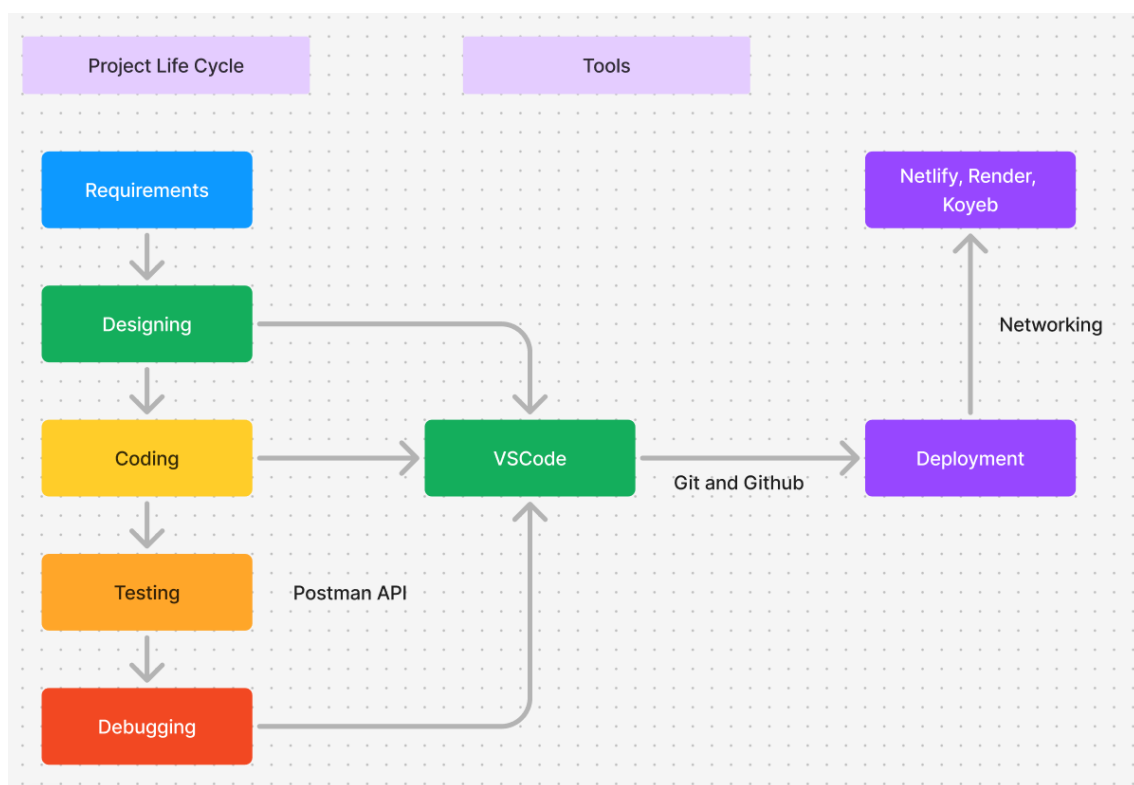


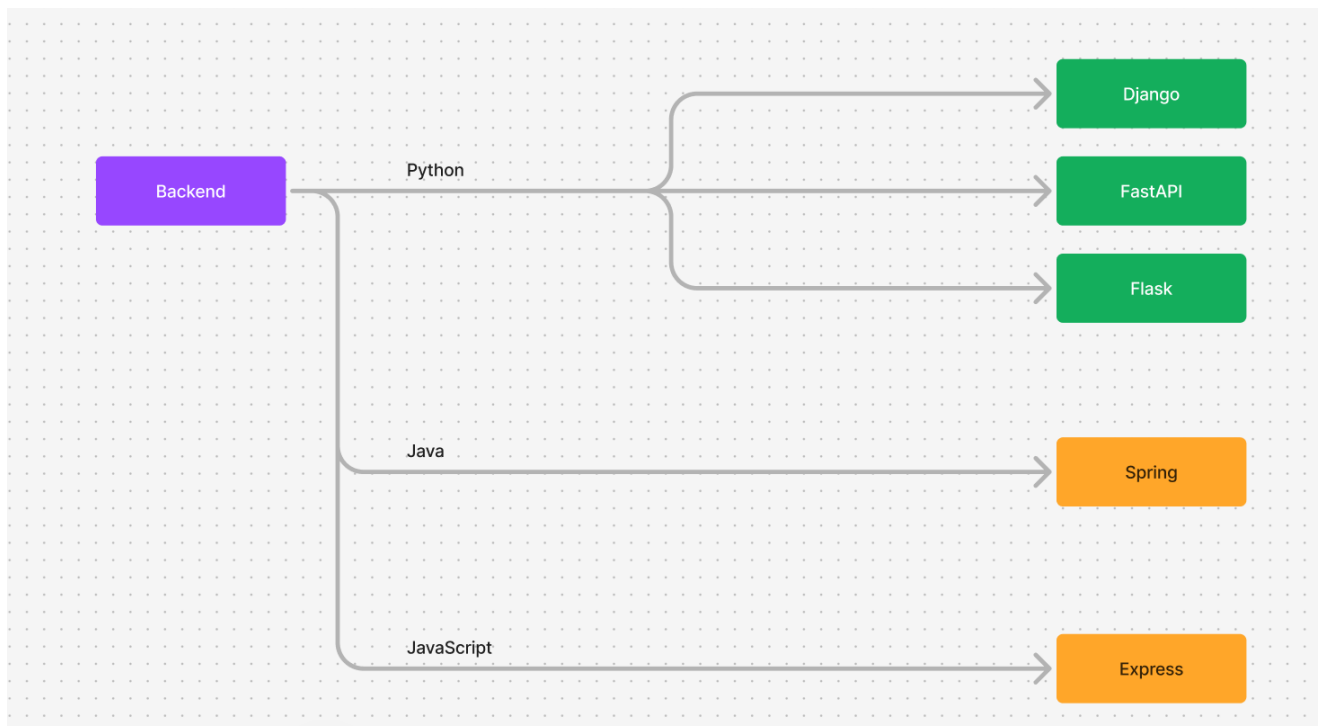
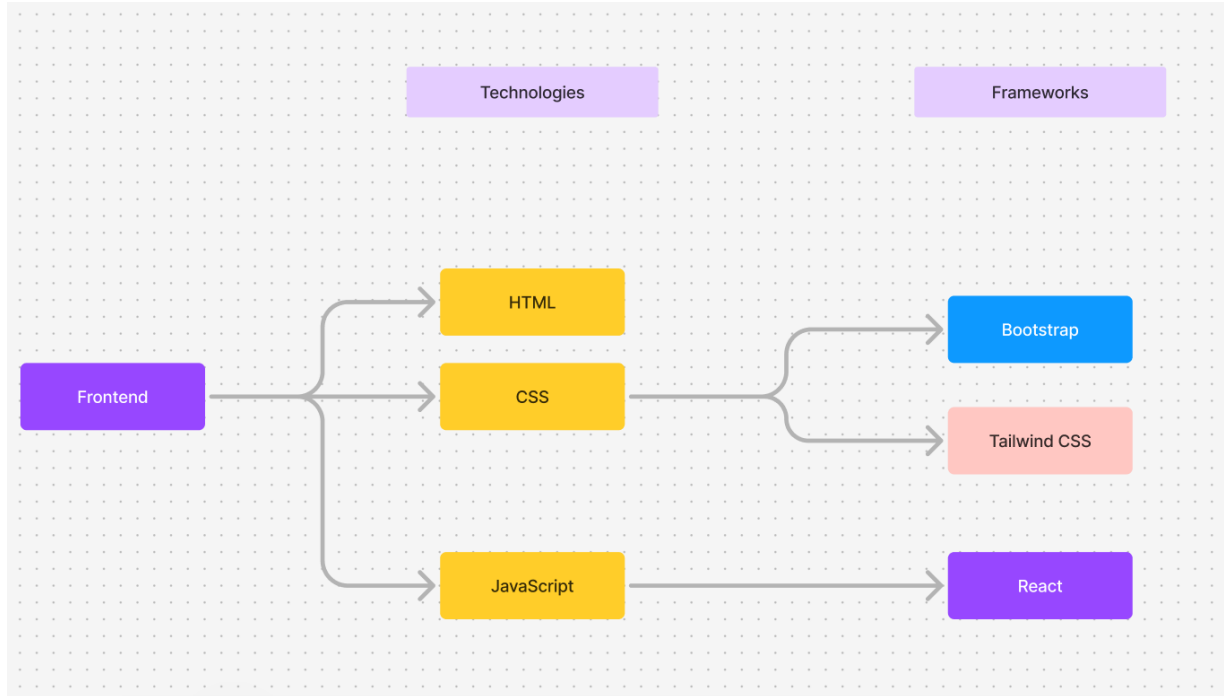
Day 1: Project and Product

