

2024

STREAMLIT CHATBOT

Designed by:
Jeffrey S. Saltz
Akit Kumar

Creating a Basic Streamlit Chatbot

1.	Online chatbot with Streamlit + Github	
	1.1	Setup GitHub account
	1.2	Sign up for Streamlit Community cloud
	1.3	Create a new App with Github Codespaces
	1.4	GitHub Codespace schema
	1.5	Streamlit Architecture
	1.6	Editing the App
	1.7	Deploying the App
	1.8	Editing the App Settings
	1.9	Multi Page app
2.	Using Google Colab	
	2.1	Creating first streamlit app on Google Colab
	2.2	Edit the App file
	2.3	Multi page app in google colab

Help from ChatGPT:

Steps to use github without installing git on your laptop
(and also how to deploy on streamlit cloud):

<https://chatgpt.com/share/df122705-e3b9-4cf4-bfa6-8ba04a1d5b5c>

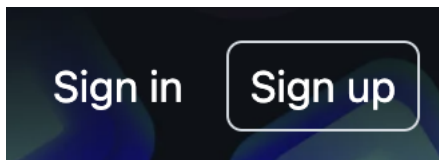
Setup GitHub Account

GitHub is a cloud-based service that enables developers to store, collaborate, and manage their codebase.

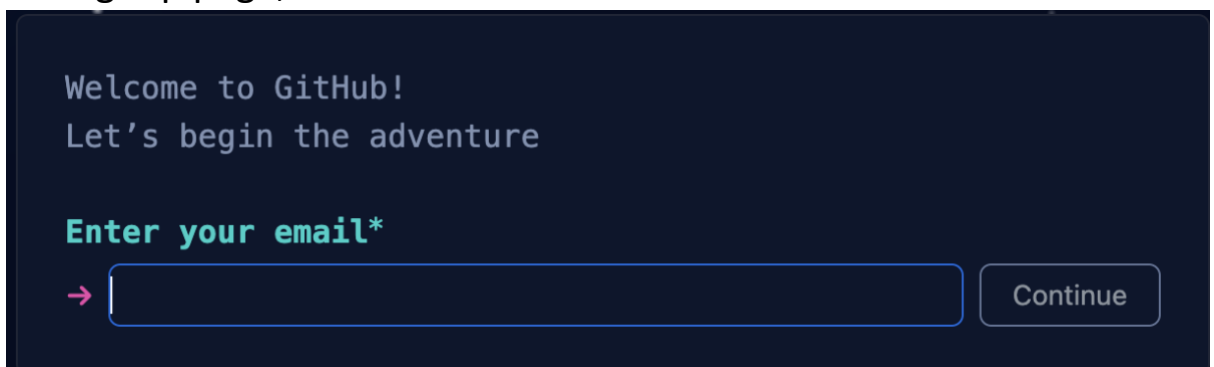
If you already have an account with GitHub, you can skip this page.

To create an account in GitHub follow the following steps:

1. Navigate to <https://github.com/>
2. Click on Signup



3. On Signup page, Enter the Email address and click on Continue



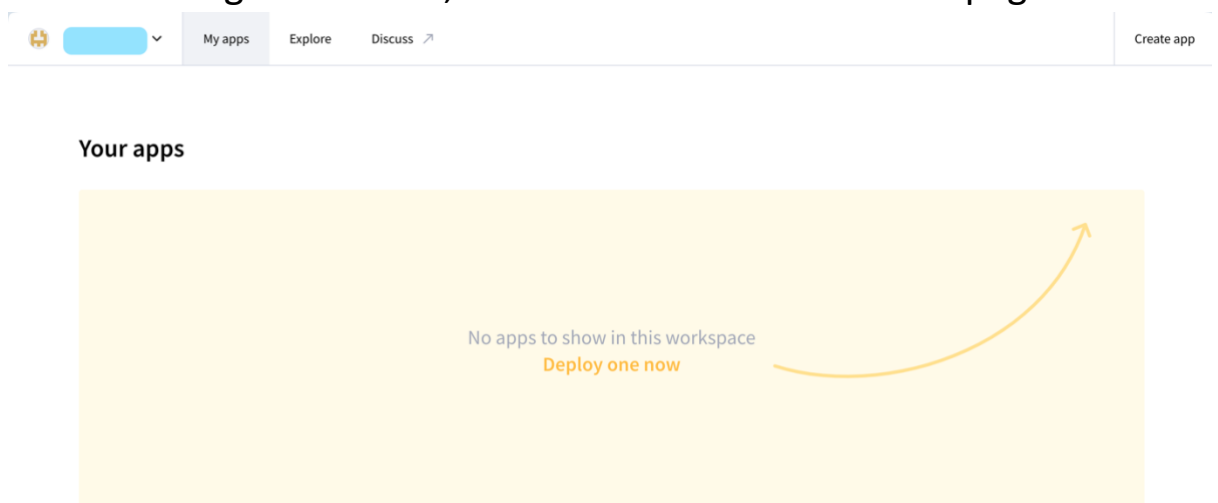
4. Create the Password. Note: *Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.*
5. Enter a unique username. If the username is taken, the website will suggest usernames that are not taken.
6. Click on continue.
7. You need to verify the account using the Puzzle and Launch code sent by email.
8. Login into your new account and set up the new account.

