INNOMATİCS®
RESEARCH LABS

INNOVATION. AUTOMATION. ANALYTICS

PROJECT ON

Regex Matching Web App Development Project

Objective of the Project

The aim of the project is to replicate the core functionality of the website regex101.com. This involves building, a web application that tests a string using a regular expression (regex) and displays all of the matches found.

Project Implementation

Creating a new directory for the **regex_matching_web_app** project and navigate into it

1. Setting up virtual environment
   python -m venv .env_regex
2. Activate the virtual environment scripts
   .env_regex\Scripts\activate
3. Install Flask framework
   `pip install flask`

Initializing a new flask application i.e, creating a new python file named as ‘app.py’ and importing flask and creating a flask instance.

```python
# step 1 - import flask
from flask import Flask, request, render_template
import re

# step 2 - create a flask object __name__ parameter
app = Flask(__name__)
```

Creating a new route (“/”) for index page and defining a function where users can input test string and regex pattern. It renders an HTML template containing a form with fields for the test string and regex, and a submit button.

```python
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        test_string = request.form['test_string']
        regex_pattern = request.form['regex_pattern']
    return render_template('index.html')
```

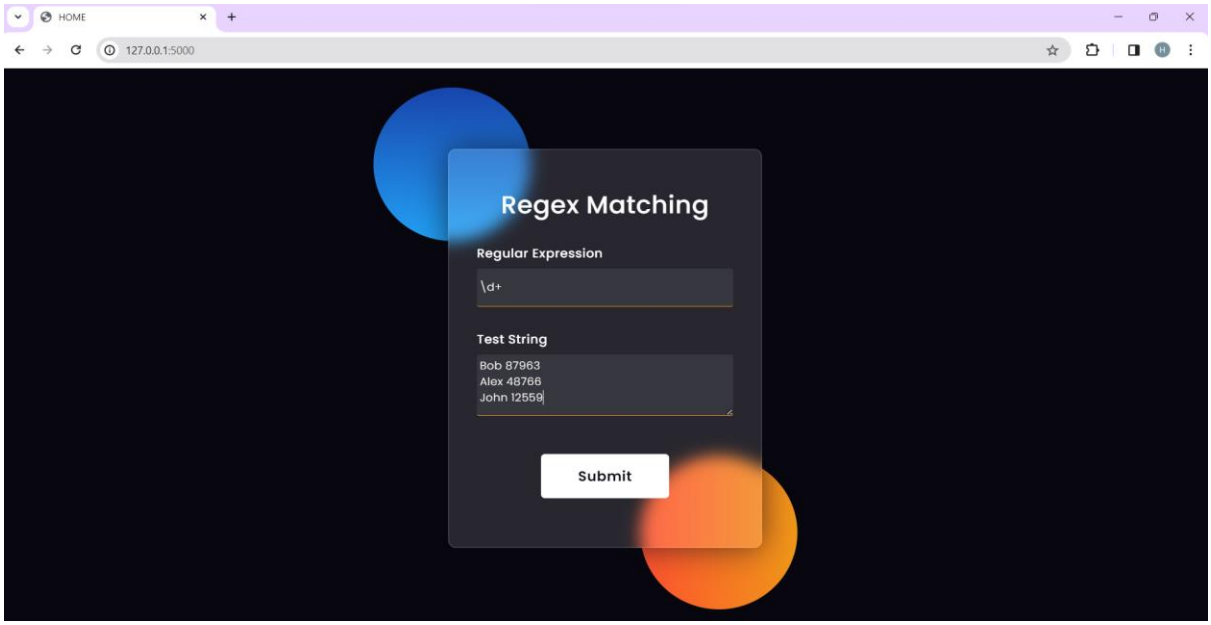
```html
<html lang="en">
<head>
    <style media= screen >
    </style>
</head>
<body>
    <div class="background">
        <div class="shape"></div>
        <div class="shape"></div>
    </div>
    <form action="/results" method="post" id="regexForm">
        <h3>Regex Matching</h3>
        <label for="regex_pattern">Regular Expression</label>
        <input type="text" id="regex_pattern" name="regex_pattern" placeholder="enter the regular expression" required>
        <label for="test_string">Test String</label>
        <textarea id="test_string" name="test_string" rows="4" cols="50" placeholder="enter the test string" required></textarea>
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

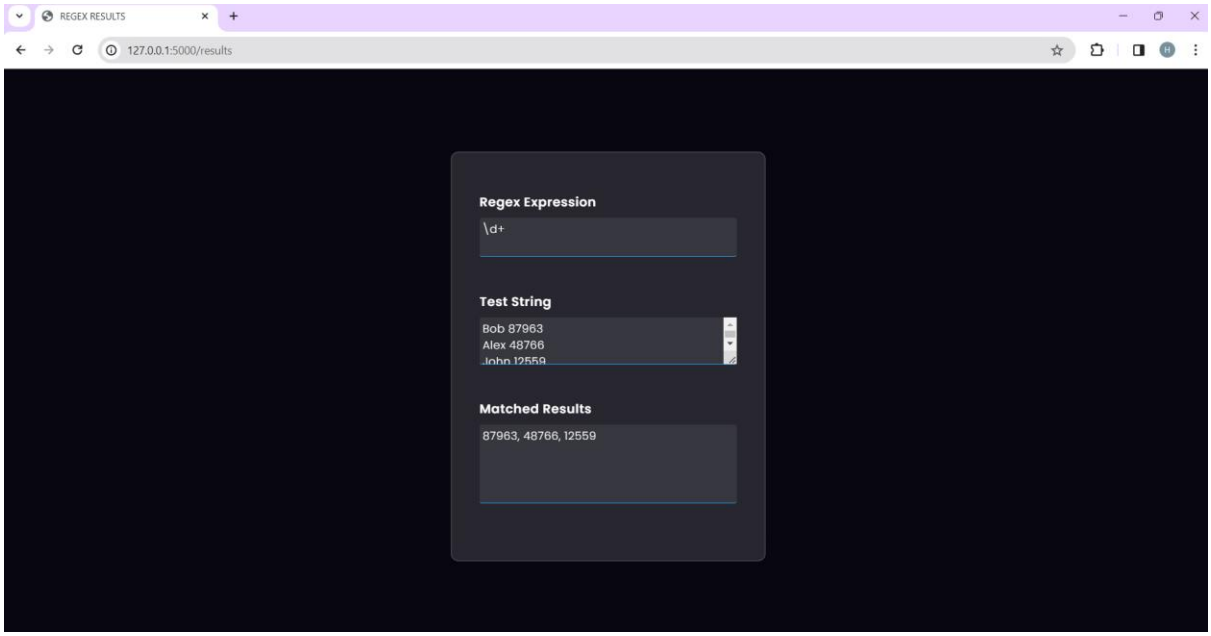
Defining a new route ("/results") in `app.py` file to handle form submission and it extracts the test string and regex submitted by the user from the form data.

