

MEDIA STREAMING WITH IBM CLOUD STREAMING

PHASE – 3

DEVELOPMENT PHASE – 1

Date	23 October 2023
Team ID	5566
Team Name	Proj 227254 Team 2
Project Name	Media streaming with IBM Cloud streaming

1.INTRODUCION:

Streaming files like audio, video and others are stored on a server somewhere on the world wide web(WWW). When a user request file ,It gets transmitted over the web as sequential packets of data that are streamed instantly . Since streaming data is broken down into data packets ,its transmission is similar to that of other types of data sent over the internet.

An audio or video player hosted by the browser accepts the flow of data packets from the streaming service's remote server and interprets them as video or audio, then plays the media for the user.

STREAMING REQUIREMENTS:

Streaming usually requires a reliable, high-speed internet connection because the media files must be retrieved from a remote location and then delivered to a user's local system with minimal lag or latency (delay). A slow connection decreases the speed at which the content is delivered, affecting the user's streaming experience.

INSTALL PYTHON IN OS:

1. Download PyCharm:

Visit the JetBrains website (<https://www.jetbrains.com/pycharm/download/>) and download the community (free) or professional version of PyCharm, depending on your needs. Make sure to download the Windows version.

2. Run the Installer:

Locate the downloaded installer (usually an .exe file) and double-click it to run the installation.

3. Choose Installation Type:

During the installation, you'll be prompted to select the installation type. You can choose the default settings or customize them according to your preferences. You can also select the option to create associations for .py files, which allows you to open Python scripts with PyCharm by default.

4. Select the Destination Folder:

Choose the folder where you want to install PyCharm, or leave it at the default location.

5. Create Shortcuts:

You can choose whether you want to create desktop shortcuts or add PyCharm to the Start menu.

6. Complete the Installation:

Click the "Install" button to start the installation process. PyCharm will be installed on your system.

7. Launch PyCharm:

Once the installation is complete, you can launch PyCharm by checking the "Run PyCharm" option in the installer or by finding it in your Start menu or desktop shortcuts.

INSTALLING PACKAGES :