Sign up for Streamlit Community cloud

Streamlit Community Cloud and GitHub codespace allow developers to build and deploy Streamlit applications on the cloud.

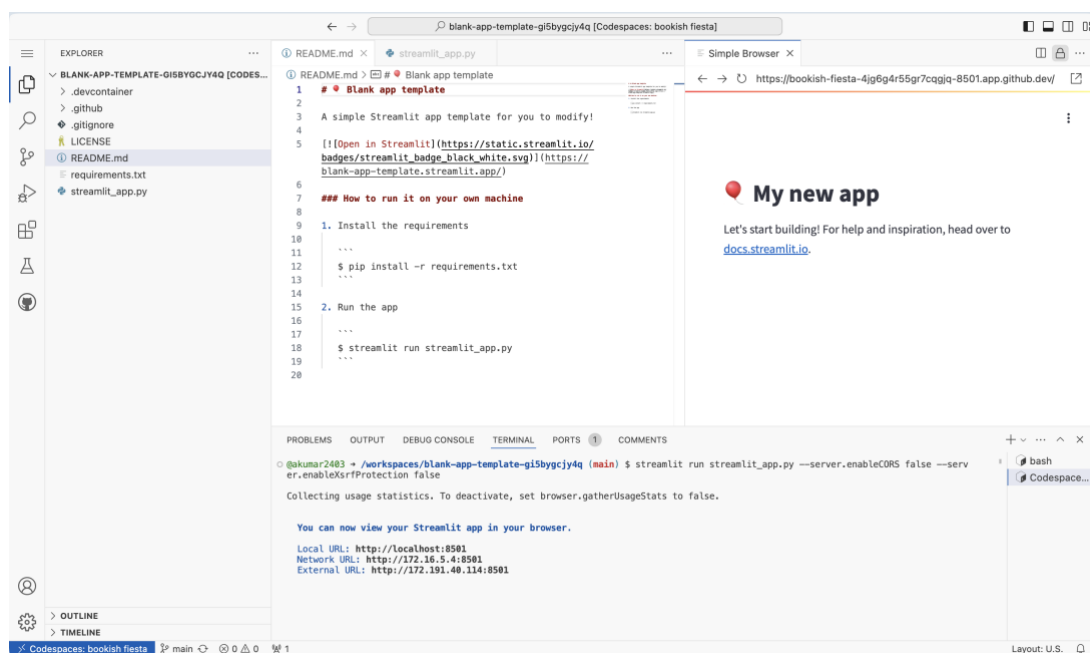
Do the following to create a Streamlit Community cloud account:

1. Go to <https://share.streamlit.io/signup>
2. Click on “Continue to sign-in”
3. If you already have a Streamlit Community cloud account, you can continue to sign in and skip the sign up process.
4. There are three options for creating account:
 - a. “Continue with Google” to create an account you’re your google account.
 - b. “Continue with GitHub” to create with your GitHub account.
 - c. “Sign up” to create account with your email id.
- 5. It is recommended by Streamlit to create an account with GitHub.**
6. Connect your GitHub account with the Streamlit. Authorize the Streamlit for your GitHub account.
7. Finish setting up your
8. After creating an account, the browser will land on this page.

