# Project Document

## PROJECT: ELECTRICITY BILL CALCULATOR

### SUMMARY

Design a **Streamlit** web application that allows users to input their electricity consumption (e.g., in kWh) and calculates the total bill based on predefined unit rates.

This Python application calculates electricity costs based on user input. It offers two modes:

- Input meter readings (previous and current)
- Manually input the number of units

It uses '**Streamlit**' library that an open-source Python library for building interactive web apps using only Python. It's ideal for creating dashboards, data-driven web apps, reporting tools and interactive user interfaces without needing HTML, CSS or JavaScript.

## PRE REQUISITES

- **Python** - support version 3.9 to 3.13.
- **Python environment manager** (recommended) - Environment managers create virtual environments to isolate Python package installations between projects.
- **Code editor –** E.g. VS Code

### How to create an environment using venv

1. Open a terminal and navigate to your project folder.

```
cd myproject
```

2. In your terminal, type:

```
python -m venv .venv
```

3. A folder named ".venv" will appear in your project. This directory is where your virtual environment and its dependencies are installed.

#### Activate your environment

4. In your terminal, activate your environment with one of the following commands, depending on your operating system.

```
# Windows command prompt
.venv\Scripts\activate.bat

# Windows PowerShell
.venv\Scripts\Activate.ps1

# macOS and Linux
source .venv/bin/activate
```

5. Once activated, you will see your environment name in parentheses before your prompt. "(.venv)"

**Install Streamlit in your environment**

6. In the terminal with your environment activated, type:

   ***pip install streamlit***

**Install pandas in your environment**

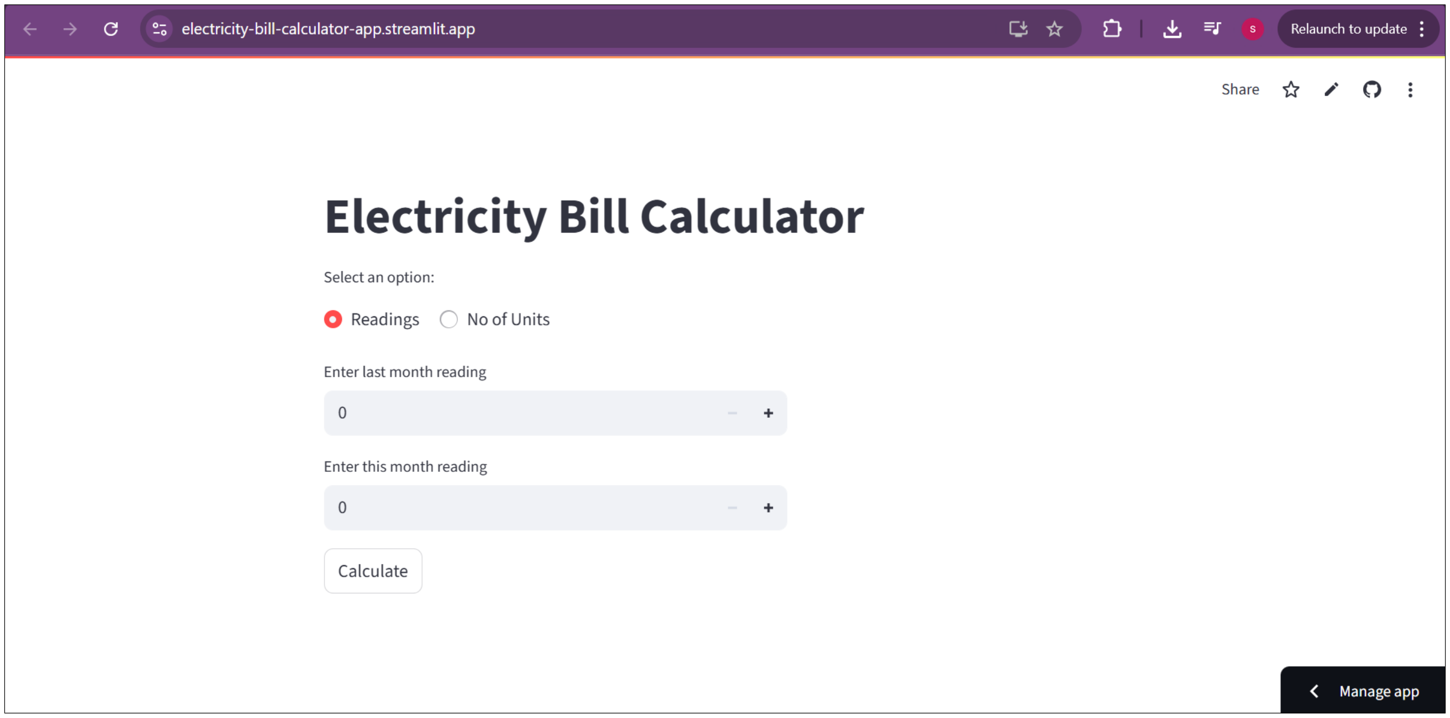
6. In the terminal with your environment activated, type:

   ***pip install pandas***

**To run a streamlit commands**

***Streamlit run your_script.py [-- script args]***

## Web Application Preview:



Electricity Bill Calculator

Select an option:

Readings [✓] No of Units

Enter last month reading

0

Enter this month reading

0

Calculate

## Code Overview:

From this code basically happened is design the user interface of getting user inputs to calculate electricity bill and display the total cost he/she must pay. Instead of this HTML/CSS/JavaScript can easily design a web application using this way. Also, stylish designs can be added to the user interface. So, let's dive deep to see what happens from this code.

Firstly, import the relevant libraries first. In here two libraires were used. Those are

1. Streamlit
2. Pandas

Then we can set a title for this web application like below.

```
st.title("Electricity Bill Calculator",width="stretch")
```

Then, initialize the variables first as good coding practice.

```
#Initialize Variables
noOfUnits = 0
```

On this web application, we can design the user interface as we wanted. So, the entire layout is divided into three columns and column width are set as below giving more portion to column 1. In here all the components aligned on column 1.

```
# Create columns to control layout
col1, col2, col3 = st.columns([3, 1, 1])
```

**The entire user interface is designed to take user inputs from two ways.**

1. Enter the units (kwh) last month and this month readings
2. Directly enter the number of units(kwh)

So, I have added two radio buttons to take the user option.

```
with col1:
    # Radio button to choose input type
    status = st.radio("Select an option:", ['Readings', 'No of Units'], horizontal=True)
```

If the user option (status) is "**Readings**", then two user input fields will appear on screen to take last month and this month units reading.

Those two user input values will be assigned to two variables called preReading and curReading, will calculate the number of units that used.

If no of units used are less than last month reading, warning message will pop saying "`Current reading must be greater than or equal to previous reading`"

```
if status == "Readings":
        preReading = st.number_input("Enter last month reading", key="pre",
        min_value=0,step=1, format="%d")
        curReading = st.number_input("Enter this month reading", key="cur", min_value=0,
        step=1, format="%d")

        if preReading and curReading:
            preReading = int(preReading)
            curReading = int(curReading)

            if curReading >= preReading:
                noOfUnits = curReading - preReading
            else:
                st.warning("Current reading must be greater than or equal to previous
                reading.")
```

If the user option (status) is "No of Units" user input field will pop up to take the number of units used by he/she.

```
elif status == "No of Units":
        noOfUnits = st.number_input("Enter number of units used", key="units",
                    min_value=0, step=1, format="%d")
```

Now, the user inputs have taken to variables is done and need to calculate the total bill amount. For that calculation button has been added.

In here mainly used try, except to avoid the errors.

Bill amount calculation is happening like below logic.

Fixed cost: Rs. 350.00

- If the no of units used is less than or equal to 20 ,

  Total cost = Fixed cost + no of units * Rs. 5.00
- If the no of units used is less than or equal to 50 ,

  Total cost = Fixed cost + 20 * Rs. 5.00 + ( no of units – 20 ) * Rs. 5.00
- If the no of units is greater than 50,

  Total cost = Fixed cost + 20* Rs. 5.00 + 30 * Rs. 7.00 + ( no of units – 50 ) * Rs. 10.00

In here initialize three variables to calculate the total bill amount.

- fixed_charge = round (350.00,2)
- unit_cost = 0.00
- units_remaining = noOfUnits

Let’s explain the calculation logic used here for the first 20 units.

