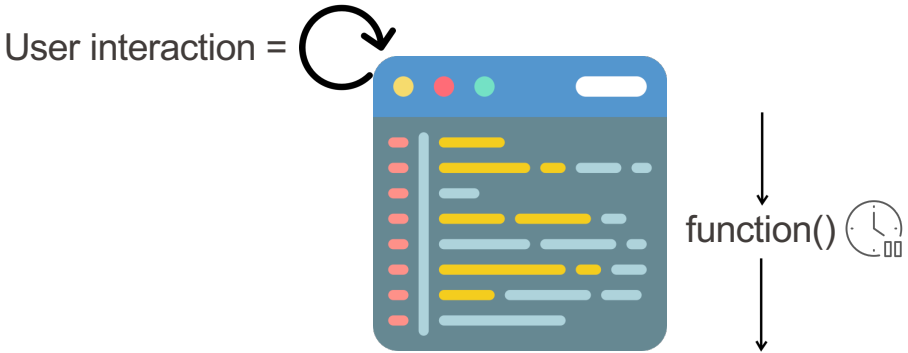


Streamlit
Reactive programming

🚨 **Implication 2:** Code can be interrupted by user interactions

➡️ but not **functions!**

- Long-running functions will **block** the app from reacting to the user inputs.
- **Design strategy**: Implementing main UI features using functions



Streamlit
Tutorial setup

Streamlit
Tutorial setup

- Clone the `streamlit` repo on the class GitLab.

1. Set up your Gitlab access, **if you have never done it before**:

Streamlit
Tutorial setup

- Clone the `streamlit` repo on the class GitLab.

1. Set up your Gitlab access, **if you have never done it before**:

   a. First, we check if you already have SSH keys.

Terminal command : `ls ~/.ssh`

- If you see : `id_rsa.pub` and other "*id_xxx*" files ➡️ Great!

- ⚠️ **If you don't see anything, generate new SSH keys:**

  Terminal command: `ssh-keygen -t rsa -b 4096 -C your_email@epfl.ch`

Streamlit
Tutorial setup

- Clone the `streamlit` repo on the class GitLab.

1. Set up your Gitlab access, **if you have never done it before**:

   a. First, we check if you already have SSH keys.

   b. Copy your public SSH key :

      Terminal command: `cat ~/.ssh/id_rsa.pub`

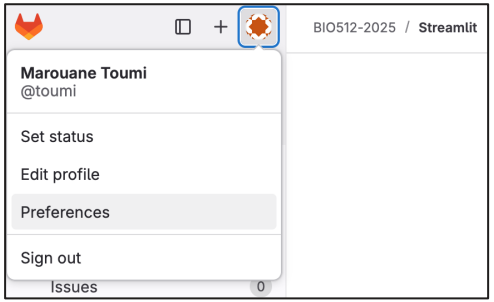
      Manually copy the entire key (should begin with *ssh-rsa*)

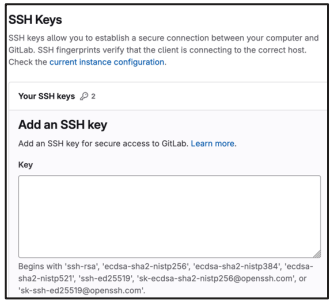
Streamlit
Tutorial setup

- Clone the `streamlit` repo on the class GitLab.

1. Set up your Gitlab access, **if you have never done it before**:
   a. First, we check if you already have SSH keys.
   b. Copy your public SSH key.
   c. Go to your Gitlab profile, preferences, and paste your SSH key !

Gitlab documentation





Streamlit
Tutorial setup

- Clone the `streamlit` repo on the class GitLab.

1. Set up your Gitlab access, **if you have never done it before.**
2. Open your terminal :

```
cd {path_to_streamlit_workshop}
git clone git@gitlab.epfl.ch:bio512-2025/streamlit.git
```


Streamlit
Tutorial setup

- Clone the `streamlit` repo on the class GitLab.
- Create an environment (optional but **recommended**):

Streamlit
Tutorial setup

- Clone the `streamlit` repo on the class GitLab.
- Create an environment (optional but **recommended**):
  - Open a terminal window
    - `cd {path_to}/streamlit`

Streamlit
Tutorial setup

- Clone the `streamlit` repo on the class GitLab.
- Create an environment (optional but **recommended**):
  - Open a terminal window
    - `cd {path_to}/streamlit`
    - `conda create -n Streamlit_workshop`

Streamlit
Tutorial setup

- Clone the `streamlit` repo on the class GitLab.
- Create an environment (optional but **recommended**):
  - Open a terminal window
    - `cd {path_to}/streamlit`
    - `conda create -n Streamlit_workshop`
    - `conda activate Streamlit_workshop`