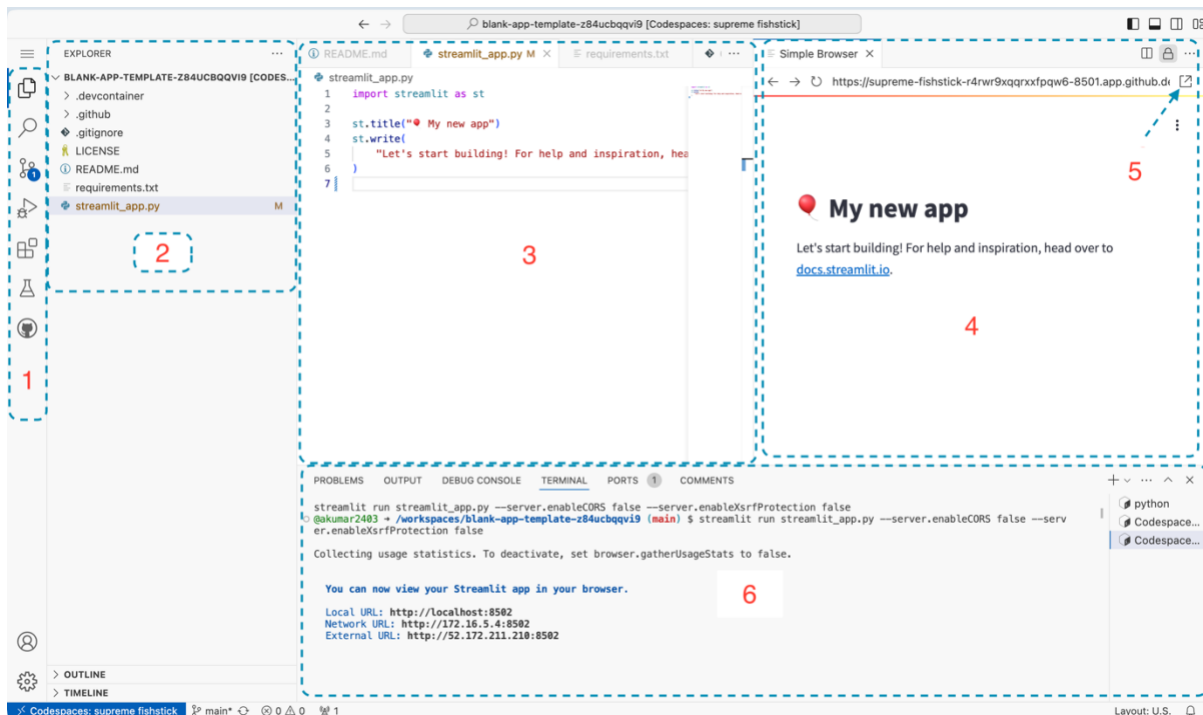


Create a new App with Github Codespaces

1. Once logged into your Streamlit cloud account, Click on “Create app” (top right corner).
2. Streamlit provides two options to start with the project:
 - a. If you have a GitHub repository that already contains the code of your Streamlit project, then you can directly provide the repository link.
 - b. ***Streamlit provides multiple templates that are designed with specific requirements. Developers can use these templates to build applications in a shorter time. You can fork from one of their existing templates.***
3. Click on “Nope, create one from a template”.
4. Under “Deploy from a template”, Click on “Blank app” and “Open GitHub Codespaces to immediately edit this app in your browser”.
5. Click on the “Deploy” button. This action will create a new repository in your github account with file shown below.
6. After complete processing, you will land on a webpage that will resemble Visual Code Studio (don’t worry if you don’t know about VSCode)



GitHub Codespace schema



Lets understand the structure of the Codespace. The above picture is marked with numbers from left to right in ascending order.

1. Activity bar: This bar provides you option to switch between views and gives you additional context-specific indicators, like the number of outgoing changes when Git is enabled(3rd option from top to bottom).
2. Primary side bar: This box contains different views like the Explorer to assist you while working on your project, github commit option, etc when clicked for the option on Activity bar.
3. Status bar: This space opens files when selected in primary side bar. You can edit and save the files from here.
4. Simple Browser: Whenever you run Streamlit app or open Codespace for the first time, This mini browser will open and shows the streamlit app.
5. Pop out button: When clicked on this button, the simple browser will open in a new tab of your browser.

6. Terminal: Here you can start and stop a Streamlit app. Whenever the Codespace begin for the first time following command will run:

```
streamlit run streamlit_app.py --  
server.enableCORS false --  
server.enableXsrfProtection false
```

This command runs Streamlit server on Codespace and defines the entry point as *streamlit_app.py*. If you want to change the entry point of your Streamlit app to different file, change the name of the .py and replace it with new .py file.

Streamlit Architecture

Streamlit apps have a client-server architecture. The Python backend is the server, while the frontend viewed through a browser is the client. When running locally, both server and client run on your computer. When deployed, they run on separate machines.

When you run `streamlit run your_app.py`, your computer starts a Streamlit server that performs computations for all users. This server runs on the machine where the app was initialized, which is called the host.

The browser frontend is the client. When you view/run the app locally, the server and client run on the same machine. But when users access the app remotely, the client runs on a different machine from the server.

Editing the App

Let add some basic features to the streamlit_app.py.