1.PACKAGE NAME: Flask

USE : to use flask framework in python

COMMAND TO INSTALL: pip install flask

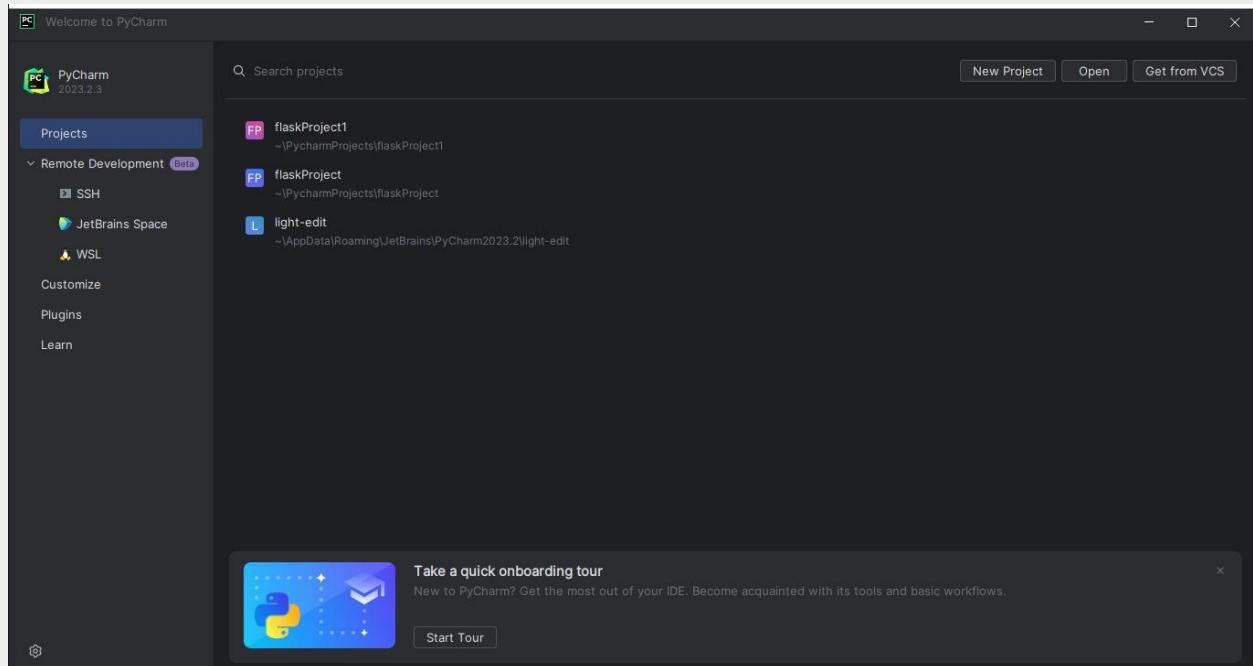
```
Command Prompt
Collecting flask
  Downloading flask-3.0.0-py3-none-any.whl (99 kB)
----- 99.7/99.7 kB 5.6 MB/s eta 0:00:00
Collecting Werkzeug>=3.0.0
  Downloading werkzeug-3.0.0-py3-none-any.whl (226 kB)
----- 226.6/226.6 kB 13.5 MB/s eta 0:00:00
Collecting Jinja2>=3.1.2
  Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)
----- 133.1/133.1 kB ? eta 0:00:00
Collecting itsdangerous>=2.1.2
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting click>=8.1.3
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