Streamlit
Tutorial setup

- Clone the `streamlit` repo on the class GitLab.
- Create an environment (optional but **recommended**):
  - Open a terminal window
    - `cd {path_to}/streamlit`
    - `conda create –n Streamlit_workshop`
    - `conda activate Streamlit_workshop`
      - To exit this environment : **`deactivate`**

Streamlit
Tutorial setup

- Clone the `streamlit` repo on the class GitLab.
- Create an environment (optional but **recommended**):
  - Open a terminal window
    - `cd {path_to}/streamlit`
    - `conda create –n Streamlit_workshop`
    - `conda activate Streamlit_workshop`
      - To exit this environment : **`deactivate`**
    - `conda install pip`

Streamlit
Tutorial setup

- Clone the `streamlit` repo on the class GitLab.
- Create an environment (optional but **recommended**):
  - Open a terminal window
    - `cd {path_to}/streamlit`
    - `conda create -n Streamlit_workshop`
    - `conda activate Streamlit_workshop`
      - To exit this environment : **`deactivate`**
    - `conda install pip`
    - `pip install -r requirements.txt`

Streamlit
Tutorial setup

- Clone the `streamlit` repo on the class GitLab.
- Create an environment (optional but **recommended**):
  - Open a terminal window
    - `cd {path_to}/streamlit`
    - `conda create –n Streamlit_workshop`
    - `conda activate Streamlit_workshop`
      - To exit this environment : **`deactivate`**
    - `conda install pip`
    - `pip install –r requirements.txt`
- Check installation:
  - Open a terminal window, (activate the environment if not),
    - **`streamlit hello`**

Streamlit
Tutorial overview

- Tutorial based on the *GlucoCare Mom* project prototype

Streamlit
Tutorial overview

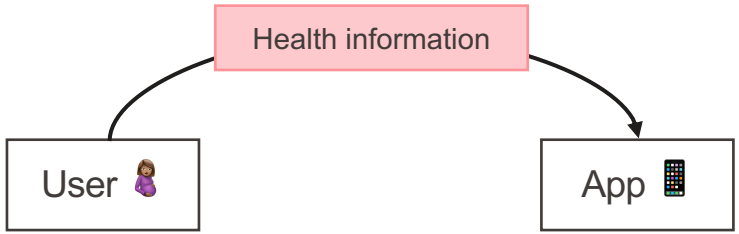
- Tutorial based on the *GlucoCare Mom* project prototype :
  - Subject: **Gestational Diabetes**

Streamlit
Tutorial overview

- Tutorial based on the *GlucoCare Mom* project prototype :
  - Subject: **Gestational Diabetes**
  - Solution : Create a smartphone application to help pregnant women living with gestational diabetes to better **understand** and **control** their blood sugar levels.

Streamlit
Tutorial overview

- Tutorial based on the *GlucoCare Mom* project prototype :
  - Subject: **Gestational Diabetes**
  - Solution : Create a smartphone application to help pregnant women living with gestational diabetes to better **understand** and **control** their blood sugar levels.

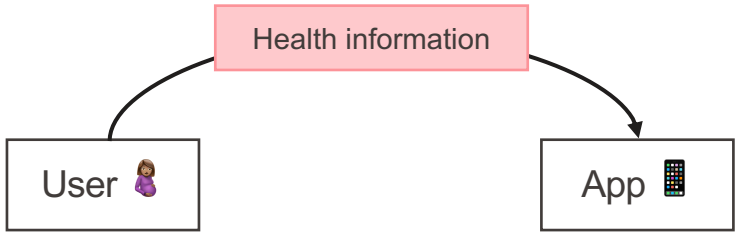


Health information

- General information,
- Blood sugar levels,
- Pregnancy,
- Medications,
- Mood…

Streamlit
Tutorial overview

- Tutorial based on the *GlucoCare Mom* project prototype :
  - Subject: **Gestational Diabetes**
  - Solution : Create a smartphone application to help pregnant women living with gestational diabetes to better **understand** and **control** their blood sugar levels.



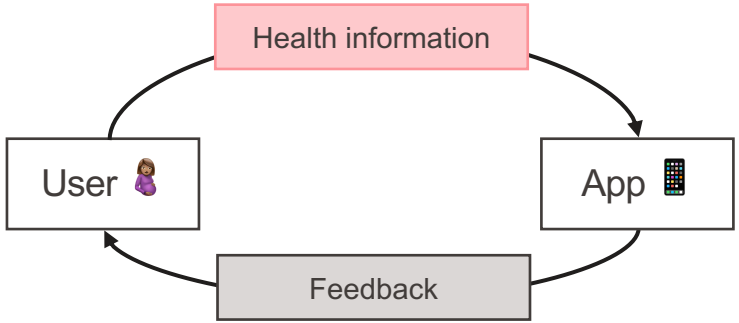
Health information

- General information,
- Blood sugar levels,
- Pregnancy,
- Medications,
- Mood…

➡️ Different types of data !

