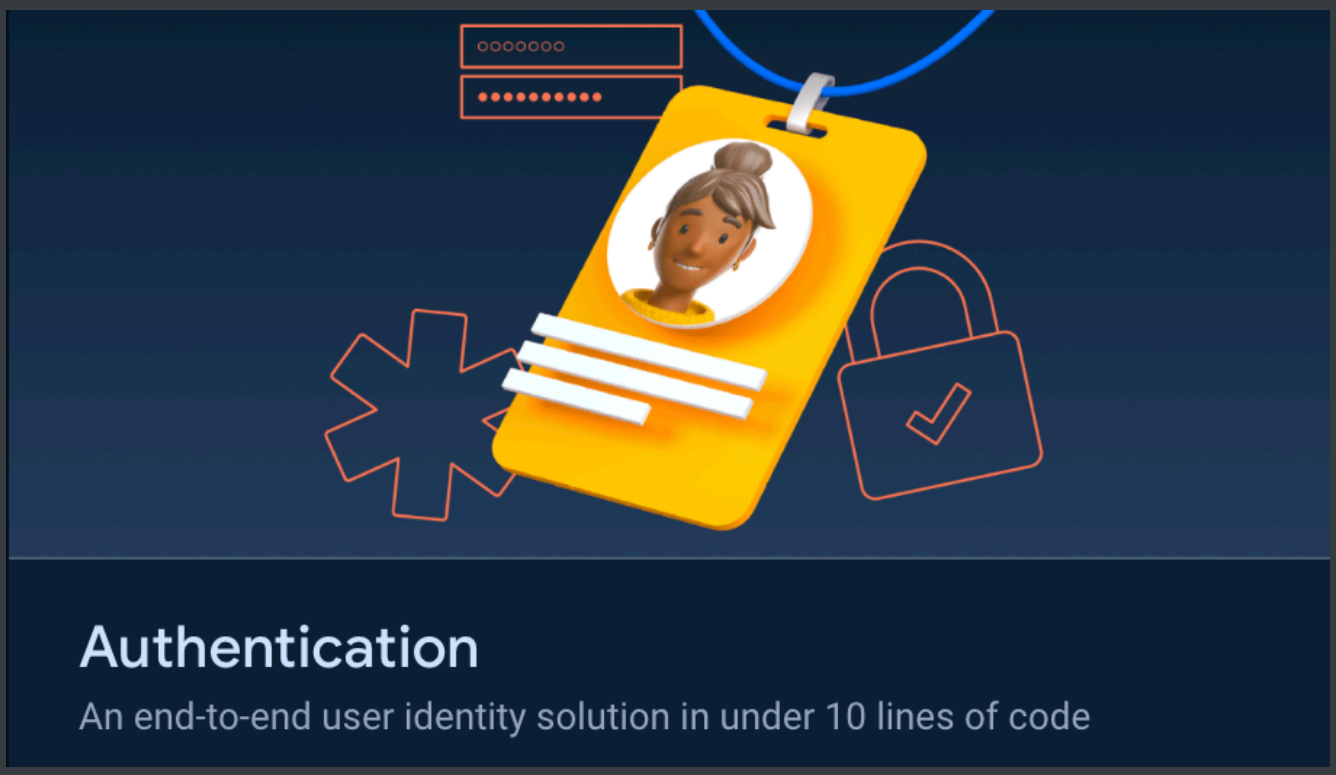
BaseSoAR Tutorial

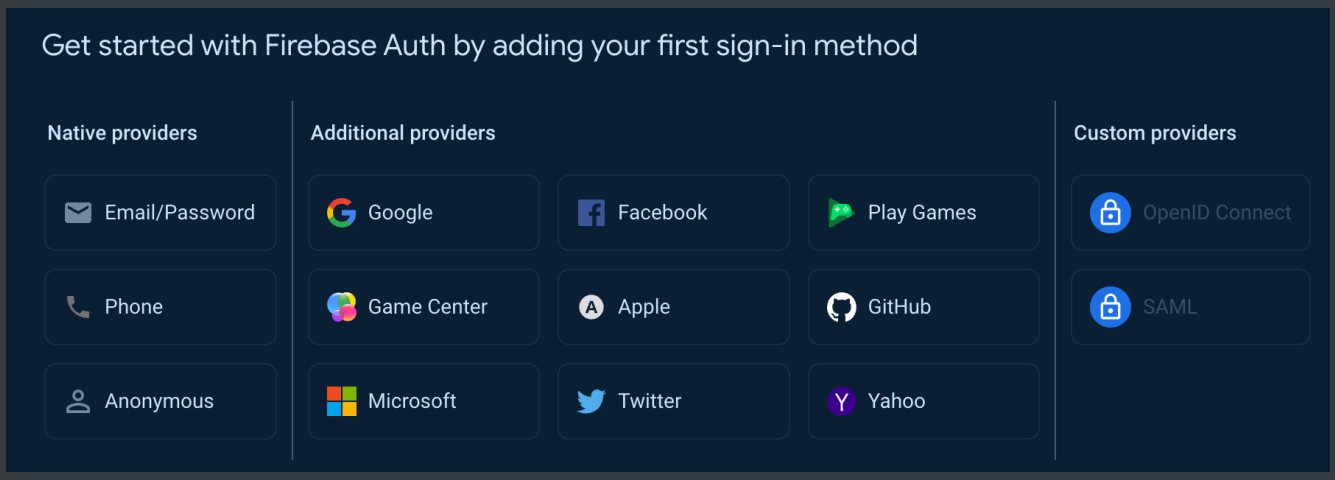
Pre-Requisite Software:

- Visual Studio Code (or any equivalent code editor / IDE)
- Python > 3.11
- Git installed with configured
- Google Firebase Access

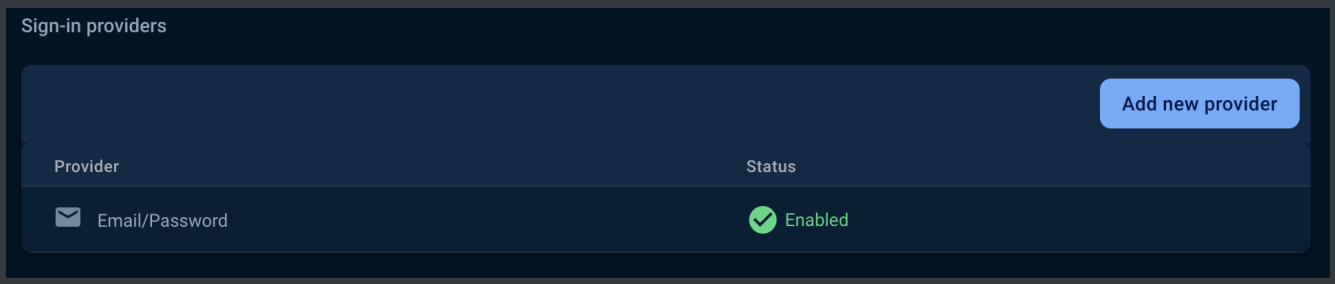
FIrebase Setting Up:

- Head over to https://console.firebase.google.com
- Click on Add Project & give this project a name
- Click on "Authentication" back on the main window:
- 

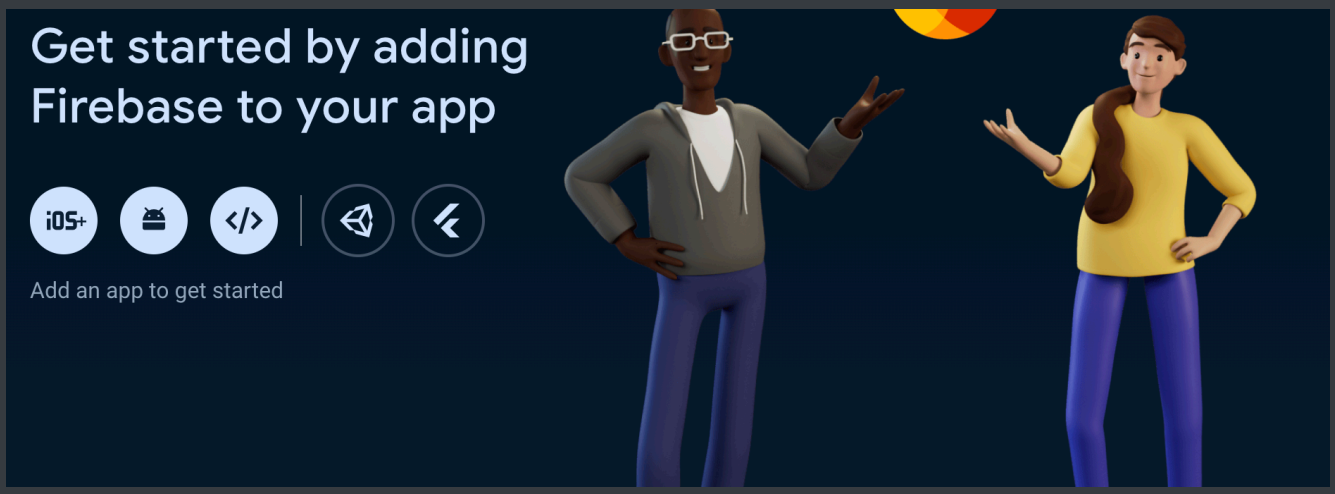
- Click on "Get Started"
- Click on "Email / Password" & enable the option of "Email / Password"
-