----- 97.9/97.9 kB 5.5 MB/s eta 0:00:00
Collecting blinker>=1.6.2
  Downloading blinker-1.6.3-py3-none-any.whl (13 kB)
Collecting colorama
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-2.1.3-cp311-cp311-win_amd64.whl (17 kB)
Installing collected packages: MarkupSafe, itsdangerous, colorama, blinker, Werkzeug, Jinja2, click, flask
Successfully installed Jinja2-3.1.2 MarkupSafe-2.1.3 Werkzeug-3.0.0 blinker-1.6.3 click-8.1.7 colorama-0.4.6 flask-3.0.0 itsdangerous-2.1.2

[notice] A new release of pip available: 22.3 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\Elcot>
```

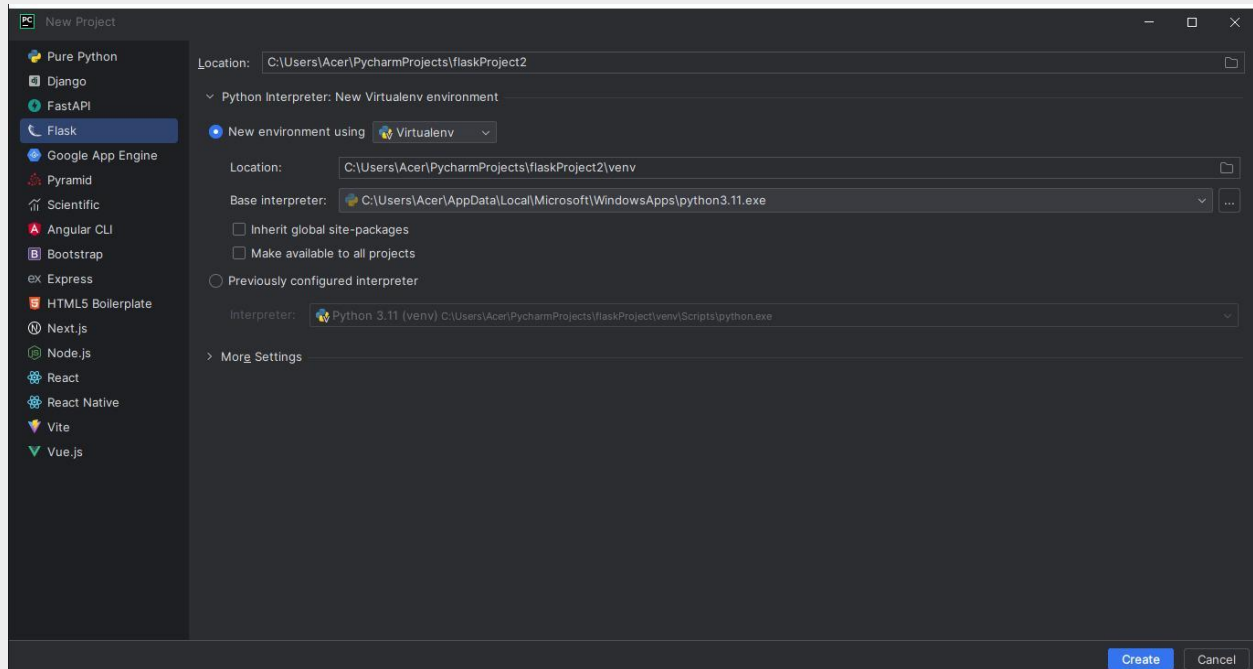
GETTING STARTED WITH PYCHARM:

1. To create a new project click on new project:

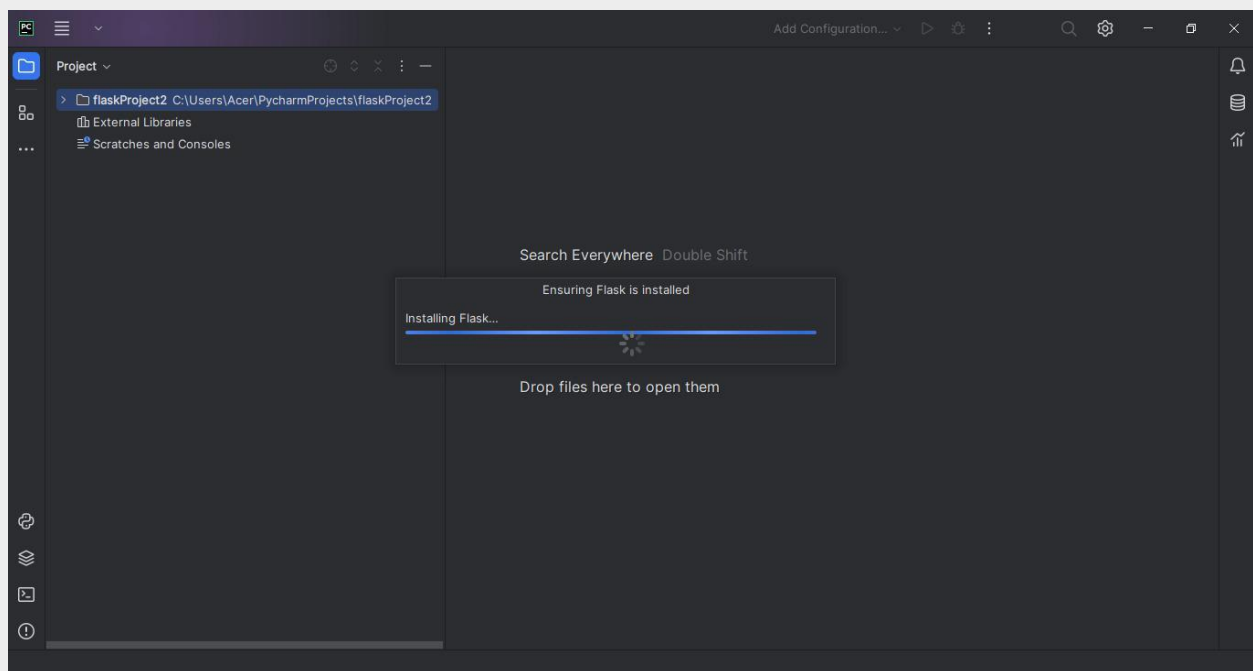


2. Select flask to create a project in flask program
 - Select your environment as you need.

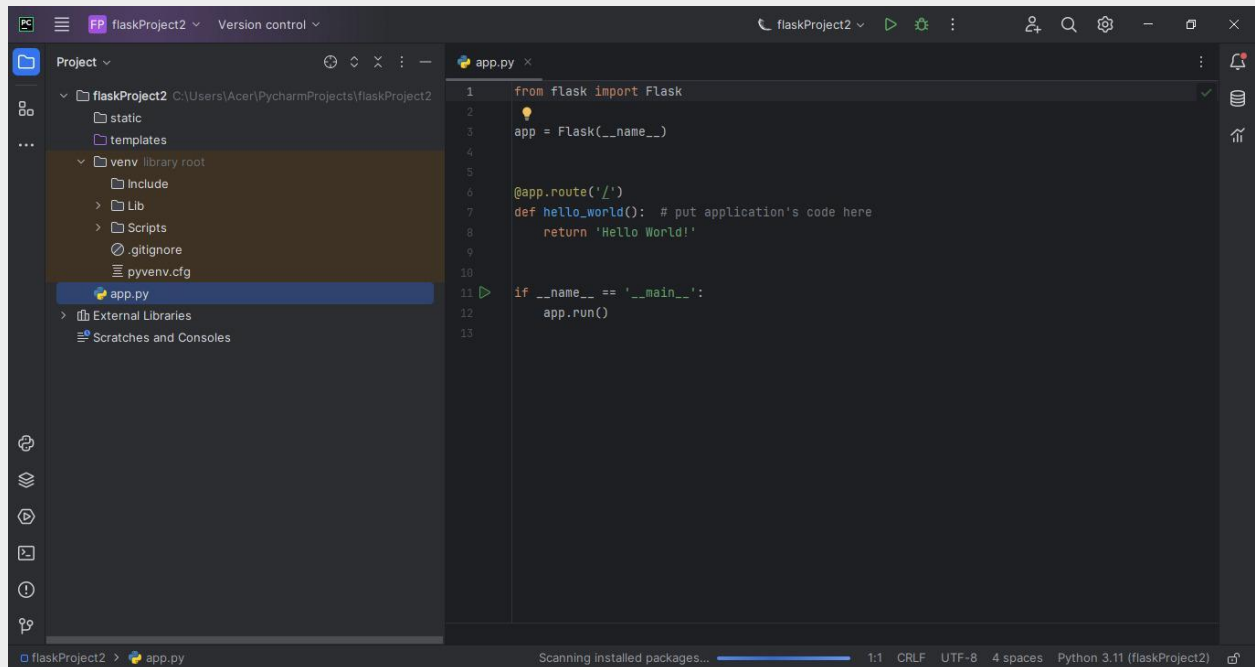
- Select the required path you need the projects to be stored.



3. Pycharm will automatically installs required packages for flask automatically.



4. Required folders will be installed automatically. Then we can ready to code successfully.



STEPS TO CREATE LOGIN AND REGISTRATION PAGE:

1. Code this to render a html code in flask

```
from flask import Flask,render_template
```

```
app = Flask(__name__)
```