When initializing variables, we have initialized “**units_remaining**” value as number of units. Always check if units_remaining are greater than zero. If then from 20 and units remaining value, choose the minimum number. For that minimum value from both two will be used to calculate the cost by multiplying by 5.00. In here **unit_cost** will be considered as changing variable to calculate the total cost from each unit’s categorization. So, for that **unit_cost** will be increased by cost variable value. Also, **units_remaining** value will be decreased by **units** variable value.

This same approach will be used in other categorizations also.

```
# Calculate button
    if st.button("Calculate") and noOfUnits > 0:
        try:
            row = []

            # Fixed charge
            fixed_charge = round(350.00,2)
            unit_cost = 0.00
            units_remaining = noOfUnits

            #For first 20 Units
            if units_remaining > 0:
                units = min(20,units_remaining)
                cost = round(5.00 * units,2)
                row.append(["First 20 units", units, 5.00, cost])
                unit_cost += cost
                units_remaining -= units




            #For second 50 Units
            if units_remaining > 0:
                units = min(50,units_remaining)
                cost = round(7.00 * units,2)
                row.append(["Second 50 units", units, 7.00, cost])
                unit_cost += cost
                units_remaining -= units
            # Remaining units
            if units_remaining > 0:
```

```
        cost = round(units_remaining * 10.00,2)
        row.append(["Remaining units", units_remaining, 10.00, cost])
        unit_cost += cost

    total_cost = fixed_charge + unit_cost
```

After calculation of bill amount, there was added display field to show the result.

```
    # Show result
    st.info(f"Total Charge: Rs. {total_cost:.2f}")
```

Additionally (Optional), can be added a table that displays how the bill amount is calculated.

So, firstly initiate an array called row = []

To show the table, it needs to convert array to data frame.

From each categorization, will append new row to array like below.

```
    row.append(["First 20 units", units, 5.00, cost])
```

To display fixed charge and total charge below rows will also added to row array.

```
    # Add fixed charge
    row.append(["Fixed Charge", "--", "--", fixed_charge])
    # Add total charge
    total_cost = round(unit_cost + fixed_charge,2)
    row.append(["Total Charge", "--", "--", total_cost])
```

As the last step, will be converted array to data frame and display as below.

```
    # Convert to DataFrame and show table
    df = pd.DataFrame(row,columns=["Description","Units(kWh)","Rate (Rs.)",
    "Cost (Rs.)"])
    # Format the Rate and Cost columns
    df["Rate (Rs.)"] = df["Rate (Rs.)"].apply(lambda x: f"{x:.2f}" if
           isinstance(x, (int, float)) else x)
    df["Cost (Rs.)"] = df["Cost (Rs.)"].apply(lambda x: f"{x:.2f}" if
           isinstance(x, (int, float)) else x)
    # Display table
    st.table(df)
```

Also, except action will be used to capture any error in the calculating as below.

```
except Exception as e:
            st.error(f"Error calculating bill: {e}")
```