Streamlit
Tutorial overview

- Tutorial based on the *GlucoCare Mom* project prototype :
  - Subject: **Gestational Diabetes**
  - Solution : Create a smartphone application to help pregnant women living with gestational diabetes to better **understand** and **control** their blood sugar levels.

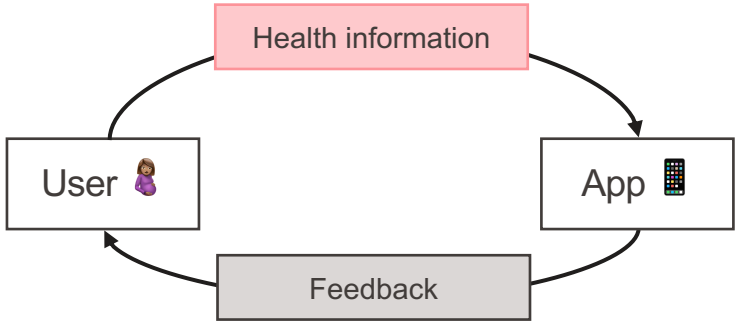


Feedback

- On a different page,
- Graphs,
- Metrics,
- Text explanation,

Streamlit
Tutorial overview

- Tutorial based on the *GlucoCare Mom* project prototype :
  - Subject: **Gestational Diabetes**
  - Solution : Create a smartphone application to help pregnant women living with gestational diabetes to better **understand** and **control** their blood sugar levels.



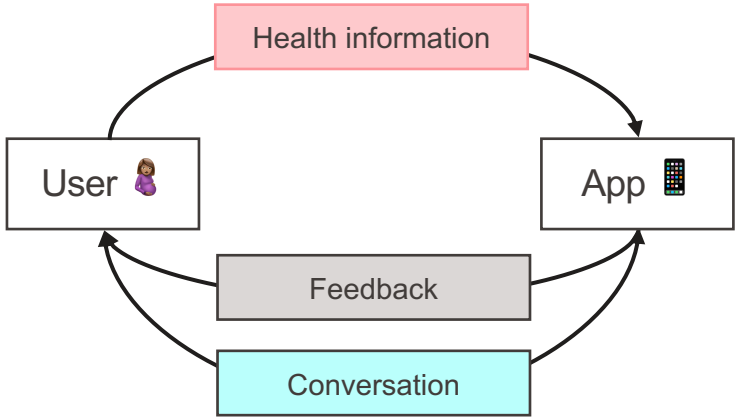
Feedback

- On a different page,
- Graphs,
- Metrics,
- Text explanation…

➡️ Different data processing !

Streamlit
Tutorial overview

- Tutorial based on the *GlucoCare Mom* project prototype :
  - Subject: **Gestational Diabetes**
  - Solution : Create a smartphone application to help pregnant women living with gestational diabetes to better **understand** and **control** their blood sugar levels.



Conversation

- Initialize a conversation,
- Define the context,
- Further insight generation,
- Personalize the chatbot…

➡️ We need a LLM !

Streamlit
LLM-derived features

- Two LLM-derived features
  - `generate_insight()` ➡️ textual explanation of data
  - `access_chatbot()` ➡️ personalized chatbot

Streamlit
LLM-derived features

- Two LLM-derived features
  - `generate_insight()` ➡️ textual explanation of data
  - `access_chatbot()` ➡️ personalized chatbot
- Use GPT model using OpenAI's Application Programming interface (API)
  - https://platform.openai.com/docs/overview

Streamlit
LLM-derived features

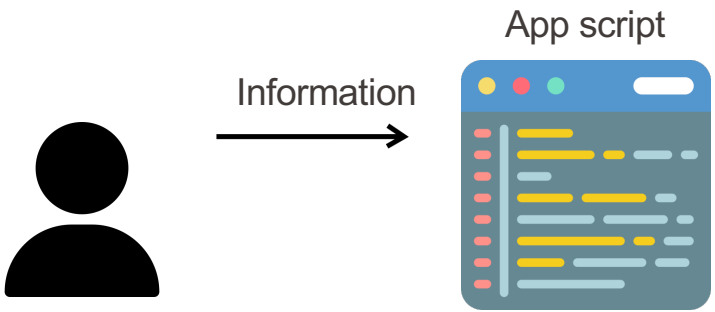
- Two LLM-derived features
  - `generate_insight()` ➡️ textual explanation of data
  - `access_chatbot()` ➡️ personalized chatbot
- Use GPT model using OpenAI's Application Programming interface (API)
  - https://platform.openai.com/docs/overview
- 🔑 API key = private access to any OpenAI model