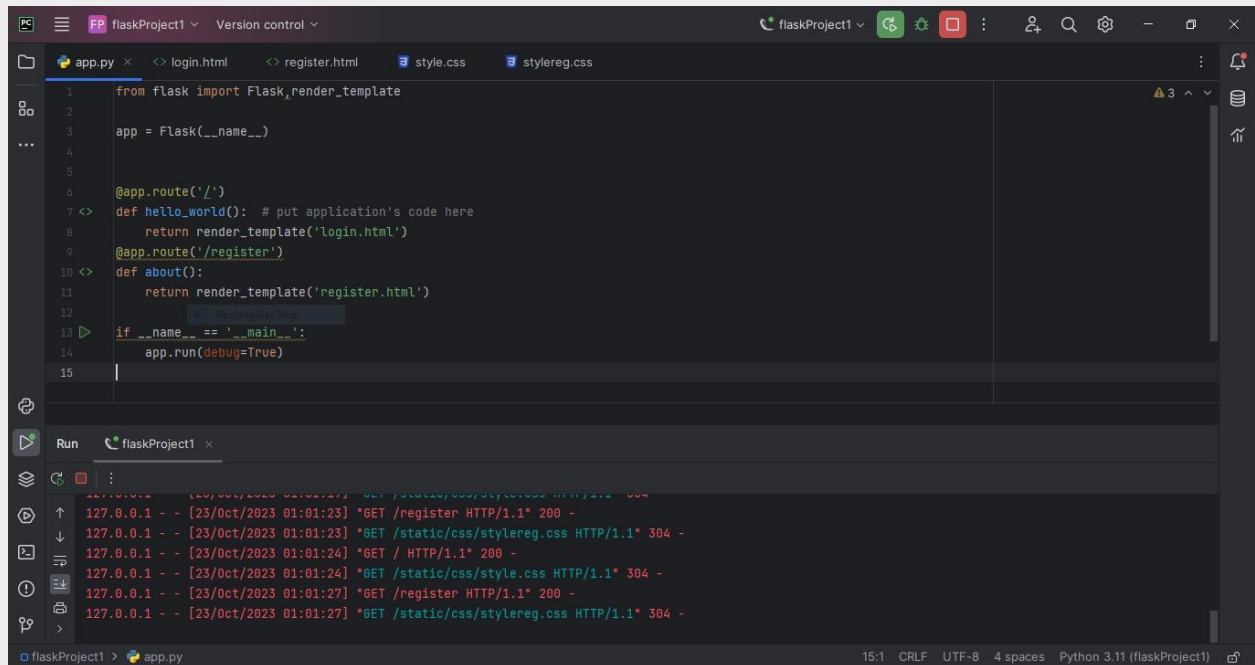
```
@app.route('/')
```

```
def hello_world(): # put application's code here  
    return render_template('login.html')
```

```
@app.route('/register')
```

```
def about():  
    return render_template('register.html')
```

```
if __name__ == '__main__':  
    app.run(debug=True)
```



```
1 from flask import Flask, render_template
2
3 app = Flask(__name__)
4
5
6 @app.route('/')
7 def hello_world(): # put application's code here
8     return render_template('login.html')
9
10 @app.route('/register')
11 def about():
12     return render_template('register.html')
13
14 if __name__ == '__main__':
15     app.run(debug=True)
```

```
127.0.0.1 - - [23/Oct/2023 01:01:23] "GET /register HTTP/1.1" 200 -
127.0.0.1 - - [23/Oct/2023 01:01:23] "GET /static/css/stylereg.css HTTP/1.1" 304 -
127.0.0.1 - - [23/Oct/2023 01:01:24] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [23/Oct/2023 01:01:24] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [23/Oct/2023 01:01:27] "GET /register HTTP/1.1" 200 -
127.0.0.1 - - [23/Oct/2023 01:01:27] "GET /static/css/stylereg.css HTTP/1.1" 304 -
```

2. Code for login page html:

<!DOCTYPE html>

<html>

<head>

<title>Slide Navbar</title>

<link rel="stylesheet" type="text/css" href="slide navbar style.css">

<link

href="https://fonts.googleapis.com/css2?family=Jost:wght@500&display=swap" rel="stylesheet">

</head>

<body>

<div class="main">

```
<input type="checkbox" id="chk" aria-hidden="true">
```

```
<div class="signup">
```

```
<form>
```

```
<label for="chk" aria-hidden="true">Sign  
up</label>
```

```
<input type="text" name="txt" placeholder="User  
name" required="">
```

```
<input type="email" name="email"  
placeholder="Email" required="">
```

```
<input type="password" name="pswd"  
placeholder="Password" required="">
```

```
<button>Sign up</button>
```

```
</form>
```

```
</div>
```

```
<div class="login">
```

```
<form>
```

```
<label for="chk" aria-hidden="true">Login</label>
```

```
<input type="email" name="email"  
placeholder="Email" required="">
```

```
<input type="password" name="pswd"  
placeholder="Password" required="">
```

```
<button>Login</button>
```