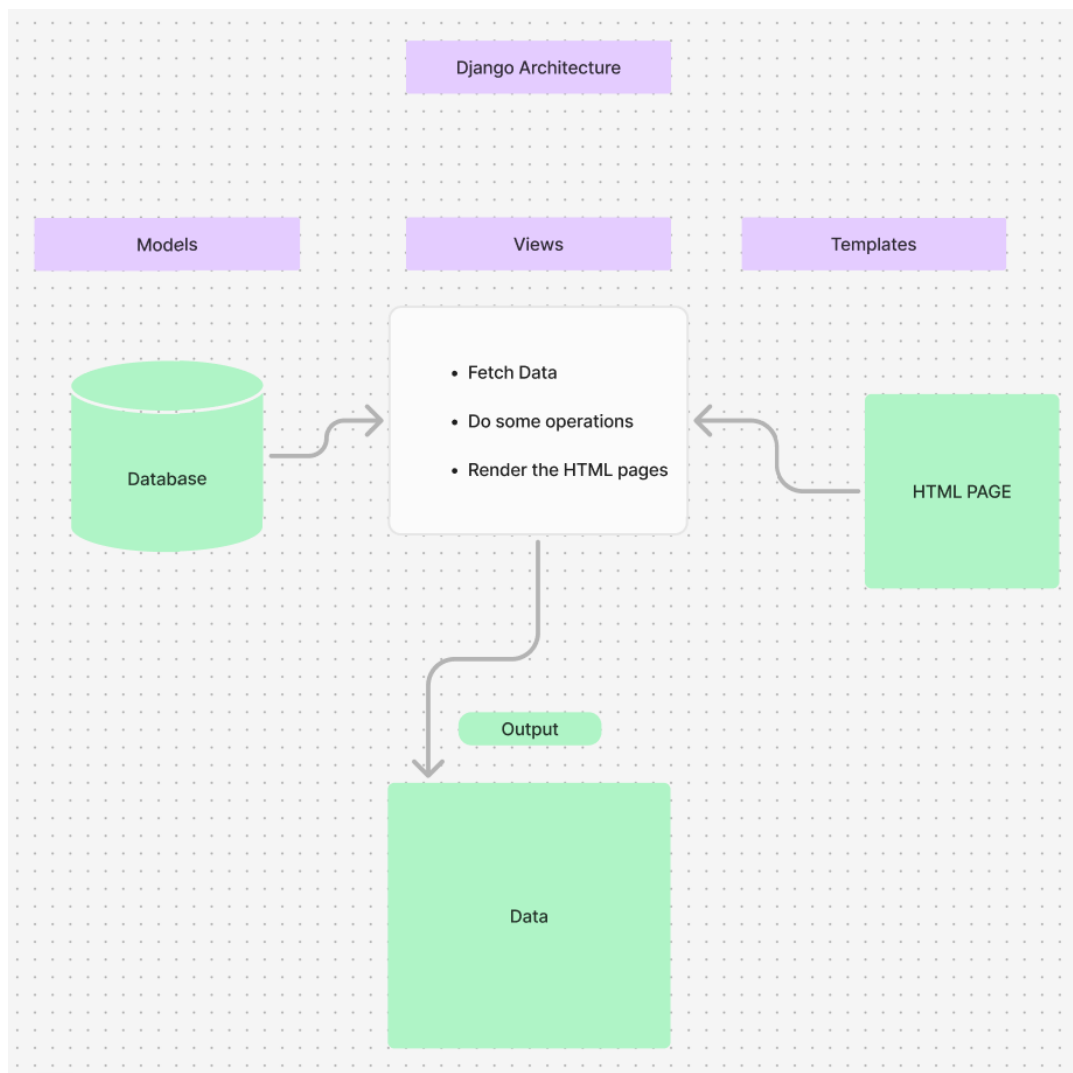
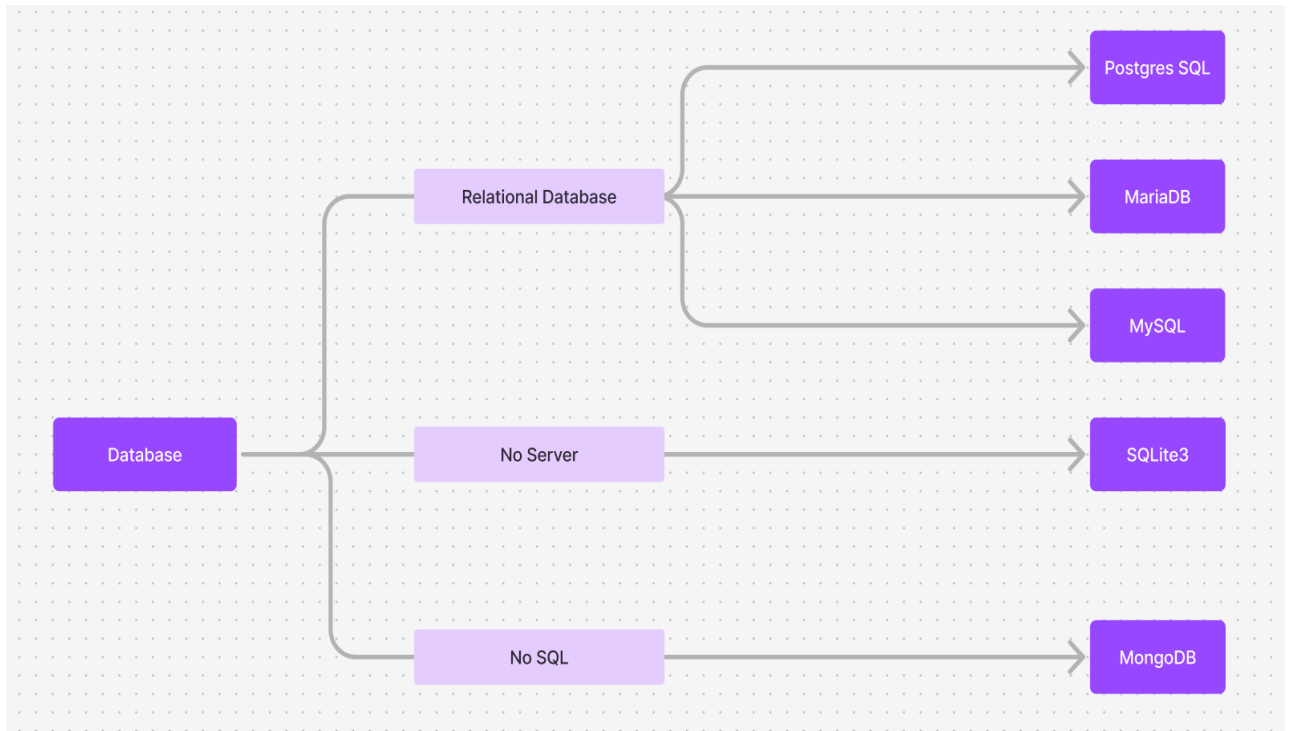
Project is a software under development and will become product once if it is completely developed.

E.g.: Instagram – Product, its clone is a Project

Project Life Cycle







Project 1

Weather App

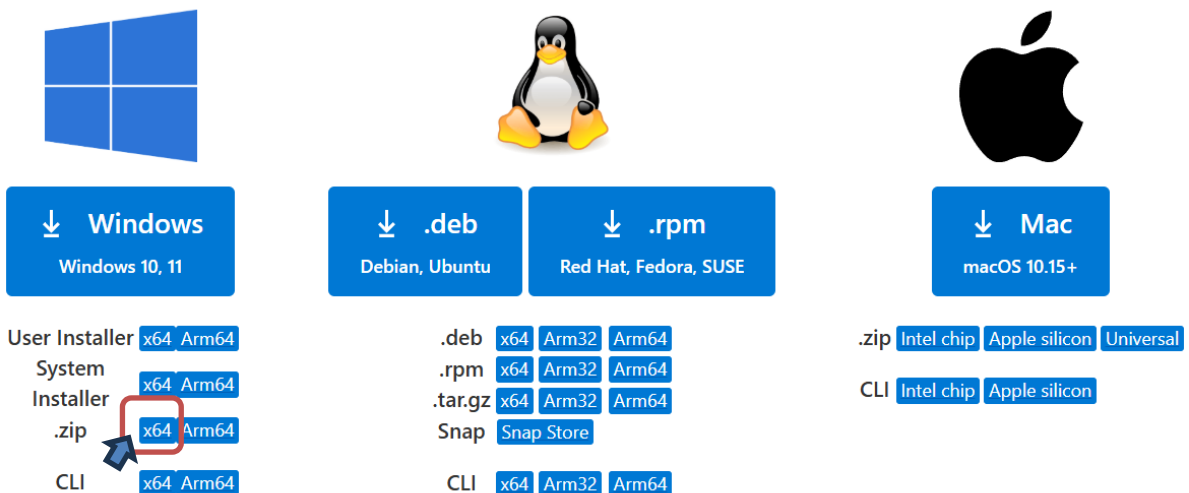
Installation

[Tailwind CLI](#) [Using PostCSS](#) [Framework Guides](#) [Play CDN](#)

The simplest and fastest way to get up and running with Tailwind CSS from scratch is with the Tailwind CLI tool. The CLI is also available as a [standalone executable](#) if you want to use it without installing Node.js.

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

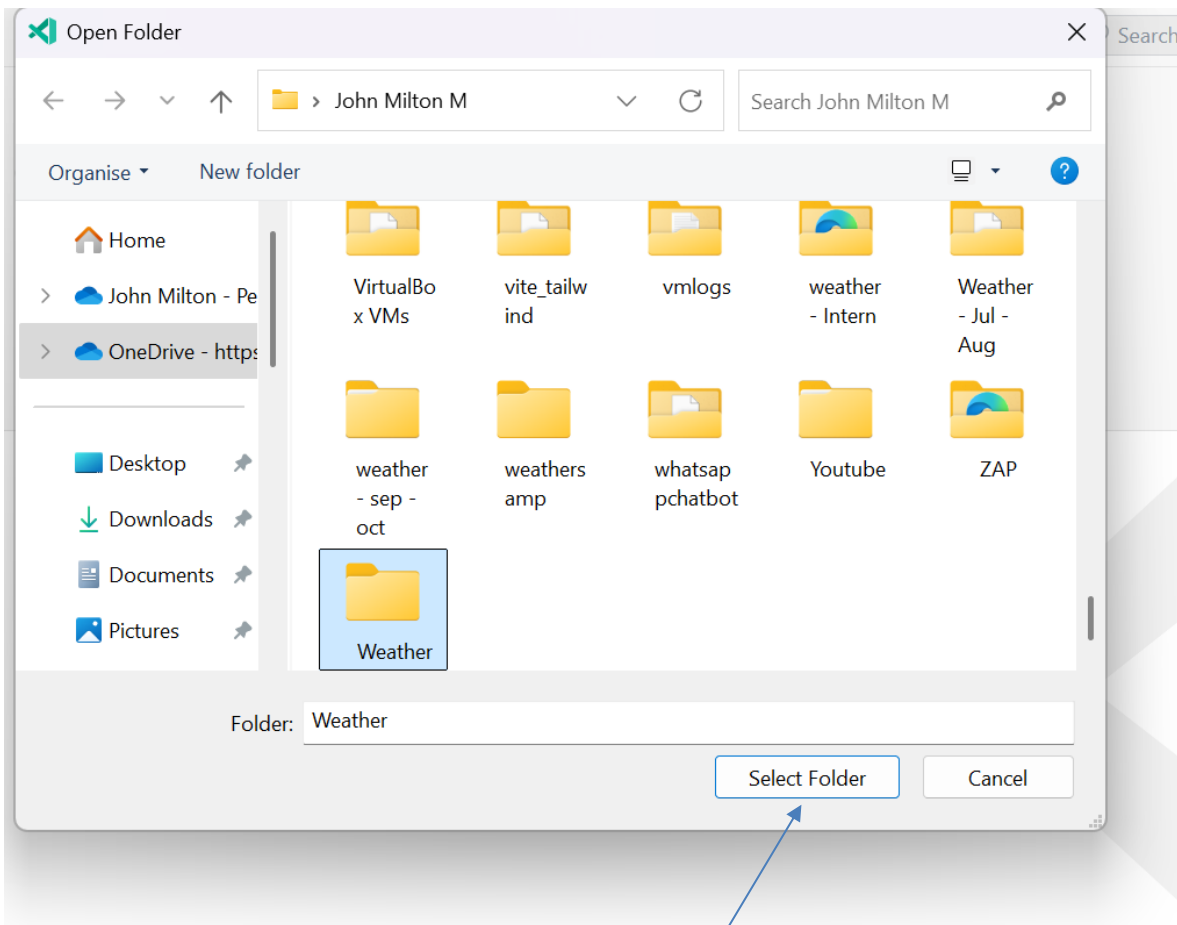


The image shows the Visual Studio Code download page layout. It features three main columns for different operating systems: Windows, Linux, and Mac. Each column has a download button and a list of available installers. The Windows column has a 'Windows' button and lists 'User Installer', 'System Installer', '.zip', and 'CLI'. The Linux column has '.deb' and '.rpm' buttons and lists various package formats. The Mac column has a 'Mac' button and lists '.zip' and 'CLI'. A red box highlights the 'x64' option under the 'User Installer' for Windows, with a blue arrow pointing to it.