1. Adding a dataframe to the app

```
df = pd.DataFrame({  
    'first column': [1, 2, 3, 4],  
    'second column': [10, 20, 30, 40]  
})  
  
st.write(df)
```

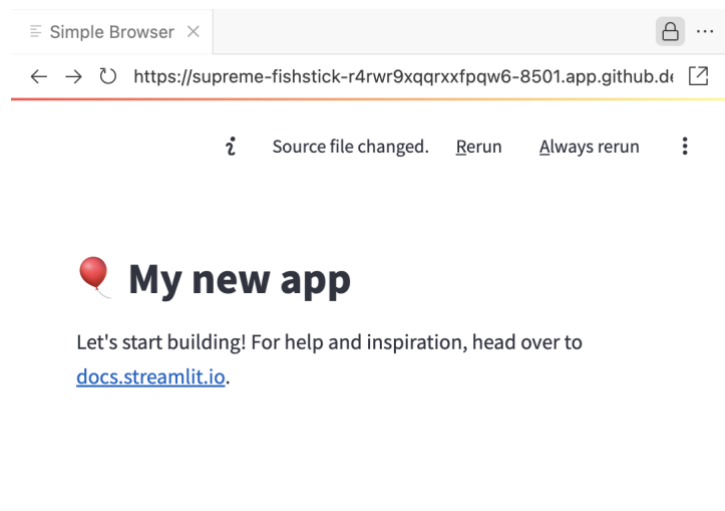
2. Adding a select dropdown to the app

```
option = st.selectbox(  
    'Which number do you like best?',  
    df['first column'])  
  
'You selected: ', option
```

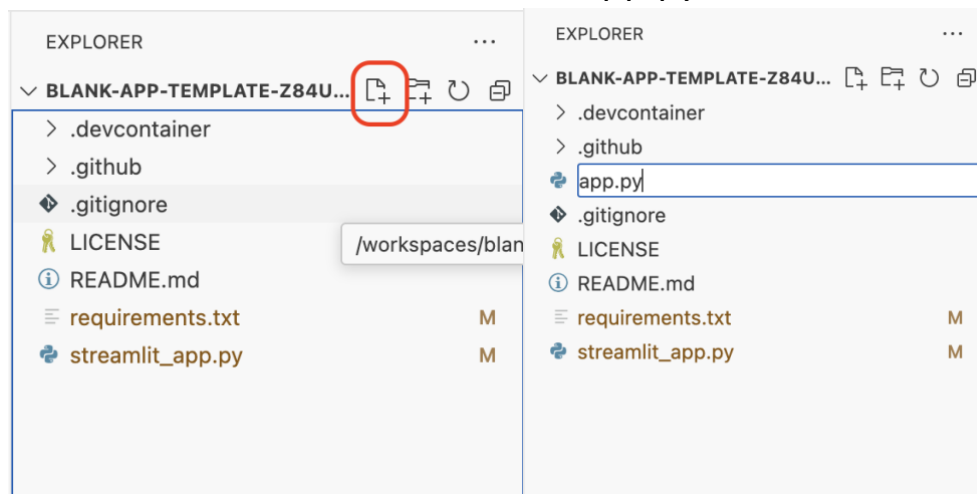
3. Adding a sidebar to the app

```
add_selectbox = st.sidebar.selectbox(  
    'How would you like to be contacted?',  
    ('Email', 'Home phone', 'Mobile phone')  
)
```

Whenever you save this app, the browser asks you to reload the app with two options; Rerun (reload the browser once), and Always rerun (reload automatically whenever there is a new change). It is advised to select “Always rerun” such that changed are reflected automatically.



To create a new app in the CodeSpace, create new file using the marked button below. Name the file as `app.py` and hit Enter.



Add the following line of code to the `app.py`:

```
import streamlit as st

st.title("New title")
st.write("New app!!!")
```

Run the new app using the command in terminal.

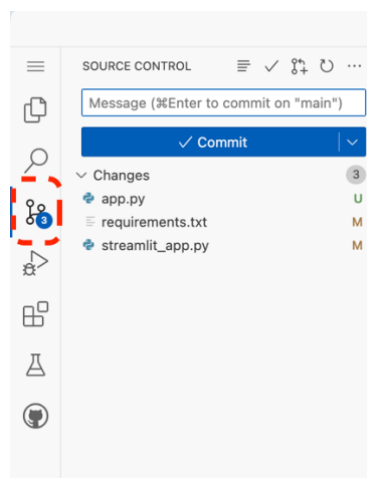
```
streamlit run app.py --server.enableCORS false
--server.enableXsrfProtection false
```

The streamlit app will look something like below.