Streamlit
LLM-derived features

- Two LLM-derived features
  - `generate_insight()` ➡️ textual explanation of data
  - `access_chatbot()` ➡️ personalized chatbot
- Use GPT model using OpenAI's Application Programming interface (API)
  - https://platform.openai.com/docs/overview
- 🔑 API key = private access to any OpenAI model
- We will use the GPT 3.5-Turbo model
  - https://platform.openai.com/docs/models/gpt-3.5-turbo

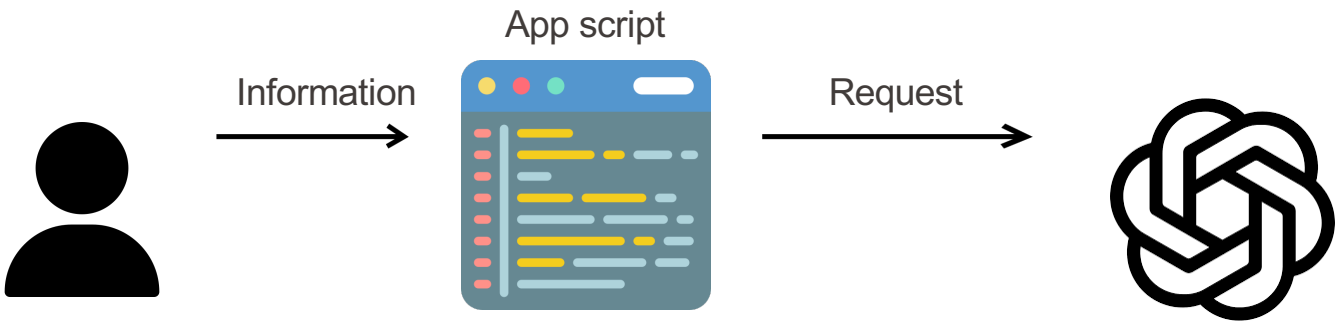
Streamlit
API interactions

- The user inputs an information



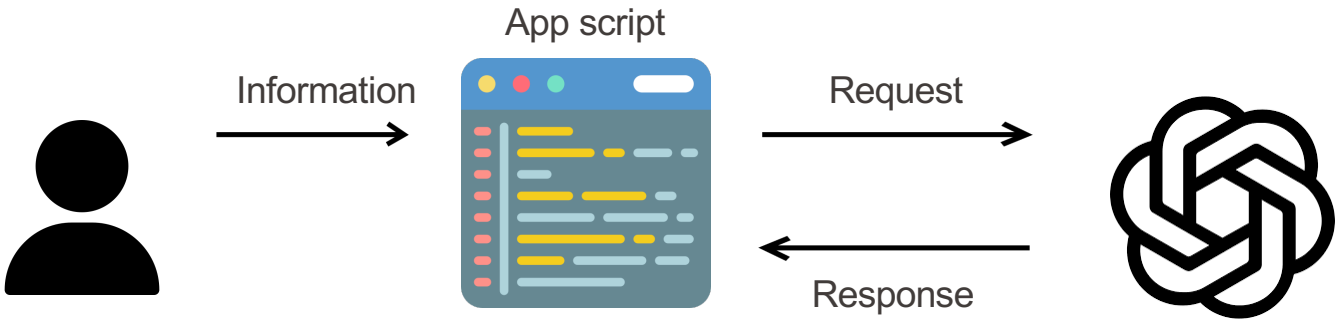
Streamlit
API interactions

- The user inputs an information
- The app sends a request to OpenAI's server with the API key.
  - https://github.com/openai/openai-python