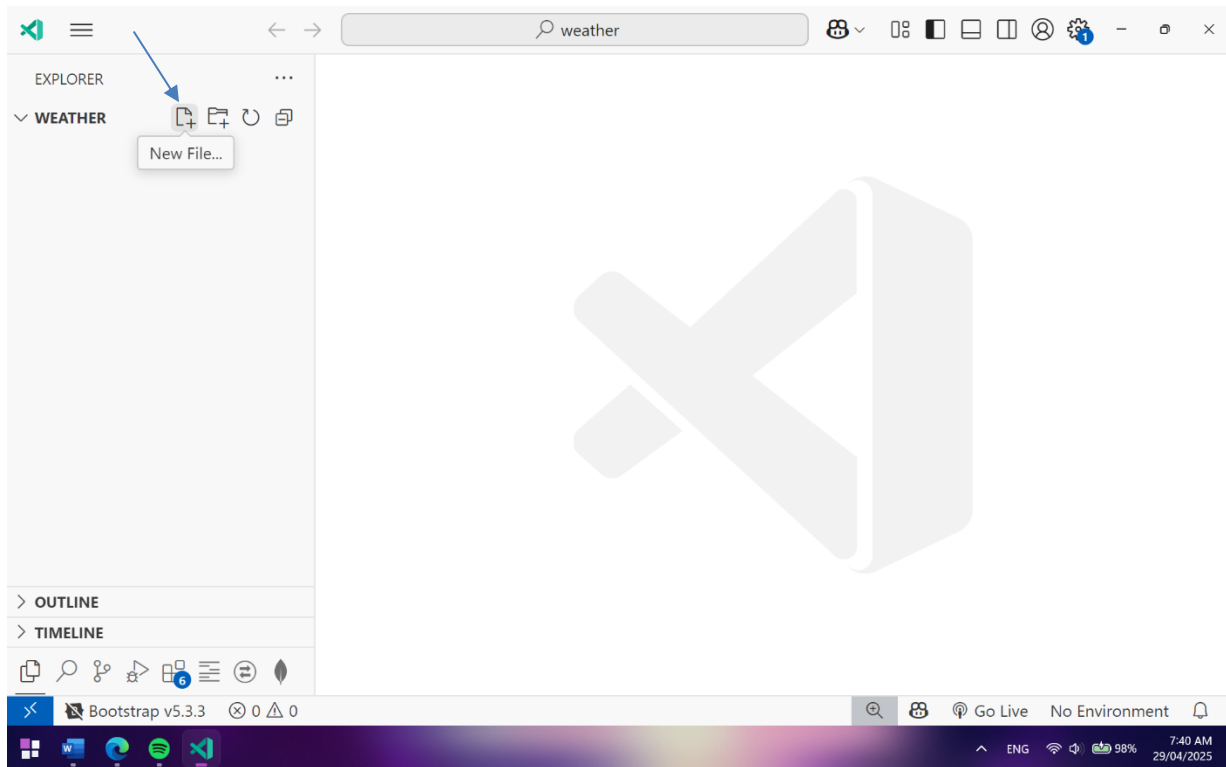
Platform	Download Button	Available Options
Windows	Windows (Windows 10, 11)	User Installer (x64, Arm64), System Installer (x64, Arm64), .zip (x64, Arm64), CLI (x64, Arm64)
Linux	.deb (Debian, Ubuntu), .rpm (Red Hat, Fedora, SUSE)	.deb (x64, Arm32, Arm64), .rpm (x64, Arm32, Arm64), .tar.gz (x64, Arm32, Arm64), Snap (Snap Store), CLI (x64, Arm32, Arm64)
Mac	Mac (macOS 10.15+)	.zip (Intel chip, Apple silicon, Universal), CLI (Intel chip, Apple silicon)

1. Install VsCode
2. Open New window (Ctrl + Shift + N)
3. File -> Open Folder

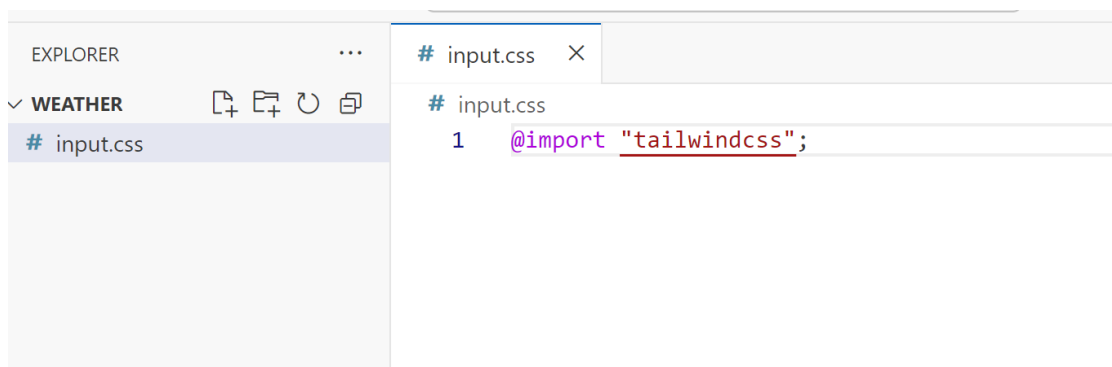




4. Click on New Folder, give name as weather
5. Click Select Folder



6. Click on New File, give name as input.css



```
@import "tailwindcss";
```


7. Make Sure you have installed with [Node.js](#)

Download Node.js®

Get Node.js® v22.15.0 (LTS) for Windows using fnm with npm

```
1 # Download and install fnm:
2 winget install Schniz.fnm
3
4 # Download and install Node.js:
5 fnm install 22
6
7 # Verify the Node.js version:
8 node -v # Should print "v22.15.0".
9
10 # Verify npm version:
11 npm -v # Should print "10.9.2".
```


PowerShell

 Copy to clipboard

"fnm" is a cross-platform Node.js version manager. If you encounter any issues please visit [fnm's website](#)

Or get a prebuilt Node.js® for Windows running a x64 architecture.

 Windows Installer (.msi)

 Standalone Binary (.zip)

8. Go to Vscode and open Integrated Terminal (Ctrl + J)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
o npm install tailwindcss @tailwindcss/cli
```

Type in the terminal and press enter:

```
npm install tailwindcss @tailwindcss/cli
```



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● npm install tailwindcss

up to date, audited 335 packages in 12s

53 packages are looking for funding
  run `npm fund` for details

12 moderate severity vulnerabilities

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
```

9. Run the Tailwindcss Watch mode in background (Necessary to run whenever you are reopening the vscode again)

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

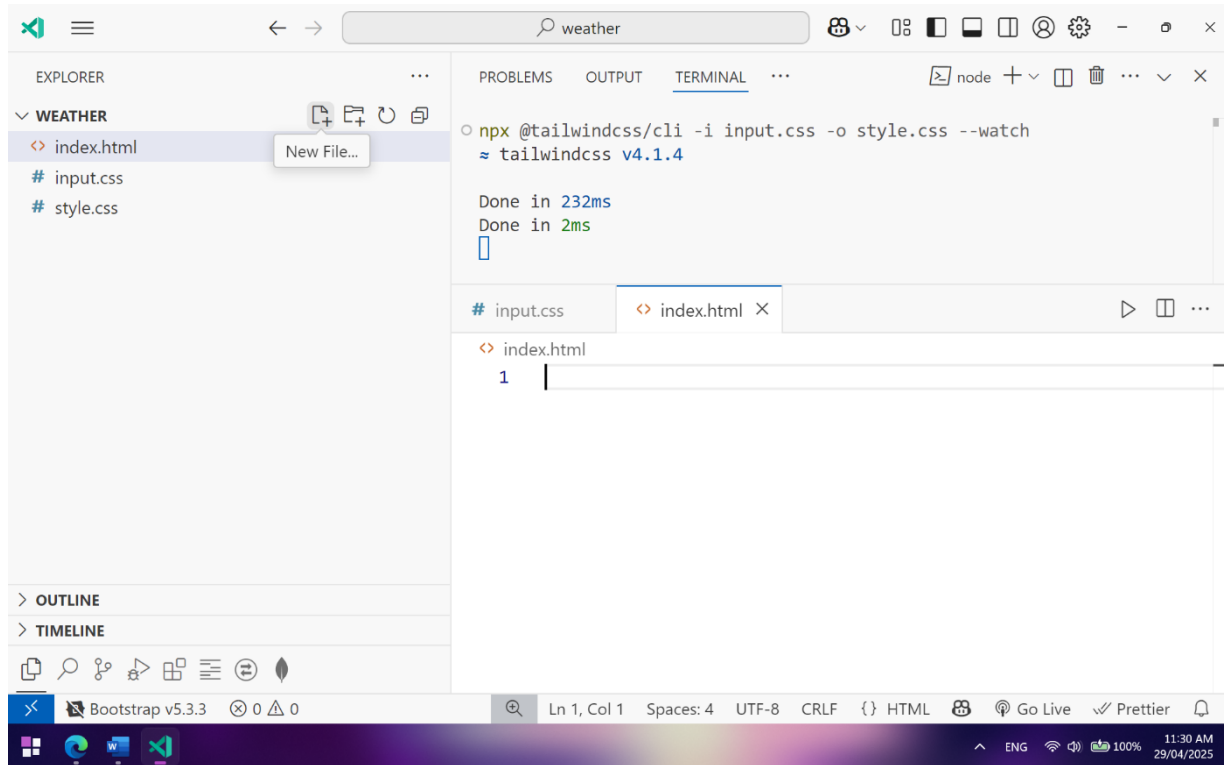
○ npx @tailwindcss/cli -i input.css -o style.css --watch
```

Type in the terminal and press enter:

```
npx @tailwindcss/cli -i input.css -o style.css --watch
```

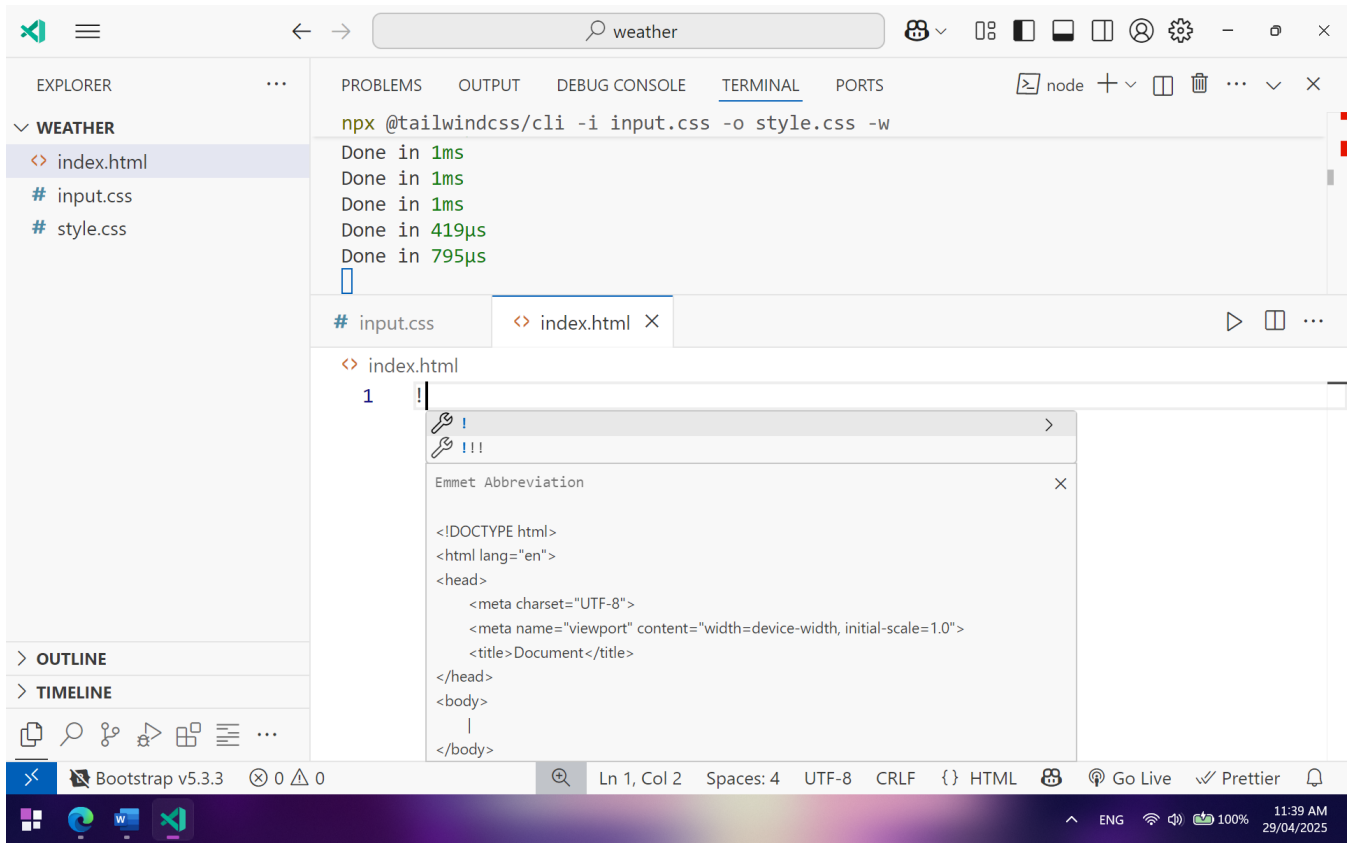
10. Now we need to create index.html and link style.css

Steps to create boiler plate code and linking `style.css` :



Let the `npx tailwindcss -i input.css -o style.css -w` be running in the background and it will write the `style.css` for you by watching the `index.html` if any predefined utility class name is there

1. Press shift key and 1 (shift + 1) to get exclamatory symbol (!) and press enter, this will generate the boiler plate code.
2. Change the title from Document to weather App
3. Inside the head block, type `link:css` and press enter

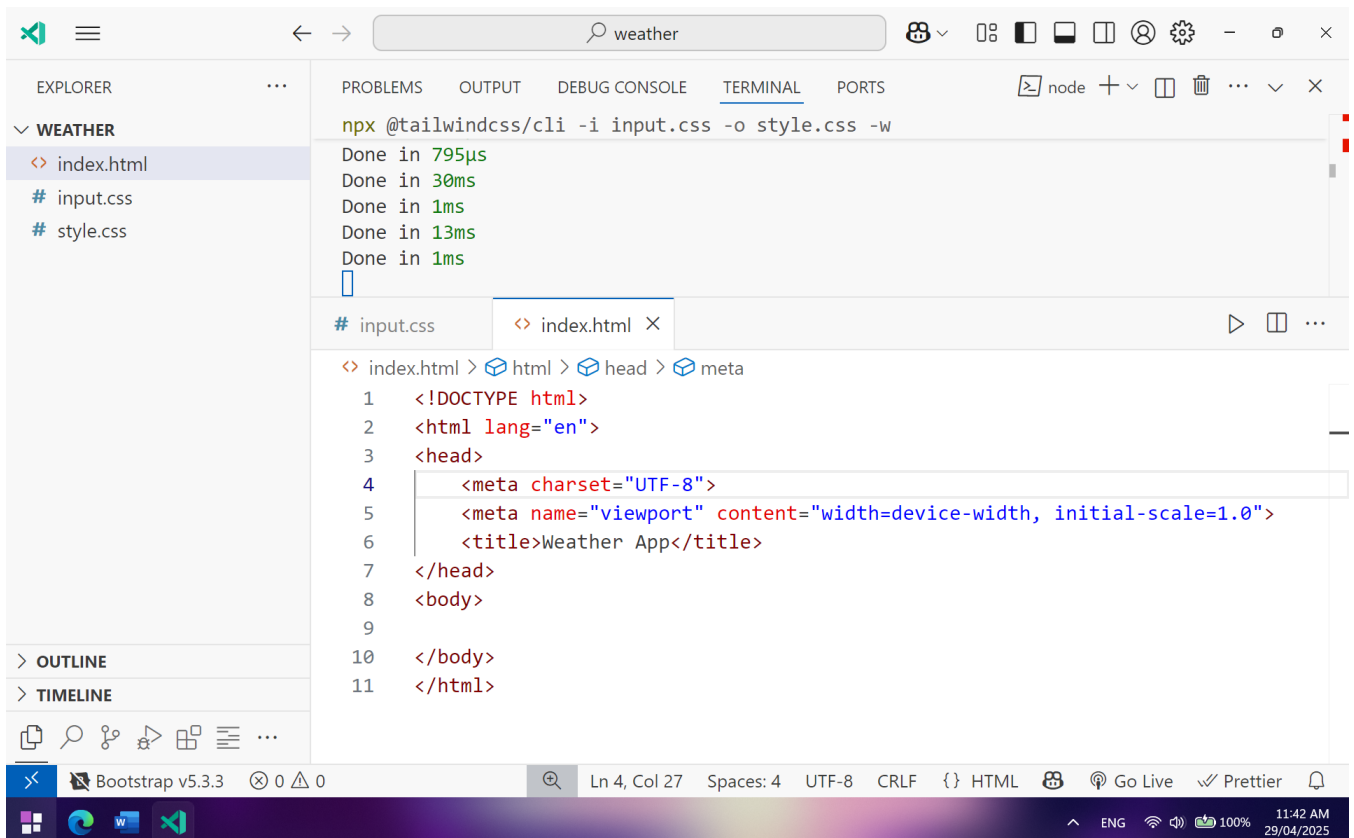


The image shows the Visual Studio Code interface for a project named 'weather'. The Explorer panel on the left shows the file structure: 'index.html', 'input.css', and 'style.css'. The Terminal panel on the right shows the command `npx @tailwindcss/cli -i input.css -o style.css -w` being executed, with output indicating that the files were processed successfully. A snippet of HTML code is visible in the editor, showing the beginning of an HTML document with a DOCTYPE declaration and a head section containing meta tags for charset and viewport.

```
index.html
# input.css
# style.css
```

```
npx @tailwindcss/cli -i input.css -o style.css -w
Done in 1ms
Done in 1ms
Done in 1ms
Done in 419µs
Done in 795µs
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  |
</body>
```



The image shows the Visual Studio Code interface for the 'weather' project. The Explorer panel on the left shows the file structure: 'index.html', 'input.css', and 'style.css'. The Terminal panel on the right shows the command `npx @tailwindcss/cli -i input.css -o style.css -w` being executed, with output indicating that the files were processed successfully. The editor shows the final HTML code, which includes the DOCTYPE declaration, the head section with meta tags for charset and viewport, and the body section.

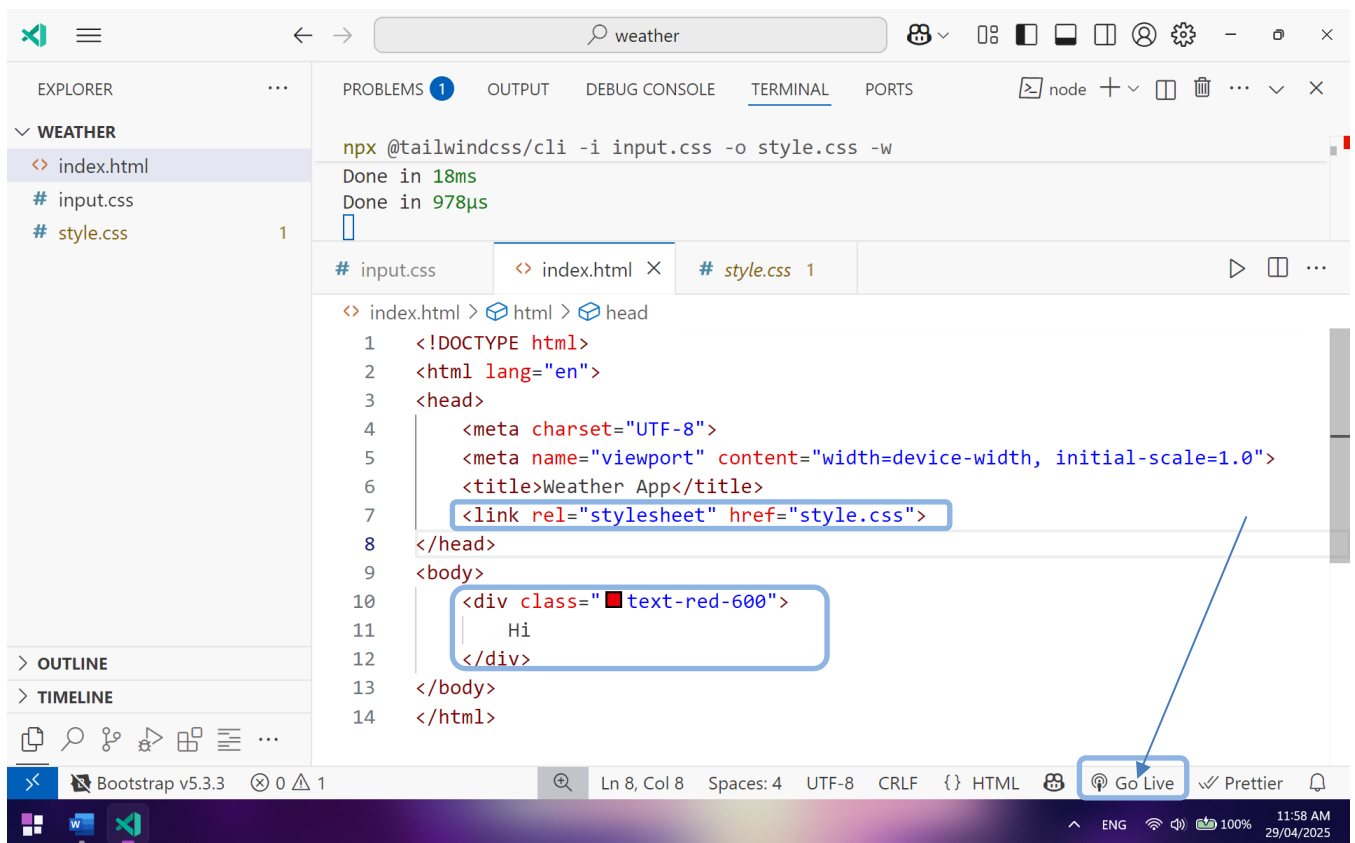
```
index.html
# input.css
# style.css
```