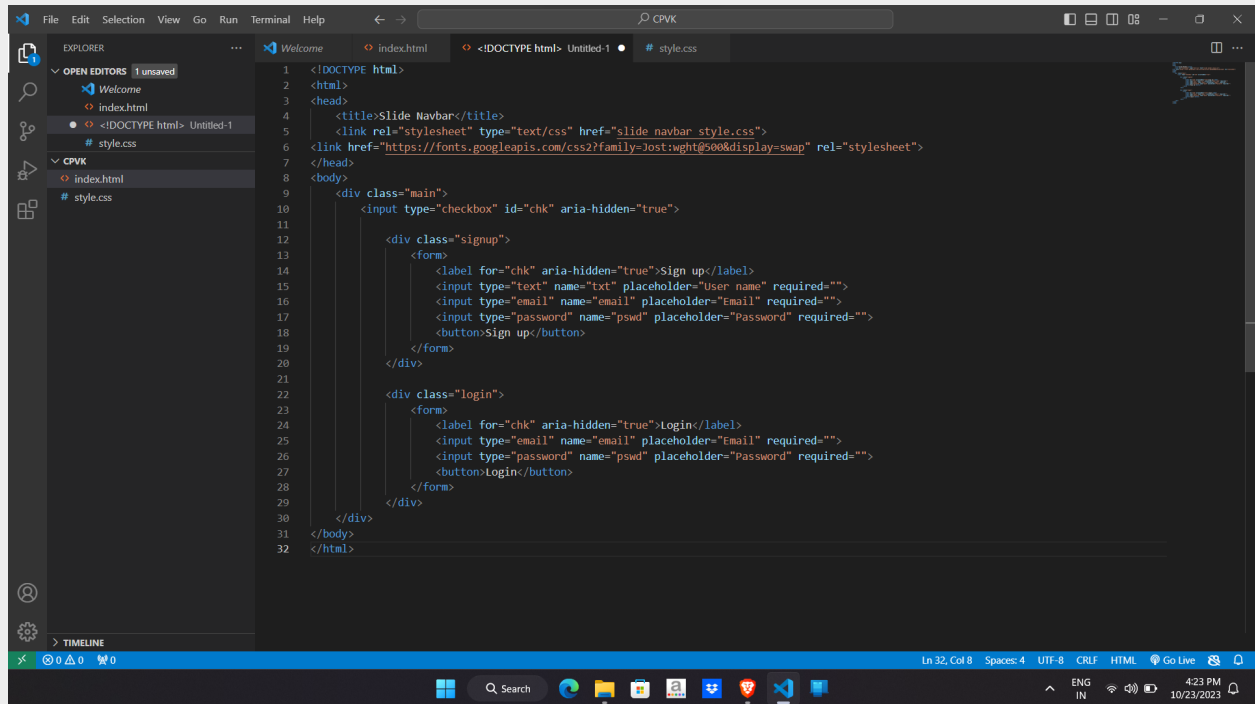
```
</form>
```

</div>

</div>

</body>

</html>



3. SYNCHRONIZING A CSS FILE WITH LOGIN HTML FOR DESIGNING PURPOSE:

body{

margin: 0;