**Full Code View:**

```
import streamlit as st
import pandas as pd

st.title("Electricity Bill Calculator",width="stretch")

#Initialize Variables
noOfUnits = 0

# Create columns to control layout
col1, col2, col3 = st.columns([3, 1, 1])

with col1:
    # Radio button to choose input type
    status = st.radio("Select an option:", ['Readings', 'No of Units'],horizontal=True)

    # Input logic
    if status == "Readings":
        preReading = st.number_input("Enter last month reading", key="pre",
        min_value=0,step=1, format="%d")
        curReading = st.number_input("Enter this month reading", key="cur",
        min_value=0, step=1, format="%d")

        if preReading and curReading:
           preReading = int(preReading)
           curReading = int(curReading)

           if curReading >= preReading:
               noOfUnits = curReading - preReading
           else:
               st.warning("Current reading must be greater than or equal to
               previous reading.")
     elif status == "No of Units":
        noOfUnits = st.number_input("Enter number of units used", key="units",
              min_value=0, step=1, format="%d")




    # Calculate button
    if st.button("Calculate") and noOfUnits > 0:
        try:
            row = []

            # Fixed charge
            fixed_charge = round(350.00,2)
            unit_cost = 0.00
            units_remaining = noOfUnits

            #For first 20 Units
            if units_remaining > 0:
                units = min(20,units_remaining)
                cost = round(5.00 * units,2)
                row.append(["First 20 units", units, 5.00, cost])
                unit_cost += cost
                units_remaining -= units

            #For second 50 Units
            if units_remaining > 0:
                units = min(50,units_remaining)
                cost = round(7.00 * units,2)
                row.append(["Second 50 units", units, 7.00, cost])
                unit_cost += cost
                units_remaining -= units

            # Remaining units
            if units_remaining > 0:
                cost = round(units_remaining * 10.00,2)
                row.append(["Remaining units", units_remaining, 10.00, cost])
                unit_cost += cost

            total_cost = fixed_charge + unit_cost

            # Add fixed charge
            row.append(["Fixed Charge", "--", "--", fixed_charge])

            total_cost = round(unit_cost + fixed_charge,2)
            row.append(["Total Charge", "--", "--", total_cost])
            # Show result
            st.info(f"Total Charge: Rs. {total_cost:.2f}")

            # Convert to DataFrame and show table
            df = pd.DataFrame(row,columns=["Description","Units(kWh)","Rate (Rs.)",
                 "Cost (Rs.)"])
            # Format the Rate and Cost columns
            df["Rate (Rs.)"] = df["Rate (Rs.)"].apply(lambda x: f"{x:.2f}" if
                               isinstance(x, (int, float)) else x)
            df["Cost (Rs.)"] = df["Cost (Rs.)"].apply(lambda x: f"{x:.2f}" if
                               isinstance(x, (int, float)) else x)
            st.table(df)

        except Exception as e:
            st.error(f"Error calculating bill: {e}")
```

**Create a requirements.txt file**

Creating and maintaining a requirements.txt file is a fundamental best practice for python development. It ensures that your project's dependencies are well-documented and easily reproducible, making it easier for others to work on your code and reducing the likelihood of compatibility issues.

Below command captures all installed dependencies and their versions in the file.