```
npx @tailwindcss/cli -i input.css -o style.css -w
Done in 795µs
Done in 30ms
Done in 1ms
Done in 13ms
Done in 1ms
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Weather App</title>
</head>
<body>
</body>
</html>
```

Now for testing the tailwindcss, inside the `body` block, we need to create a `div` block with the class name `text-red-600` with some content

1. Inside `body` block type `.text-red-600` and press enter
2. Now it has been created one `div` block with class name `text-red-600`
3. Give Content `Hi` and turn on the live server (Need to install Extension)



```
npm run dev
npx @tailwindcss/cli -i input.css -o style.css -w
Done in 18ms
Done in 978µs


# input.css  < index.html  < # style.css 1

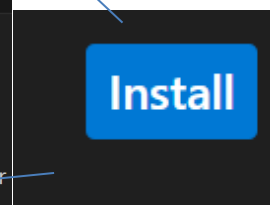
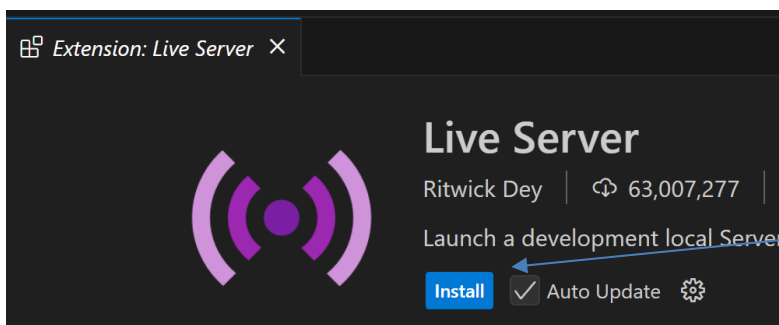
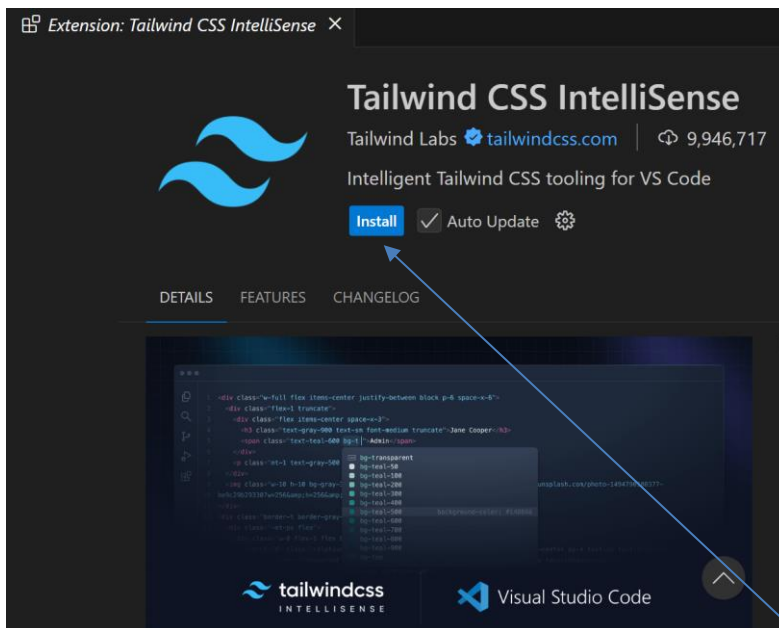
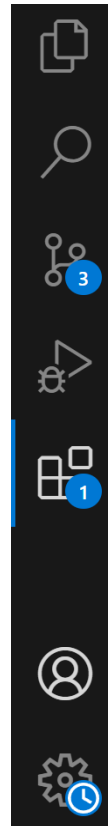
< index.html > html > head
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Weather App</title>
7    <link rel="stylesheet" href="style.css">
8  </head>
9  <body>
10   <div class="text-red-600">
11     Hi
12   </div>
13 </body>
14 </html>
```

4. Click on Go Live

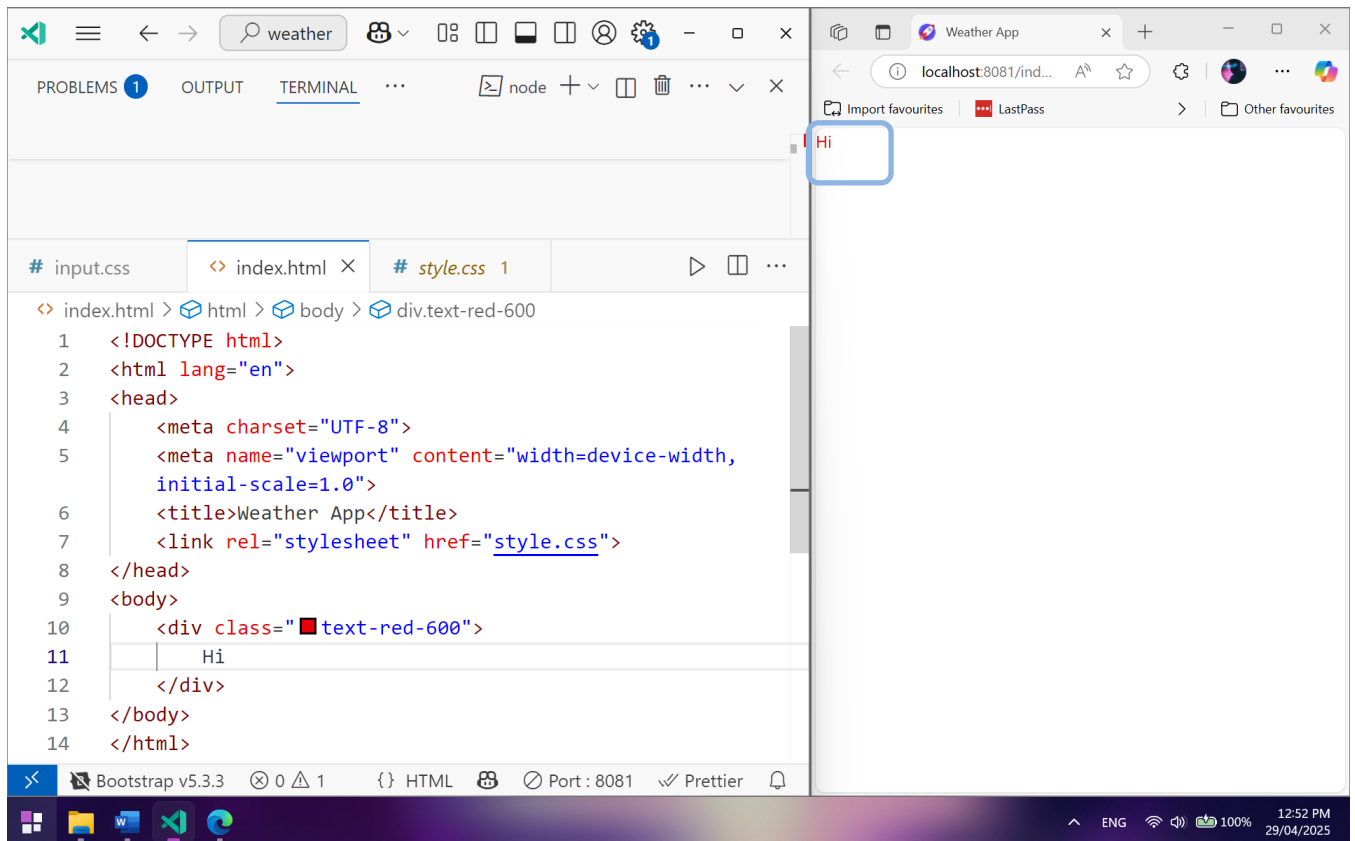
For color and suggestions, we need to install `tailwindcss` `intellisense` extension.

Extension Installation Guide:

1. Click on Extension Market Place  (It will be in this row ->)
2. Search for
 - Tailwindcss intellisense
 - Live server
3. Click on install.



Output:



If you can able to see the text content in red color, Voila, you have successfully configured frontend for you project !!!

Day 2: Weather App – Frontend Part -1, 2, 3

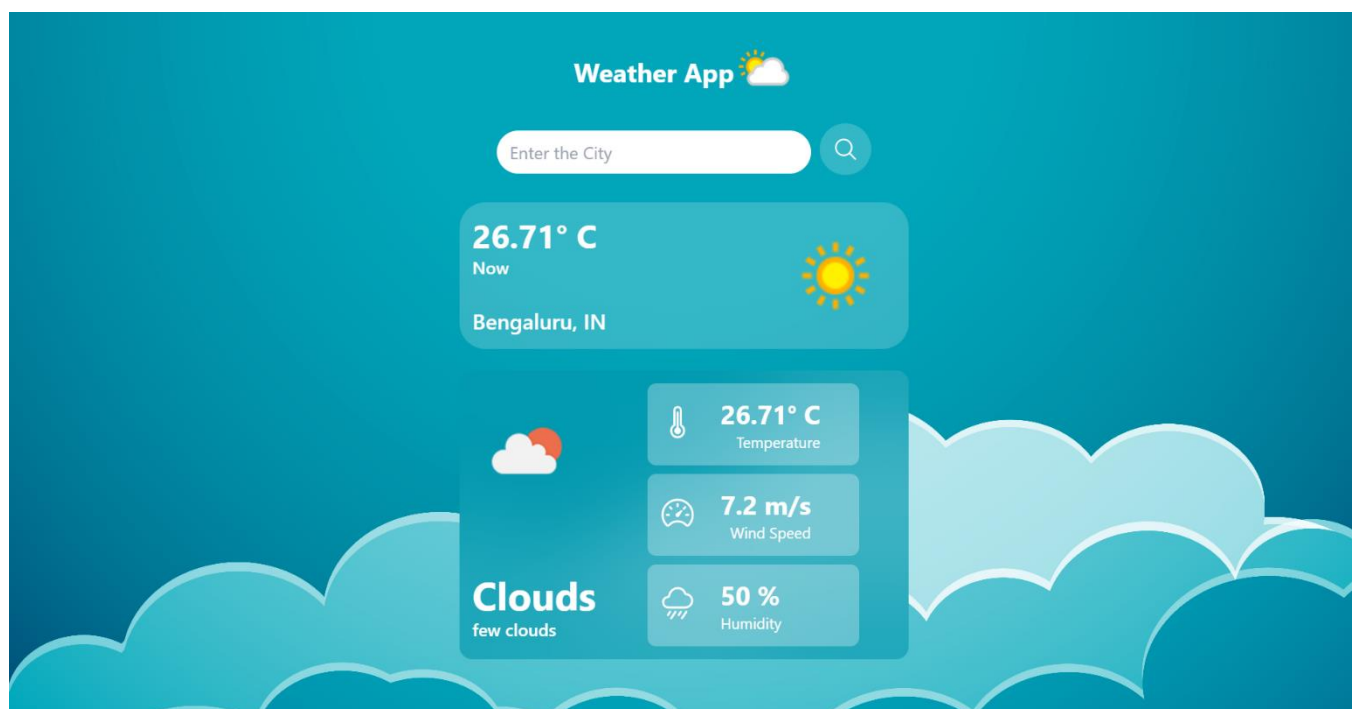
Now, we just need to Design the Frontend part.

Here we are going to learn CSS flex model easily with the help of CSS Framework (TailwindCSS)

As per our project life cycle, initially we need to gather the requirements.

Requirement:

We want to design the page like this,



Now, let's divide the design part by part. And we have 4 blocks.

Part 1



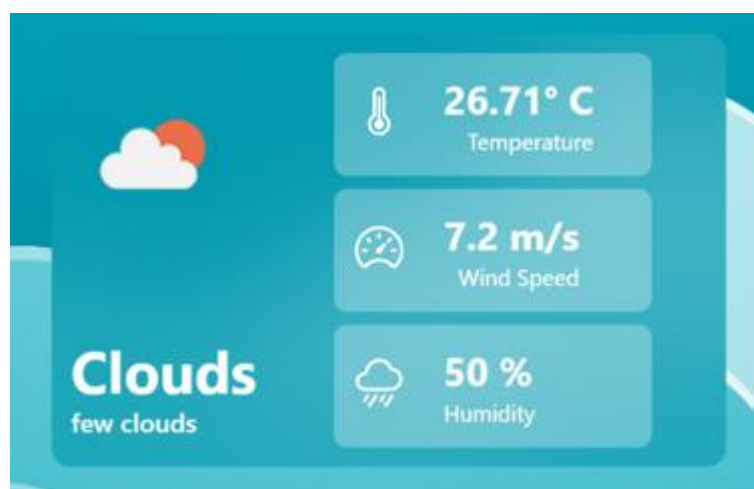
Part 2



Part 3



Part 4



Initially we need to apply background, wait I'll share all the required images
In compressed format. [Click here to Download](#). Extract into images folder

Applying background:

-----Step 1-----