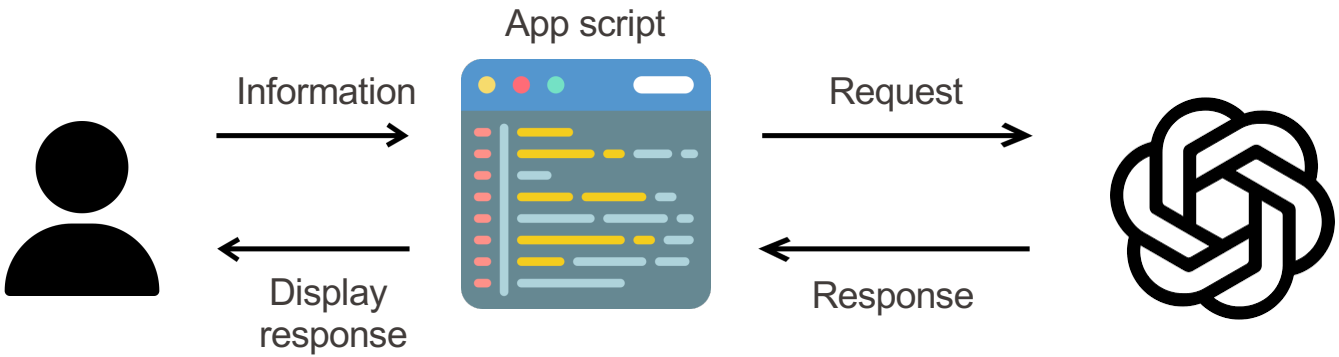
Streamlit
API interactions

- The user inputs an information
- The app sends a request to OpenAI's server with the API key.
  - https://github.com/openai/openai-python
- OpenAI's servers **generate a response** using GPT-3.5-turbo.

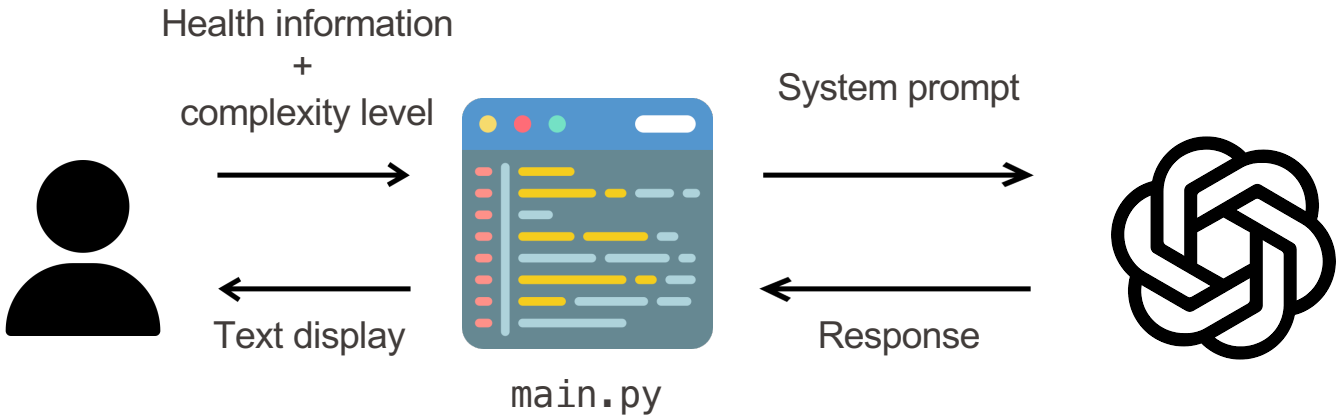


Streamlit
API interactions

- The user inputs an information
- The app sends a request to OpenAI's server with the API key.
  - https://github.com/openai/openai-python
- OpenAI's servers **generate a response** using GPT-3.5-turbo.
- The app receives the response and **shows it to the user**.



Streamlit
Insight generations overview



Streamlit
Chatbot overview

- Just like insight generations, but iterated over user messages :

1. **Initialization**:
   - Define the context (health information + expectations)
   - Set up the conversation history (Python dictionary stored in the Session State)

Streamlit
Chatbot overview

- Just like insight generations, but iterated over user messages :

1. **Initialization**:
   - Define the context (health information + expectations)
   - Set up the conversation history (Python dictionary stored in the Session State)

2. **FOR each new user message** :
   1. Get the user message.
   2. Generate the API response using the message history + the new user message
   3. Display the conversation it in the chat
      ➡️ st.chat_message()
   4. Save both the user message + API response in the message history,
      ⚠️ within the Session State

Projects

Tips for a successful project

- Start with a clear core idea.

Projects

Tips for a successful project

- Start with a clear core idea.
- Think about real users.

Projects

Tips for a successful project

- Start with a clear core idea.
- Think about real users.
- The prototype building shouldn’t limit your idea.

Projects

Tips for a successful project

- Start with a clear core idea.
- Think about real users.
- The prototype building shouldn’t limit your idea.
- Tell a good story.

Projects

Tips for a successful project

- Start with a clear core idea.
- Think about real users.
- The prototype building shouldn’t limit your idea.
- Tell a good story.
- Explore ideas that matter to you.