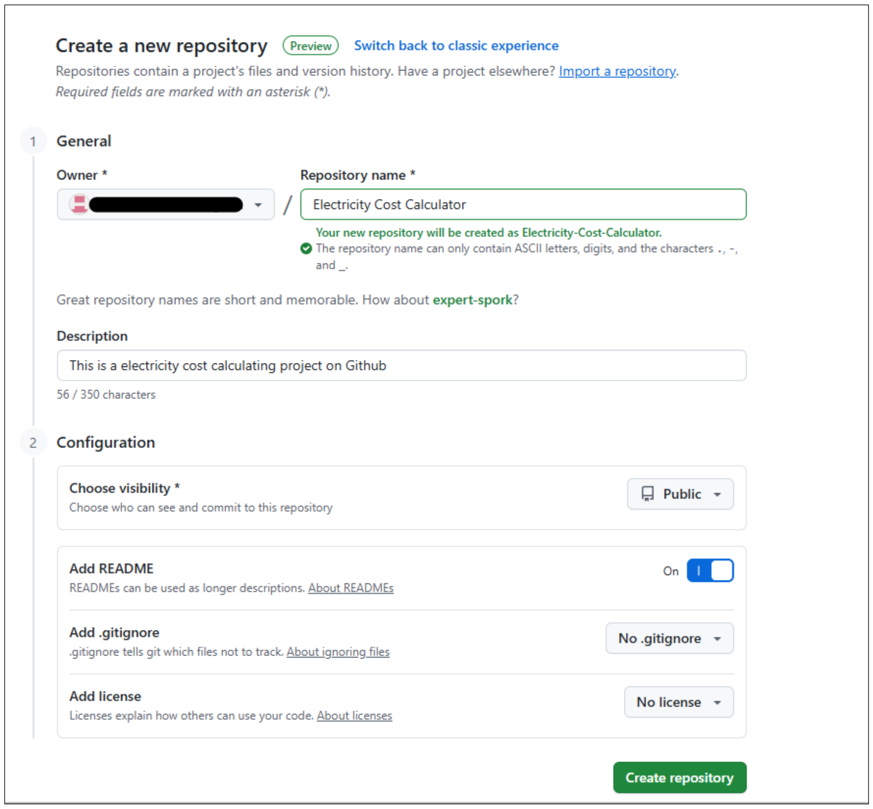
***pip freeze > requirements.txt***

## Deploy the solution

**Pre Requisites:**

- Github Account
- Streamlit Community Cloud

- ✓ **Step 1:** Go to GitHub, log in to your account, and create a new repository.
- ✓ **Step 2:** In the upper-right corner of any page, select, then click new repository.
- ✓ **Step 3:**
  - In the "repository name" box, type a name for your project. For example, type "Electricity Cost Calculator"
  - In the "description" box, type a short description. For example, type "this is electricity cost calculation project on GitHub."
  - Select whether your repository will be public or private. Select "public" if you want others to be able to see your project.
  - Select and add a readme file. You will edit this file in a later step.
  - Click creates repository.



Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? Import a repository.

*Required fields are marked with an asterisk (*).*

1 General

Owner *: [redacted]

Repository name *: Electricity Cost Calculator

Your new repository will be created as Electricity-Cost-Calculator.

The repository name can only contain ASCII letters, digits, and the characters ., -, and _.

Great repository names are short and memorable. How about expert-spork?

Description: This is a electricity cost calculating project on Github

56 / 350 characters

2 Configuration

Choose visibility *: Public

Choose who can see and commit to this repository

Add README: On

READMEs can be used as longer descriptions. About READMEs

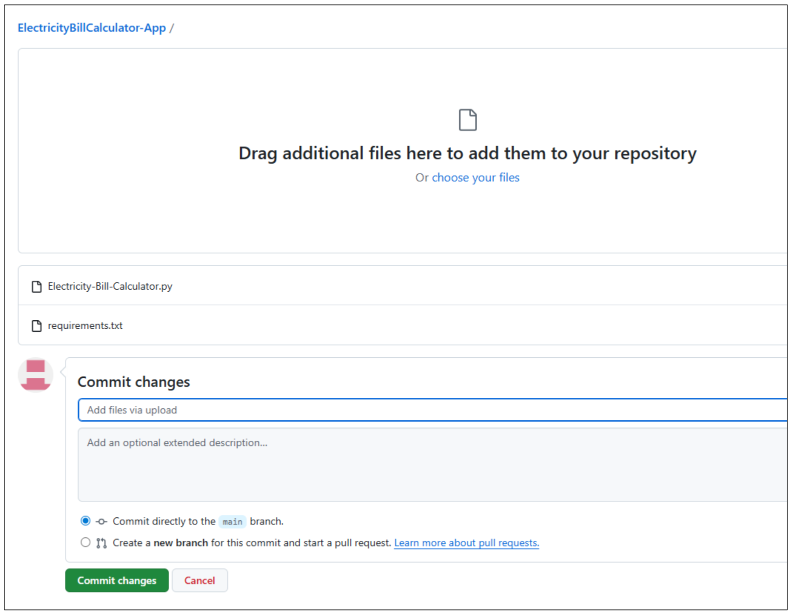
Add .gitignore: No .gitignore

.gitignore tells git which files not to track. About ignoring files

Add license: No license

Licenses explain how others can use your code. About licenses

- ✓ **Step 4:** Upload files to your project's repository
  - To the right of the page, select the add file dropdown menu.
  - From the dropdown menu, click upload files.
  - On your computer, open the folder containing your work, then drag and drop all files and folders into the browser.
  - At the bottom of the page, under "commit changes", select "commit directly to the main branch, then click commit changes.