```
<body class="bg-[url('images/bg_image.jpg')]">  
  
</body>
```

We need to cover the background(**bg-cover**), avoid repetition(**bg-no-repeat**),
height should be screen height (**min-h-screen**)

-----Step 2-----

```
<body class="bg-[url('images/bg_image.jpg')] bg-cover bg-no-repeat min-h-screen">  
  
</body>
```

We have 4 blocks inside the body and we need to give gap(**gap-5**) between each
and every block and it is in column direction (**flex-col**), space around content
(**p-10**)

-----Step 3-----

```
<body class="bg-[url('images/bg_image.jpg')] bg-cover bg-no-repeat min-h-screen items-  
center flex flex-col gap-5 p-10">  
  
</body>
```

Part 1



```
<div class="flex items-center w-fit">

  <h1 class="text-white text-2xl font-bold">Weather App</h1>
  

</div>
```

Part 2



```
<form action="" method="get" class="flex gap-2 justify-center items-center w-4/5 md:w-3/5 lg:w-[70%" ">

  <input type="text" name="city" class="rounded-full py-2 px-3 focus:bg-white/20
outline-white/70 bg-white focus:placeholder:text-white text-white w-4/5 md:w-3/5 lg:w-
[35%" " placeholder="Enter the City">

  <button class="search" type="submit">
    <ion-icon class="text-white text-2xl bg-white/20 rounded-full p-3"
name="search-outline"></ion-icon>
  </button>
</form>
```

Part 3

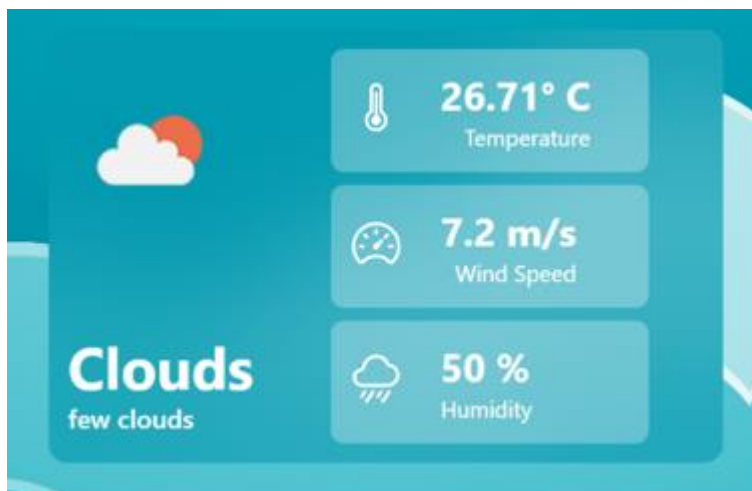


```
<div class="flex bg-white/20 p-3 rounded-3xl backdrop-blur-sm text-white w-4/5 md:w-3/5
lg:w-[35%] cursor-pointer hover:scale-[105%] ease-in-out duration-500 justify-between">

<div id='left' class="flex flex-col">
  <div id="top" class="mb-6 ">
    <p class="font-bold text-3xl">26.71&deg; C</p>
    <p class="font-semibold">Now</p>
  </div>
  <div id="bottom" class="font-semibold text-xl">
    {{name}}, {{country}}
  </div>
</div>
<div id="right" class="justify-center items-center">
  
</div>
</div>
```

Day 3: Weather App – Frontend Part -4

Part 4



```
<div class="flex p-3 bg-[#0198afa9] rounded-xl backdrop-blur-2xl md:w-3/5 lg:w-[35%]
w-[80%] gap-12" >

  <div id="left" class="text-white">
    <div class="top mb-16 mt-3 ">
      
    </div>
    <div id="bottom">
      <p class="flex font-bold text-4xl"> Clouds </p>
      <p class="font-semibold">few clouds</p></div></div>
```

```

<div class="flex w-1/2 flex-col space-y-2">

<div id="right" class="flex p-3 bg-white/30 rounded-lg gap-6">
  <div id="left"><ion-icon class="text-white w-8 h-8 mt-2"
name="thermometer-outline"></ion-icon></div>

  <div id="right" class="text-white">

    <div class="font-bold text-2xl">26.71&deg; C</div>
    <div class="text-sm text-right">Temperature</div>

  </div>

</div>

<div id="right" class="flex p-3 bg-white/30 rounded-lg gap-6">
  <div id="left"><ion-icon class="text-white w-8 h-8 mt-2"
name="speedometer-outline"></ion-icon></div>

  <div id="right" class="text-white">

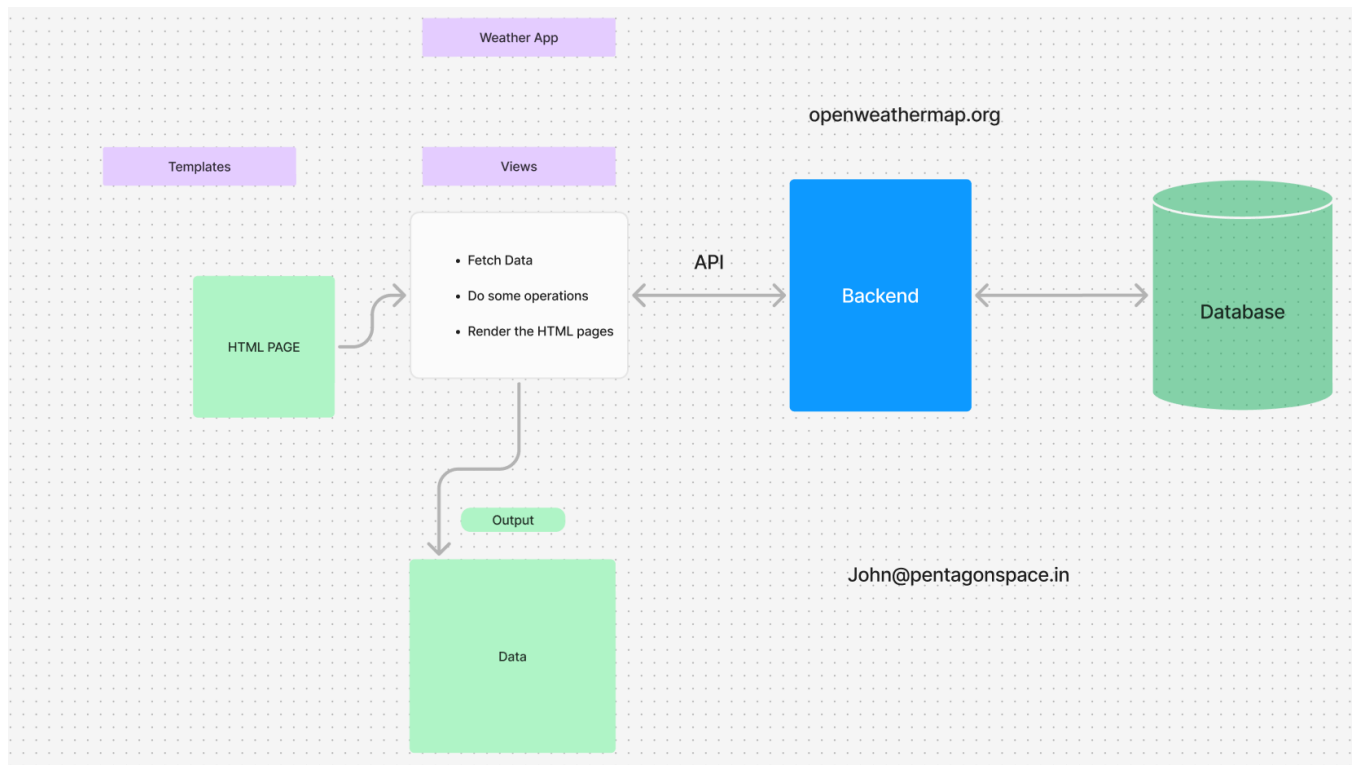
    <div class="font-bold text-2xl">7.2 m/s</div>
    <div class="text-sm text-right">Wind Speed</div>
  </div>
</div>
<div id="right" class="flex p-3 bg-white/30 rounded-lg gap-6">
  <div id="left"><ion-icon class="text-white w-8 h-8 mt-2" name="rainy-
outline"></ion-icon></div>

  <div id="right" class="text-white">

    <div class="font-bold text-2xl">50 %</div>
    <div class="text-sm text-right">Humidity</div> </div>
  </div></div></div>

```

Day 4: Weather App – API + Backend



```
from django.shortcuts import render
import requests

def home(request):
    city = request.GET.get('city', 'bangalore')
    api_key = 'b87770e0c526050cc7426cd23fa9cf28'
    url =
f'https://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}&units=metric
'

    print(url)