New title

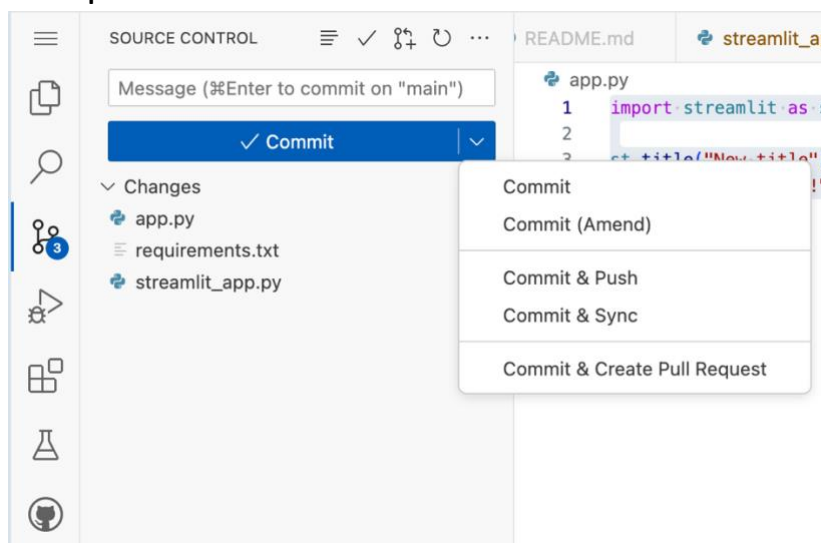
New app!!!

Next, updates need to be pushed to GitHub. Click on the Github Icon on the Activity bar (shown below). As seen, there are changes in streamlit_app.py, app.py and requirements.txt



Add a message in the Message input, for example, “changes to streamlit_app.py”.

Click on the drop down next to “Commit” button.



Click on “Commit & Push”.

You can view the new changes in the GitHub repository where this app is stored.

Deploying the App

App Dependencies

The main reason apps fail to build properly on Streamlit Community Cloud is that it can't find your dependencies. There are two types of dependencies: *Python* and *external dependencies*. Python dependencies are packages that are imported into your script, while external dependencies are installed with apt-get outside the Python environment.

Python dependencies

For every import statement in your app, you need to tell Streamlit Community Cloud to install these dependencies through a python package manager. To do so, you can add those dependencies into *requirements.txt*. For example, if you have imported following libraries

```
import streamlit as st
import pandas as pd
import numpy as np
```

then you need to add following libraries to the *requirements.txt*:

```
streamlit
```

```
pandas  
numpy
```

The *requirements.txt* file should exist in the root directory of the app. Python build-in libraries are not required to be added to *requirements.txt*.

External dependencies

Streamlit Community Cloud is built on Debian Linux. If your Streamlit app requires apt-get installation, add those dependencies to the *packages.txt* file. *packages.txt* should exist in root directory.

Secrets management

Secret values just as OpenAI API key, database credentials, or passwords should not be included directly in the app or pushed to GitHub. Streamlit allows users to store these secret values in a file which is not pushed to GitHub and pass them as environment variables.

Deploying Streamlit app on Streamlit Community Cloud with secret management

After commit and push to the *streamlit_app.py* on Github, go to <https://share.streamlit.io>. Click on “Create App”, select “Yup, I have an app”. Under “*Repository*”, select the repository which contains your code. Branch should be main and Main file path will be the file containing your app code, in this case, *streamlit_app.py*. Click on advanced settings.

Branch

main

Main file path

streamlit_app.py

App URL (optional)

blank-app-2-cbkjbsnjescjvv4cbbboem8 .streamlit.app

Domain is available

Advanced settings...

Deploy!

Click here

An advanced settings pop up will appear.

Advanced settings

Python version

3.11

Secrets

Provide environment variables and other secrets to your app using TOML format. This information is encrypted and served securely to your app at runtime. Learn more about Secrets in our docs. Changes take around a minute to propagate.

```
db_username = "Jane"
db_password = "12345qwerty"
```

Save

Enter your secret value in format of key and value pair, where key is the variable which will be used in the .py file and value is the secret value. Click on the Save button and deploy the app by clicking “Deploy!” button. The streamlit_app.py app will load on the browser.

How would you like to be contacted?

Email

Share ☆ ↺ ⋮

My new app

Let's start building! For help and inspiration, head over to docs.streamlit.io.

	first column	second column
0	1	10
1	2	20
2	3	30
3	4	40

Which number do you like best?

1

After Deploying the app, you can access your secrets as environment variables or by querying the `st.secrets` dict directly in your `streamlit_app.py` or any of your .py file.

```
st.write("DB username:", st.secrets["db_username"])
st.write("DB password:", st.secrets["db_password"])
```

The app will look like this:

How would you like to be contacted?

Email

Share ☆ ↺ ⋮

My new app

Let's start building! For help and inspiration, head over to docs.streamlit.io.

	first column	second column
0	1	10
1	2	20
2	3	30
3	4	40

Which number do you like best?

1

You selected: 1

DB username: Jane

DB password: 12345qwerty

As seen, the secret values are now accessible on the app and are not uploaded on the GitHub repository.