padding: 0;


```
display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

font-family: 'Jost', sans-serif;

background: linear-gradient(to bottom, #0f0c29, #302b63, #24243e);
}

.main{

width: 350px;

height: 500px;

background: red;

overflow: hidden;

background:
url("https://doc-08-2c-docs.googleusercontent.com/docs/securesc/68c90smigl
ihng9534mvqmq1946dmis5/fo0picsp1nhiucmc0l25s29respgpr4j/16315242750
00/03522360960922298374/03522360960922298374/1Sx0jhdpEpnNIydS4rnN
4kHSJtU1EyWka?e=view&authuser=0&nonce=gcrocepgbb17m&user=0352236
0960922298374&hash=tfhgbs86ka6divo3llbvp93mg4csvb38") no-repeat
center/ cover;

border-radius: 10px;

box-shadow: 5px 20px 50px #000;
}

#chk{

display: none;
}
```

```
.signup{
    position: relative;
    width:100%;
    height: 100%;
}
label{
    color: #fff;
    font-size: 2.3em;
    justify-content: center;
    display: flex;
    margin: 60px;
    font-weight: bold;
    cursor: pointer;
    transition: .5s ease-in-out;
}
input{
    width: 60%;
    height: 20px;
    background: #e0dede;
    justify-content: center;
    display: flex;
    margin: 20px auto;
    padding: 10px;
```

```
border: none;

outline: none;

border-radius: 5px;
}

button{

width: 60%;

height: 40px;

margin: 10px auto;

justify-content: center;

display: block;

color: #fff;

background: #573b8a;

font-size: 1em;

font-weight: bold;

margin-top: 20px;

outline: none;

border: none;

border-radius: 5px;

transition: .2s ease-in;

cursor: pointer;

}

button:hover{

background: #6d44b8;
```

```
}
```

```
.login{
```

```
    height: 460px;
```

```
    background: #eee;
```

```
    border-radius: 60% / 10%;
```

```
    transform: translateY(-180px);
```

```
    transition: .8s ease-in-out;
```

```
}
```

```
.login label{
```

```
    color: #573b8a;
```

```
    transform: scale(.6);
```

```
}
```

```
#chk:checked ~ .login{
```

```
    transform: translateY(-500px);
```

```
}
```

```
#chk:checked ~ .login label{
```

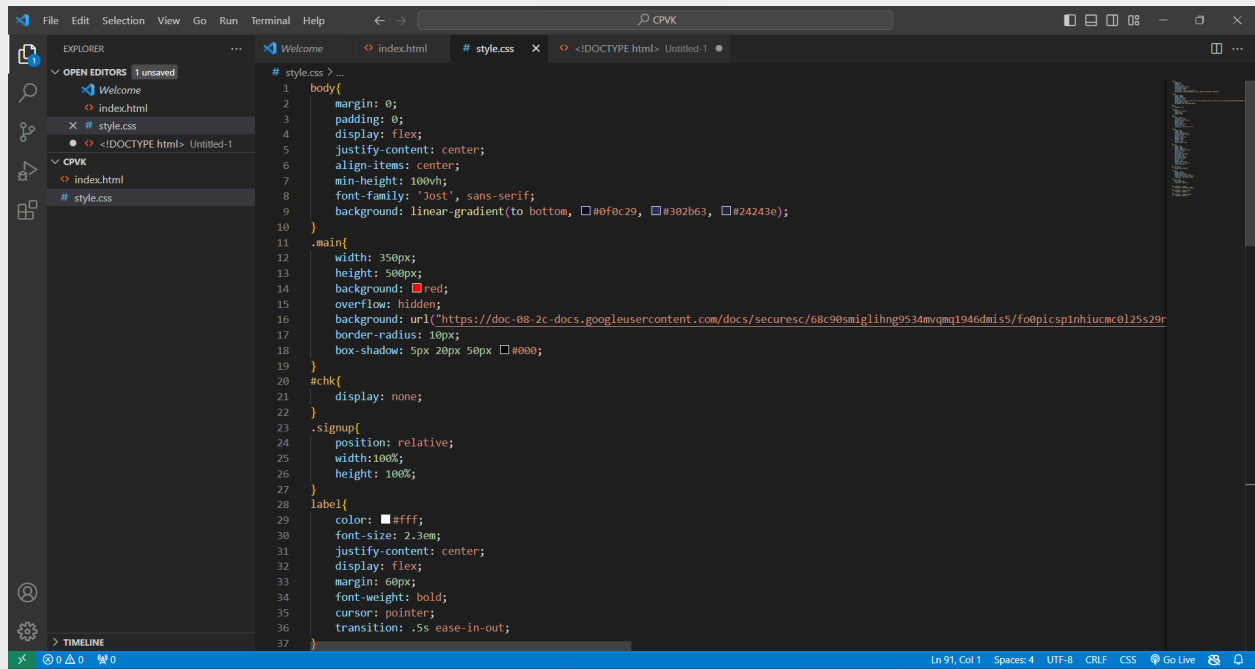
```
    transform: scale(1);
```

```
}
```

```
#chk:checked ~ .signup label{
```

```
    transform: scale(.6);
```

```
}
```