ElectricityBillCalculator-App /

Drag additional files here to add them to your repository

Or choose your files

Electricity-Bill-Calculator.py

requirements.txt

Commit changes

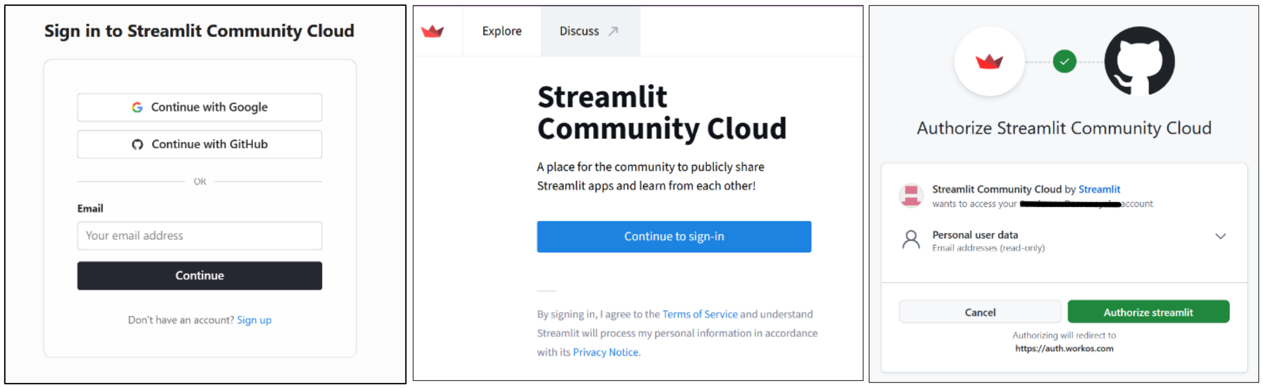
Add files via upload

Add an optional extended description...

Commit directly to the main branch. [✓]

Create a new branch for this commit and start a pull request. Learn more about pull requests.

✓ **Step 5: Connect your GitHub Account-** After you create your Community Cloud account, connect GitHub for source control.


Sign in to Streamlit Community Cloud

Email

Your email address

Don't have an account? Sign up

Streamlit Community Cloud

A place for the community to publicly share Streamlit apps and learn from each other!

By signing in, I agree to the Terms of Service and understand Streamlit will process my personal information in accordance with its Privacy Notice.

Authorize Streamlit Community Cloud

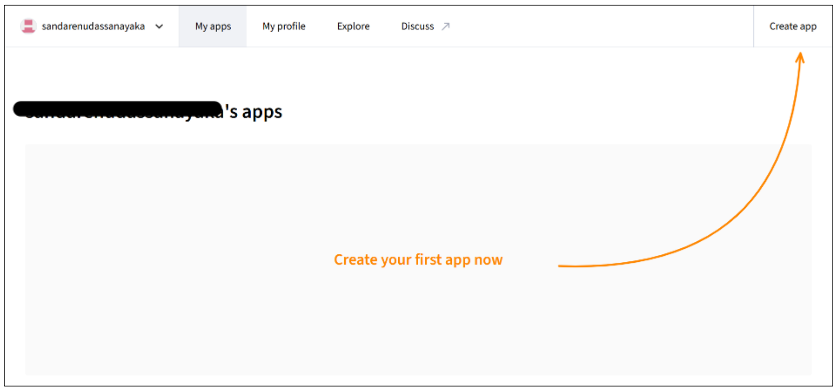
Streamlit Community Cloud by Streamlit wants to access your [redacted] account

Personal user data

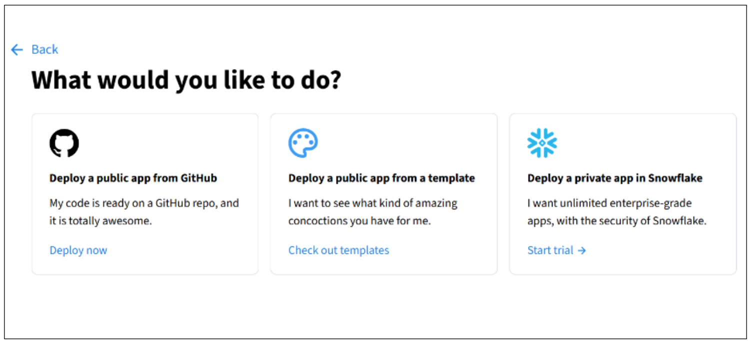
Email addresses (read-only)

Authorizing will redirect to https://auth.workos.com

✓ **Step 6:** From your workspace at share. streamlit.io, in the upper-right corner, click "Create app."


[redacted]'s apps

✓ **Step 7:** From the pop-up window, select “**Deploy app from GitHub**”.