Secret management on local systems

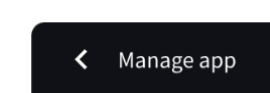
When working on Streamlit locally, create a folder named *“.streamlit”* in the root directory. Add a file name *“secrets.toml”* in the folder. In the file, add the values as added on the cloud. In .py file, utilizing these key values pairs should be same as cloud.

Editing the App Settings

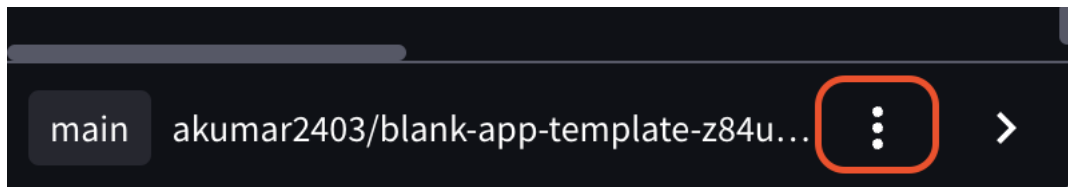
Update the URL

After the app is deployed on Streamlit Community Cloud, it will have a <<GitHubRepositoryURL>>.streamlit.app. This might reduce the readability or accessibility of the URL by search engines such as Google or Bing. Thus, it is beneficial to update the URL of the deployed app to something relevant to the app feature. Since this is a tutorial for IST 688, let's see how to change the URL to ist688tutorial.streamlit.app.

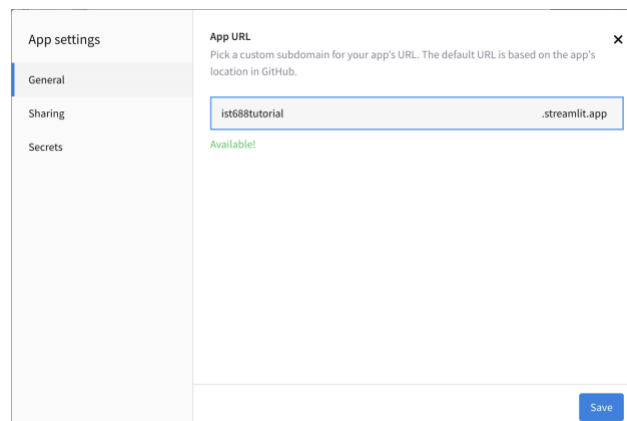
There will be a slider button at the bottom right corner of the app named “*Manage app*”. This button is visible only to user who have admin right to the app.



After click on the button, click on the three dot options and click on settings.



Under General tab, enter the “ist688tutorial”. If the URL is available, an “*Available*” with green color will appear below. If not, an message with red color will appear to choose other name. Next, Click on save.

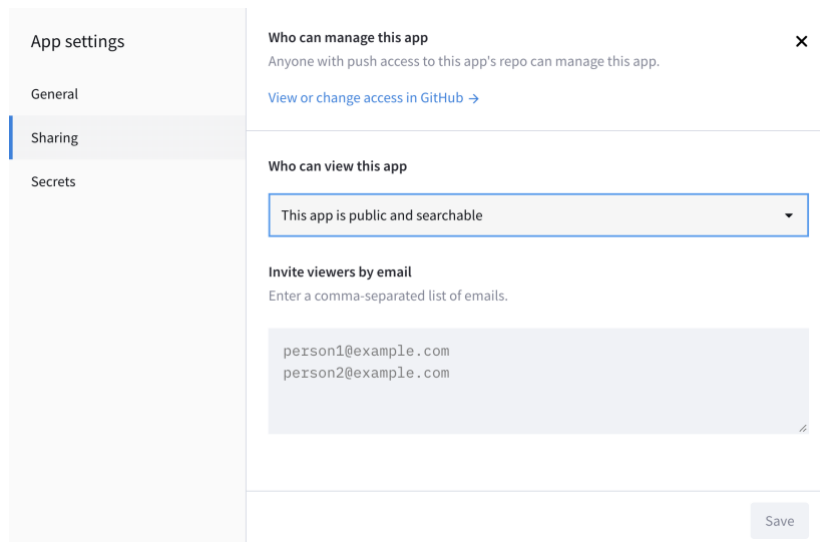


The app URL will be changed to <<entered URL>>.streamlit.app.

Public vs Private

When you deploy a app to Streamlit Community Cloud, the visibility is automatically set to public. Further, it is indexed by search engines like Google and Bing on a regular basis. This makes your app discoverable, as anyone can find it by searching for its custom subdomain or by searching for the app's title.

However, if you want to keep your deployed app private, you can do so by going to the app setting. Click on Manage app, then select options(three vertical dots), and settings. Under app setting, click on sharing.