- When successful, this should be on the main window:
-



- Click on "Home" and select the `<>` option
-



- Register the App with the same project name and take note of the values stored in `const firebaseConfig`:

```
const firebaseConfig = {
  apiKey: "...",
  authDomain: "...",
  projectId: "...",
  storageBucket: "...",
  messagingSenderId: "...",
  appId: "..."
};
```

OpenAI API Key:

- **For accounts without credit card, the API Key will only work 1 month from the account creation date**
- Head over to "https://platform.openai.com/api-keys" & Sign in to your OpenAI account
- Click on "Create a Secret Key"
- Copy the API Key generated (starting with `sk-....`)

Clone the Repo!

- Create a folder in your any location and Open that Folder in VS Code:
- Spawn a terminal in that folder and run `git clone https://github.com/DangerousPotential/BaseSoAR.git`

Installing the Dependencies

- After cloning, you can install the project dependencies by opening a new terminal with VS Code
- Run: `pip install -r requirements.txt`

Setting Up with OpenAI API Key + Firebase Credentials

- Create a folder in the project's root directory named: `.streamlit`
- Inside `.streamlit` , create a file called `secrets.toml`
- In the file created, create a new line with `OPENAI_API_KEY = "sk...."`
- For the Firebase Configuration, we will need to add a line called `[firebase]`
- Copy the `firebaseConfig` value from the page after the `[firebase]` line
- Remove all `:` and replace it with `=`
- Add a new line with `databaseURL = ""`

```
OPENAI_API_KEY "sk-..."
[firebase]
apiKey = ""
authDomain = ""
projectId = ""
storageBucket = ""
messagingSenderId = ""
appId = ""
databaseURL = ""
```

Starting BaseSoAR:

We can now start our web-app by opening a Terminal and run:

```
streamlit run BaseSoAR.py
```

If you did not add Python to PATH, you can just use:

```
python -m streamlit run BaseSoAR.py
```

A web-app will spawn in your default browser. (**Note for Mac Users: Safari may not be able to show Streamlit Local Web Apps, use Chrome / Firefox**)

Editing the Structure & Content of the Web App:

To create a new page with minimal modifications, you can just duplicate `Chapter 1.py`, and start editing after `else` (line 9)

You can determine how many columns to store the buttons with `st.columns()`:

To create 4 columns we can use:

```
col1, col2, col3, col4 = st.columns(4)
```

We can now store the different buttons in each of the columns:

```
with col1:
  b1 = st.button("Button 1")
  b2 = st.button("Button 2")
with col2:
  b3 = st.button("Button 3")
with col3:
  b4 = st.button("Button 4")
with col4:
  b5 = st.button("Button 5")
```

Now we can control the content showing up in the web app when the different buttons are pressed with control statements:

```
if b1:
  st.markdown("Button 1 is pressed")
if b2:
  st.markdown("Button 2 is pressed")
```

To print images, math equations and formatted text, you can refer to the document `Streamlit Basics.pdf`

"Training" the Chatbot:

For different chapters, our knowledge base for the Chatbot would be different.

To create a new chatbot for different topics:

Place the PDF file into the `documents` folder

To make a newly trained chatbot, you can make a copy of `RAGChapter1.py` and name it differently (e.g. `RAGChapter2.py`) in the same directory and edit the line 16:

Change the name of the file loaded in from `Chapter1.pdf` to `Chapter2.pdf`

```
@st.cache_resource
def create_vector_storage(_documents):
    vectorstore = FAISS.from_documents(_documents, embedding =
OpenAIEmbeddings(openai_api_key = st.secrets["OPENAI_API_KEY"]))
    return vectorstore
pdf_path = "./documents/Chapter2.pdf" # Change This
pages = load_and_split_pdf(pdf_path)
vectorstore = create_vector_storage(pages)
```

Now, we can just need to import the `answer` function in the respective chapter pages:

For example, in `Chapter2.py` we will change:

```
from RAGChapter1 import answer
```

to:

```
from RAGChapter2 import answer
```