← Back

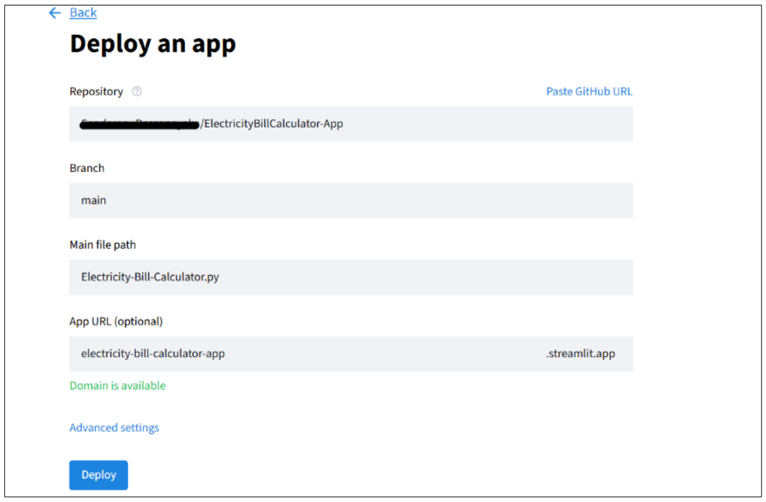
What would you like to do?

**Deploy a public app from GitHub**
My code is ready on a GitHub repo, and it is totally awesome.
Deploy now

**Deploy a public app from a template**
I want to see what kind of amazing concoctions you have for me.
Check out templates

**Deploy a private app in Snowflake**
I want unlimited enterprise-grade apps, with the security of Snowflake.
Start trial →

✓ **Step 8:** As below figure, fill the repository, branch, and file path.



← Back

Deploy an app

| | |
|---|---|
| Repository | [redacted]/ElectricityBillCalculator-App |
| Branch | main |
| Main file path | Electricity-Bill-Calculator.py |
| App URL (optional) | electricity-bill-calculator-app .streamlit.app |

Paste GitHub URL

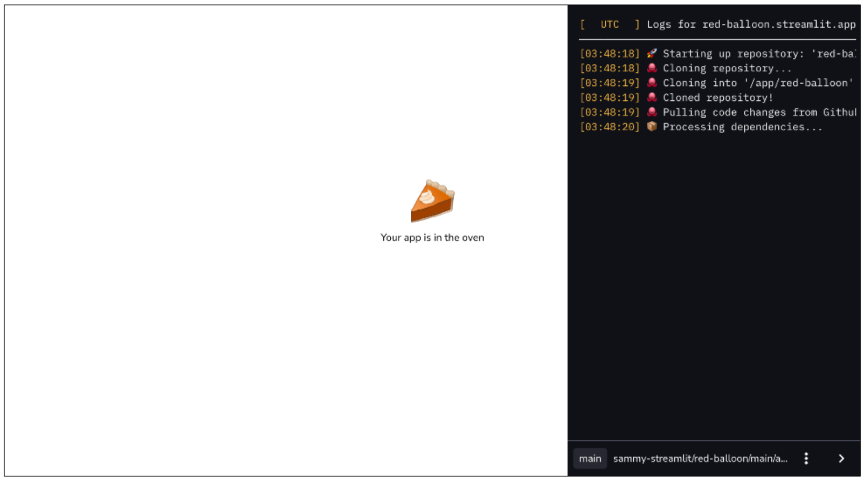
Domain is available

Advanced settings

Deploy

## Watch your app launch

Your app is now being deployed, and you can watch while it launches. Most apps are deployed within a few minutes, but if your app has a lot of dependencies, it may take longer. After the initial deployment, changes to your code should be reflected immediately in your app. Changes to your dependencies will be processed immediately but may take a few minutes to install.



Your app is in the oven

```
[  UTC  ] Logs for red-balloon.streamlit.app
[03:48:18] 🚀 Starting up repository: 'red-ba
[03:48:18] 🐙 Cloning repository...
[03:48:19] 🐙 Cloning into '/app/red-balloon'
[03:48:19] 🐙 Cloned repository!
[03:48:19] 🐙 Pulling code changes from Githul
[03:48:20] 📦 Processing dependencies...
```

main sammy-streamlit/red-balloon/main/a...

## View your app

That's it—you're done! Your app now has a unique URL that you can share with others.
Link : https://electricity-bill-calculator-app.streamlit.app/

**Date :** 2025/08/22

**Written By:** Sandarenu Dassanayaka