The screenshot shows the 'App settings' dialog box with the 'Sharing' tab selected. On the left, a sidebar lists 'App settings', 'General', 'Sharing' (highlighted), and 'Secrets'. The main content area has a title bar 'Who can manage this app' with a close button 'x'. Below it, a message states 'Anyone with push access to this app's repo can manage this app.' with a link 'View or change access in GitHub →'. A horizontal separator follows. Below the separator, the section 'Who can view this app' contains a dropdown menu currently set to 'This app is public and searchable'. Underneath is the 'Invite viewers by email' section with the instruction 'Enter a comma-separated list of emails.' and a text input field containing 'person1@example.com' and 'person2@example.com'. At the bottom right of the dialog is a 'Save' button.

As you can see this app is public and searchable, change to “only specific people can view this app”. Invite viewers by entering their emails on the text input below and Save. This will make your app private and accessible to only those who are invited.

For IST 688, Labs should be public and HWs must be private.

Multi Page app

With the 1.36 update of Streamlit, the app can now support Multi Page application using `st.navigation` and `st.Page` methods. These methods allow user to combine multiple `.py` files as Pages into your main app. The pages will can be selected from the left side panel.

st.Page

This method let you define a page. The first and only required argument for defining your Streamlit app's pages is the source, which can be either a Python file or a function. When using Python files, the page files can be located in a subdirectory or a superdirectory, but the path must be relative to the app's entrypoint file. To create a Page for a python file follow the below code

```
create_page = st.Page("fileName.py",  
title="Enter page title here",  
icon=":material/add_circle:")
```

st.navigation

This method should be used in your main streamlit app `.py` file. This method act as an entrypoint for your pages. It bind all the pages that are attached to this method to the left side bar. let's create two python scripts named `Create.py` and `Delete.py`. Create pages using `st.Page` for these two python scripts. Attach these two pages to the left side panel named "Create" and "Delete" using `st.navigation`.

```
your-repository/  
├─ create.py  
├─ delete.py  
└─ streamlit_app.py
```

In `create.py`, write `st.write("Welcome to create.py")`.
In `delete.py`, write `st.write("Welcome to delete.py")`.

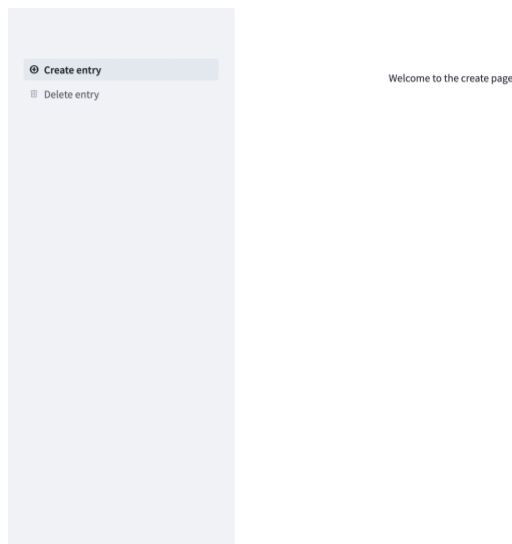
The streamlit_app.py code:

```
import streamlit as st

create_page = st.Page("create.py",
title="Create entry",
icon=":material/add_circle:")
delete_page = st.Page("delete.py",
title="Delete entry",
icon=":material/delete:")

pg = st.navigation([create_page, delete_page])
st.set_page_config(page_title="Data manager",
page_icon=":material/edit:")
pg.run()
```

The landing page would be create.py as it was first in the list in st.navigation. The left side panel contains two pages “*Create entry*” and “*Delete entry*”. When clicked on “*Delete entry*”, the delete.py page will load.



For IST 688, students need to create an main app.py file which will contain pages for labs and HWs such as Lab1, Lab2, Lab3, etc, on the left side panel.

For more reference visit: <https://docs.streamlit.io/develop/concepts/multipage-apps/page-and-navigation>

Using Google Colab

Creating first streamlit app on Google Colab

1. Create a “app.py” file in your local system.
2. Write the following code in it:

```
import streamlit as st
st.title("IST688")
st.write("Welcome to IST 688 course")
```

3. Create a new file named “requirements.txt” and add “streamlit” to the file.
4. Use your personal gmail or g.syr.edu account to login into Google drive.
5. Create a folder named “IST688” on your google drive.
6. Upload the app.py and requirements.txt to the IST688 folder.
7. Create a new Google Collaboratory file named “run_streamlit_app.ipynb” in IST688 folder and run the upcoming code in each cell.
8. To run streamlit app on Google Colab, we need to install a localtunnel. localtunnel exposes the localhost to the world for easy testing and sharing.

```
!npm install localtunnel --silent
```

9. Next, we mount the google drive on Google Colab.

```
import os
from google.colab import drive
drive.mount('/content/drive')
```