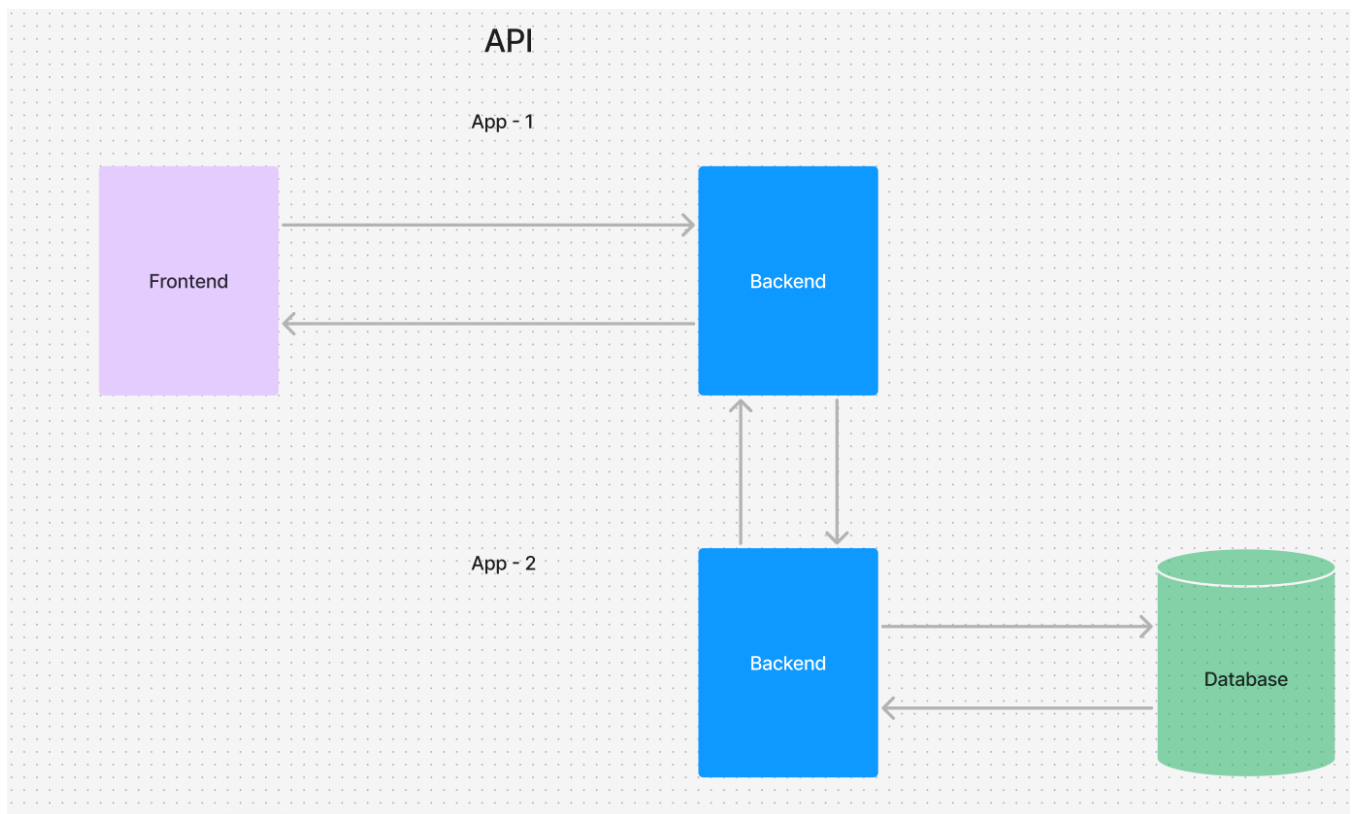
    api = requests.get(url).json()

    temperature = api['main']['temp']
    wind_speed = api['wind']['speed']
    humidity = api['main']['humidity']
    name = api['name']
    country = api['sys']['country']
    icon = api['weather'][0]['icon']
    weather = api["weather"][0]['main']
    description = api['weather'][0]['description']
```

```
img_url = f"https://openweathermap.org/img/wn/{icon}@2x.png"

#
https://api.openweathermap.org/data/2.5/weather?q=bangalore&appid=b87770e0c526050cc7426
cd23fa9cf28&units=metric

return render(request, 'index.html', {'temperature':temperature,
'wind_speed':wind_speed, 'humidity':humidity, 'name':name,
'country':country, 'img_url':img_url, 'weather':weather, 'description':description})
```



Note:

You have to install module `requests` in order to handle API !

```
pip install requests
```

Hint: How to Setup Backend?

Step 1: Ensure that you have installed with python (Install the latest stable version 3.11.9) and check in command prompt

```
py -V
```

(or)

```
python --version
```

Step 2: Install Django (if first command not working try others)

```
pip install django
```

(or)

```
py -m pip install django
```

(or)

```
python -m pip install django
```

Step 3: Create Django project

```
django-admin startproject project
```

(or)

```
py -m django startproject project
```

Rename the project folder to backend and get into that (change directory)

```
cd backend
```

Create app,

```
py manage.py startapp app
```

Create templates, static folder (only inside the app folder so that no need of extra configuration in settings.py)

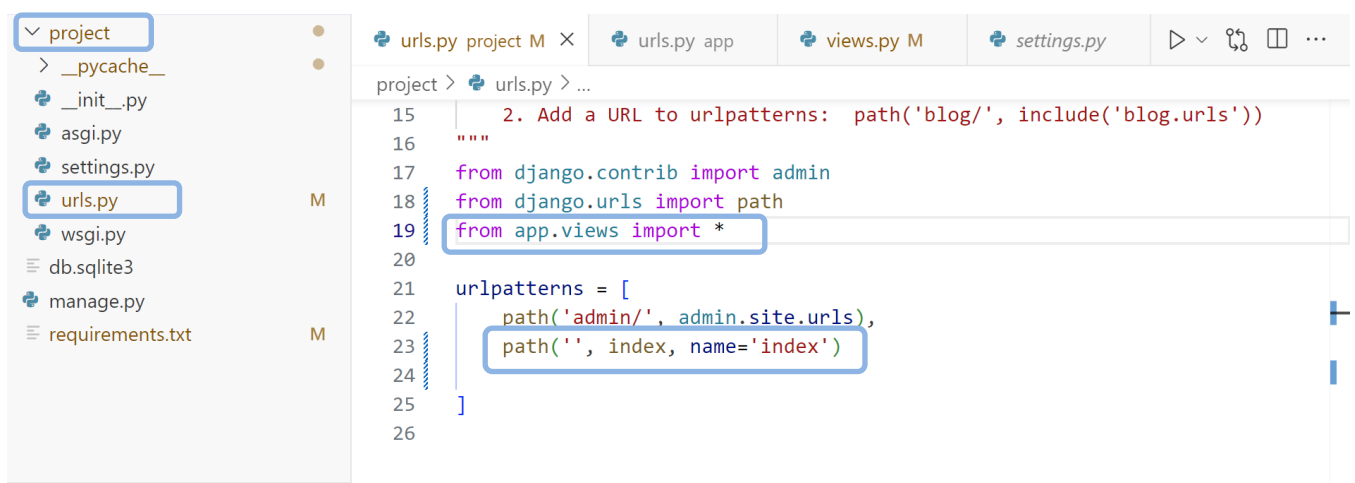
Day 5: Weather App – Frontend + Backend

This is the crucial day where you will learn how to connect frontend and the backend.

Step 1: Move the `index.html` to `templates` folder and move the `images` folder, `style.css` into `static` folder.

Step 2: Configure the `urls.py` of project folder (refer the screenshot below)

Step 3: Create one view - `index` function (refer API + Backend) inside the `views.py` (inside the app folder)

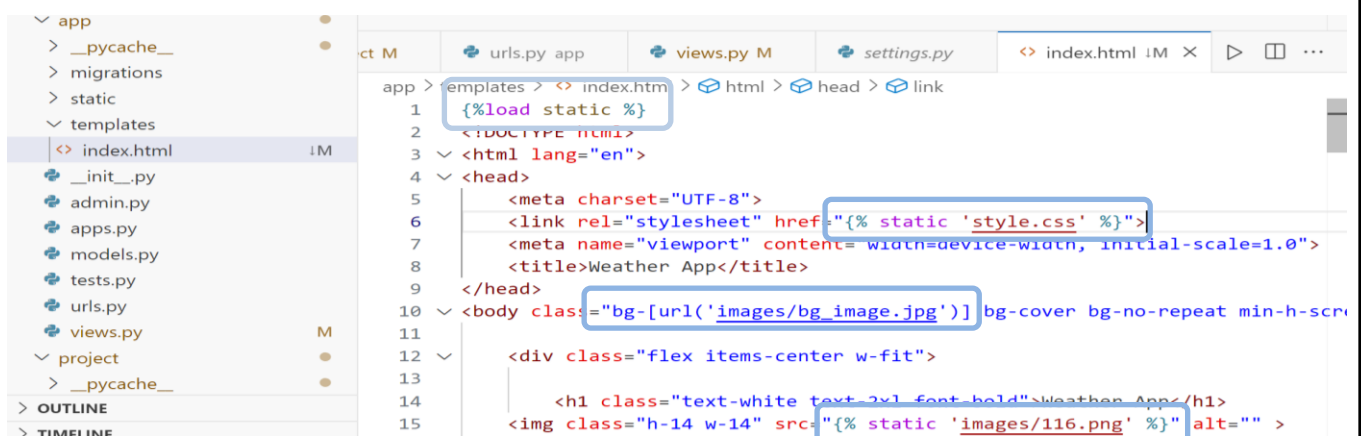


The screenshot shows the Django project's `urls.py` file. The left sidebar shows the project structure with `urls.py` selected. The main editor shows the following code:

```

15 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17 from django.contrib import admin
18 from django.urls import path
19 from app.views import *
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('', index, name='index')
24 ]
25
26
  
```

Step 4: Load the `static` inside the `index.html` and link the `style.css` all the images (except background image) using static url



The screenshot shows the Django app's `templates/index.html` file. The left sidebar shows the app structure with `index.html` selected. The main editor shows the following code:

```

1 {%load static %}
2 <!DOCTYPE html>
3 <html lang="en">
4 <head>
5     <meta charset="UTF-8">
6     <link rel="stylesheet" href="{% static 'style.css' %}">
7     <meta name="viewport" content="width=device-width, initial-scale=1.0">
8     <title>Weather App</title>
9 </head>
10 <body class="{% static 'images/bg_image.jpg' %}" bg-cover bg-no-repeat min-h-screen">
11
12     <div class="flex items-center w-fit">
13
14         <h1 class="text-white text-2xl font-bold">Weather App</h1>
15         
  
```

Step 5: Finally run the server and follow the URL <http://127.0.0.1:8000/>

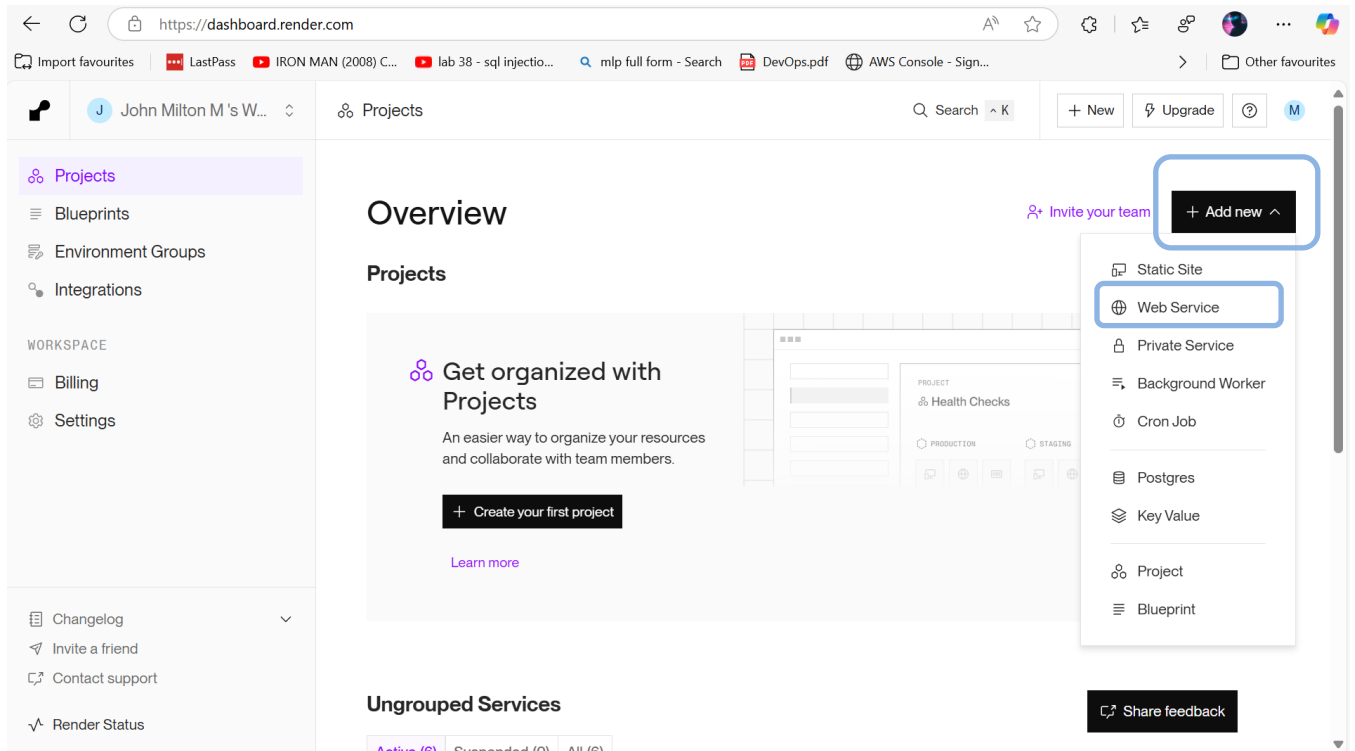
```
py manage.py runserver
```


Day 6: Deployment (Production Server)

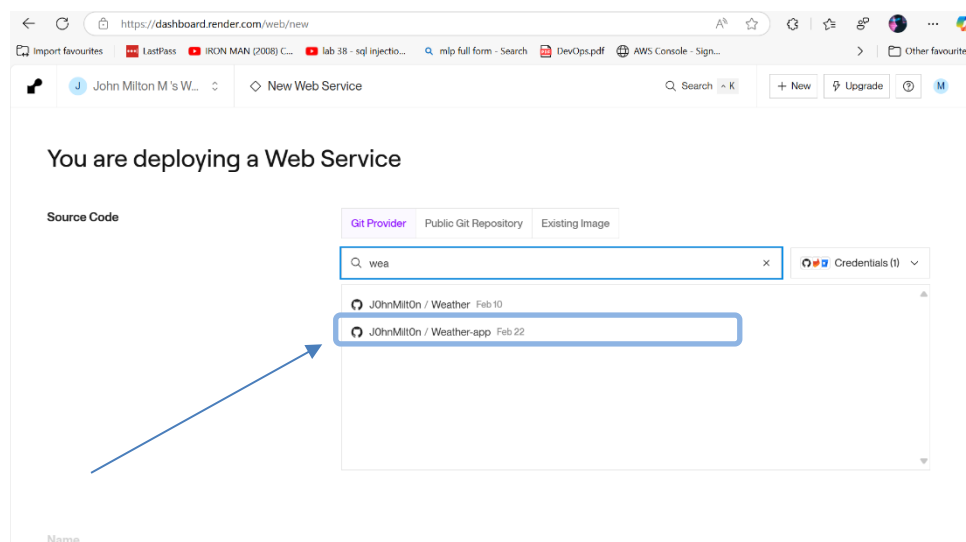
This is the last and foremost step involved in hosting your own public website

Step 1: Login to render.com and go to dashboard.

Step 2: Create new web service.

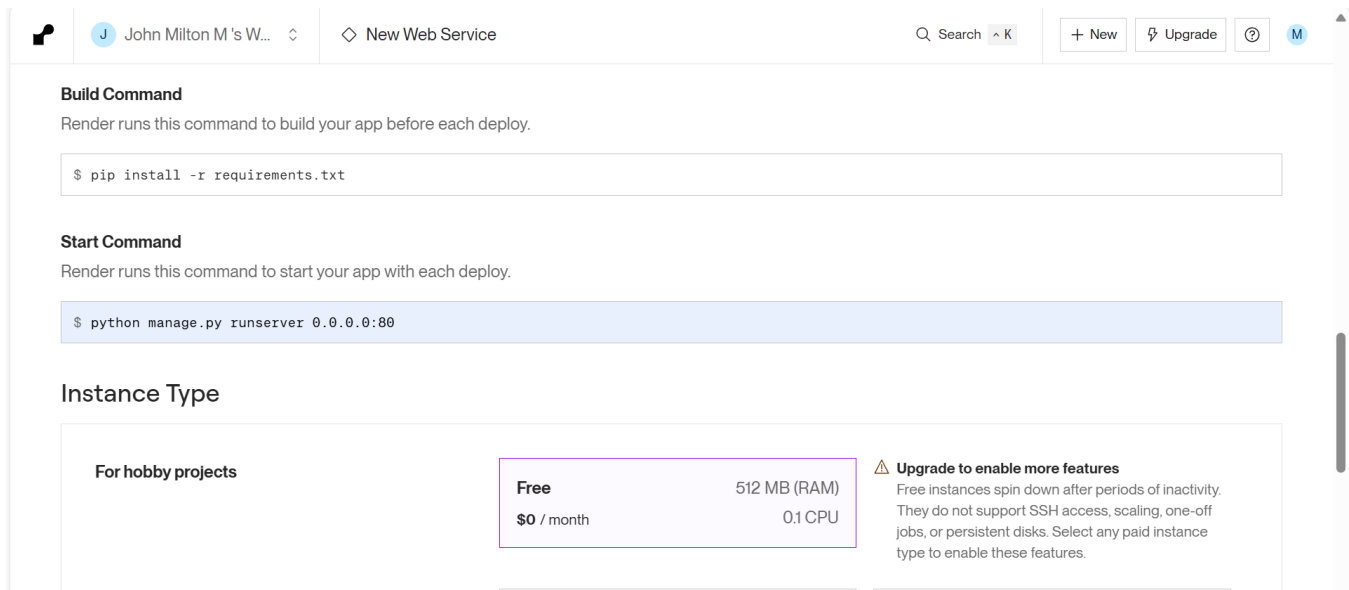


Step 3: Upload the project to github Repository and connect to render.



Step 4: Add start command and select Instance type as free

```
py manage.py runserver 0.0.0.0:80
```



Build Command

Render runs this command to build your app before each deploy.

```
$ pip install -r requirements.txt
```

Start Command

Render runs this command to start your app with each deploy.

```
$ python manage.py runserver 0.0.0.0:80
```

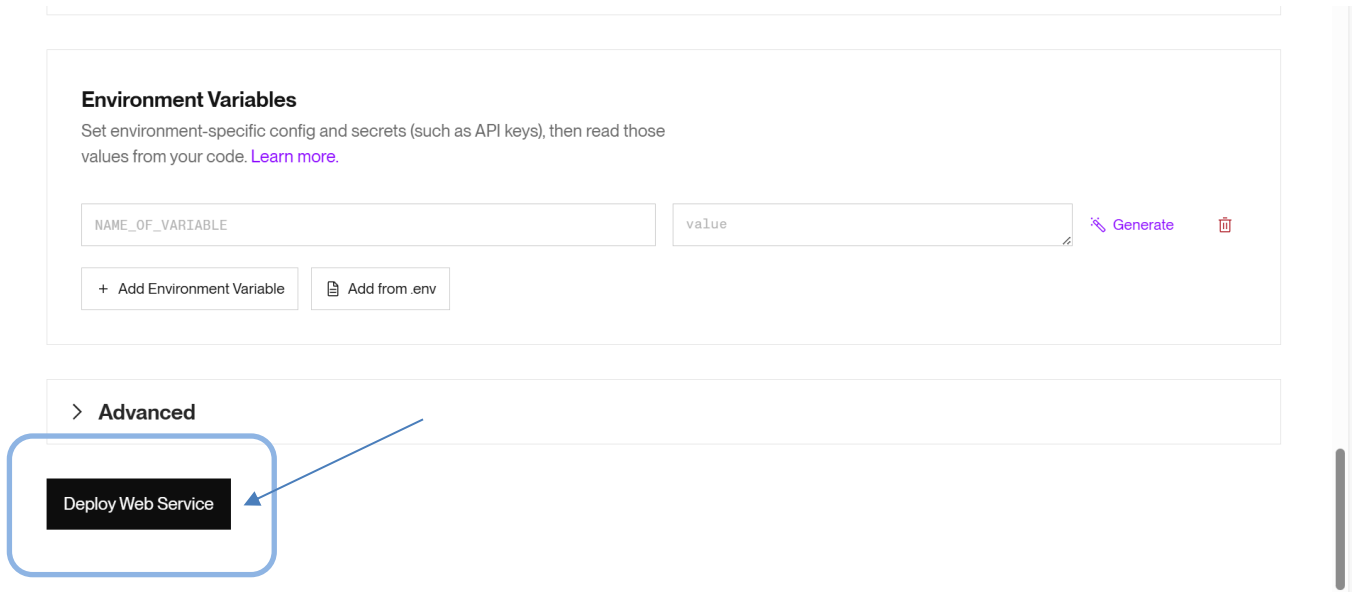
Instance Type

For hobby projects

Free	512 MB (RAM)
\$0 / month	0.1 CPU

⚠ Upgrade to enable more features
Free instances spin down after periods of inactivity. They do not support SSH access, scaling, one-off jobs, or persistent disks. Select any paid instance type to enable these features.

Step 5: Click on Deploy and wait until you get green signal.



Environment Variables

Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more.](#)

NAME_OF_VARIABLE	value

+ Add Environment Variable Add from .env

> Advanced

Deploy Web Service

Now click on the link. Voila, Your website is live !!!