4. CREATING A SQL TO STORE THE DATA'S IN THE DATABASE:

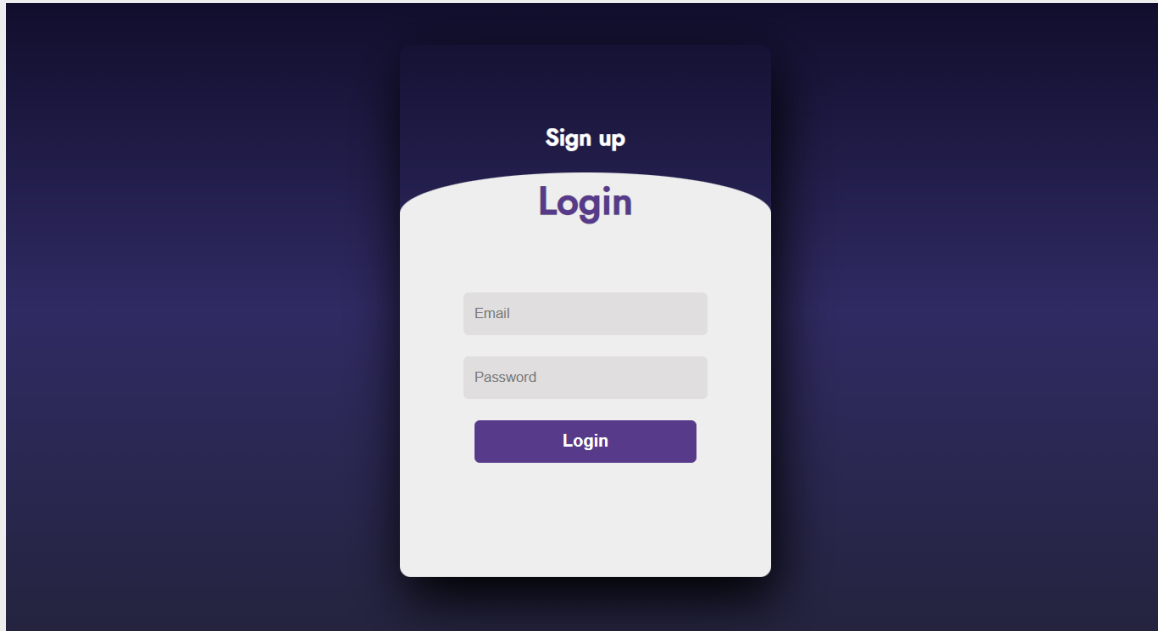
CREATE DATABASE IF NOT EXISTS UserDatabase;

USE UserDatabase;

CREATE TABLE IF NOT EXISTS Users (
 UserID INT AUTO_INCREMENT PRIMARY KEY,
 Username VARCHAR(50) NOT NULL,
 MobileNo VARCHAR(15) NOT NULL,
 EmailID VARCHAR(100) NOT NULL,
 Password VARCHAR(255) NOT NULL

);

5. FINAL OUTPUT FOR LOGIN PAGE:



The image displays a login page with a dark blue background. A white, rounded rectangular card is centered on the page. At the top of the card, the text "Sign up" is written in white, and below it, the word "Login" is written in a larger, bold, dark blue font. Underneath the "Login" text, there are two input fields: the first is labeled "Email" and the second is labeled "Password", both in a light gray font. Below these input fields is a solid dark blue button with the word "Login" written in white. The card has a subtle drop shadow against the background.

6. FINAL OUTPUT FOR REGISTRATION PAGE:

Sign up

User name

Email

Password

Sign up

Login