10. Next, we change the directory to IST688

```
os.chdir("/content/drive/MyDrive/IST688/")
```

11. We install the packages required using requirements.txt

```
!pip install -q -r requirements.txt
```

12. Now our system is ready to run streamlit app. Since localtunnel will expose the system to world, we need to know the Endpoint of the system.

```
import urllib
print("Password/Endpoint IP for localtunnel is:",urllib.request.urlopen('https://ipv4.i  
canhazip.com').read().decode('utf8').strip(  
"\n"))
```

13. The IP address displayed as the output is the IP of local system and a password when logging into the streamlit app
14. Next, we run the streamlit in background and store the logs in “log.txt” for future analysis.

```
!streamlit run app.py &>log.txt &
```

15. Lastly, we expose the local system to the world using the localtunnel.

```
!npx localtunnel --port 8501
```

16. The output from this code will generate a link. Click on the link and it will ask for a password. Enter the IP address we optioned from step 13. Click on “Click to Submit”.

To access the website, please enter the tunnel password below.

If you don't know what it is, please ask whoever you got this link from.

Tunnel Password:

Enter password here

Click to Submit

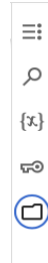
17. The streamlit app will load on the browser.

IST688 

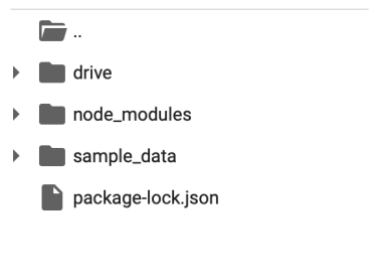
Welcome to IST688 course

Edit the App file

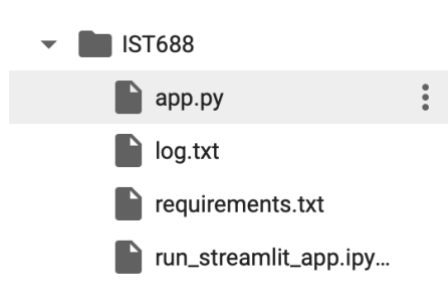
When the `run_streamlit_app.ipynb` file is open in google colab, click on the file explorer icon on the left activity panel.



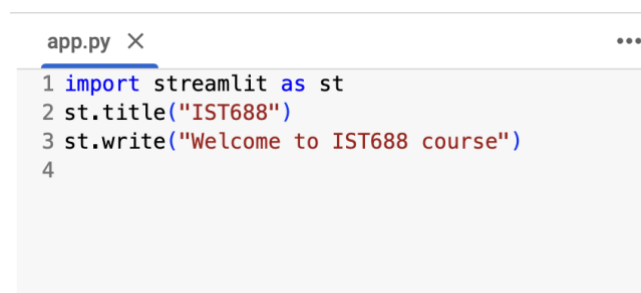
Since the google drive is already mounted, you can see your drive in the Explorer named “drive”.



Open your drive, Mydrive, and IST688 folder. Click on `app.py`.



A file editor will open on the right side of the browser.

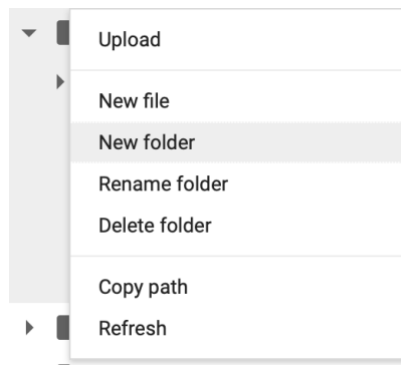


`app.py` can be edited here.

Multi page app in google colab

To create a multi-page app in google colab, you can follow the similar instructions in the [Multi Page app](#).

1. In google colab, Create a new folder under “IST688” folder named “labs”. To create a folder, click the options next to the “IST688” (three vertical dots). Select New folder and name it as “labs”.



2. Under labs folder, create two .py files named lab1.py and lab2.py.
3. Double-click on lab1.py and lab2.py. Both files will open in the file editor on the right panel.
4. Add the following code to lab1.py.

```
import streamlit as st
st.write("This is lab1")
```

5. Add the following code to lab2.py

```
import streamlit as st
st.write("This is lab2")
```

6. Now create pages in app.py for lab1.py and lab2.py and add to st.navigation.

```
lab1_page = st.Page("labs/lab1.py",
title="Lab 1")
lab2_page = st.Page("labs/lab2.py",
title="Lab 2")
pg = st.navigation([lab1_page, lab2_page])
```

```
st.set_page_config(page_title="Lab  
manager")  
pg.run()
```

7. Save all the files.
8. If the streamlit app is not running, follow the previous section to run the streamlit app on google colab. If it is running in a separate tab, open that tab. Refresh the page or click on “always rerun”.