```python
@app.route('/results', methods=['POST'])
def results():
    test_string = request.form['test_string']
    regex_pattern = request.form['regex_pattern']
    matched_strings = re.findall(regex_pattern, test_string)
    return render_template('results.html', test_string=test_string, regex_pattern=regex_pattern, matched_strings=matched_strings)
```

```html
<html lang="en">
<head>
    <style media="screen">
    </style>
</head>
<body>
    <form>
        <strong>Regex Expression</strong>
        <p>{{ regex_pattern }}</p><br>
        <strong>Test String</strong>
        <textarea>{{ test_string }}</textarea><br>
        <strong>Matched Results</strong><br>
        <font color="#ffffff">{{ matched_strings|join(', ') }}</font>
    </form>
</body>
</html>
```

Run the Flask application **`python app.py`** and Open a web browser and navigate to http://127.0.0.1:5000 to access your application.





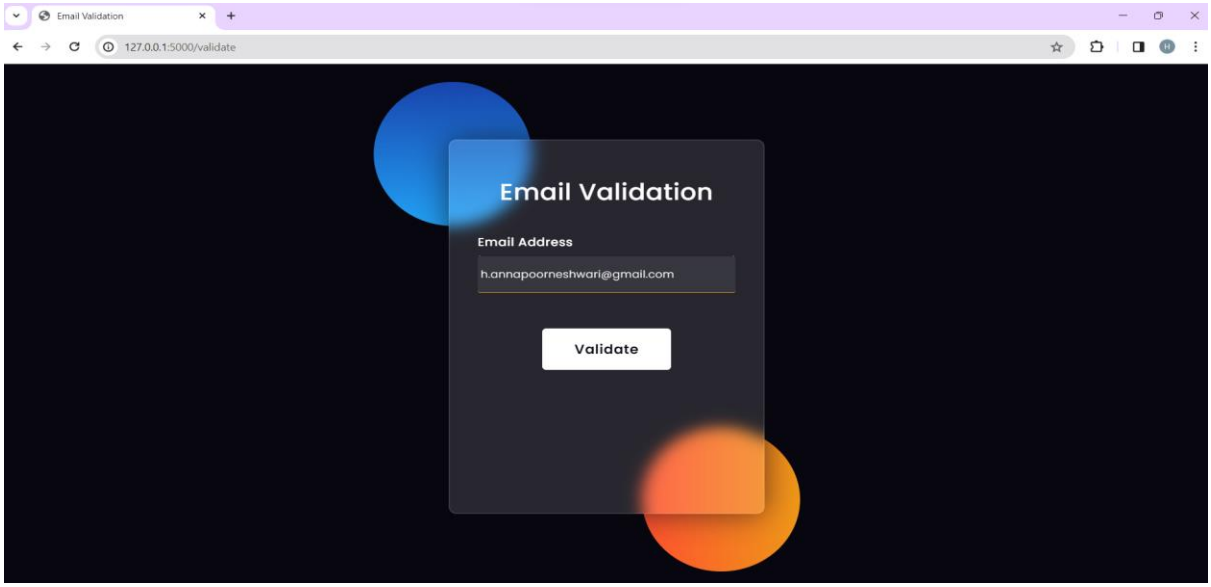
Defining a new route ("/validate") in `app.py` file for email_validation page and defining a function where users input email-id. It renders an HTML template containing a form with fields for the email-id and a validate button. Also, it handles form submission and validates the email-id submitted by the user from the form data.

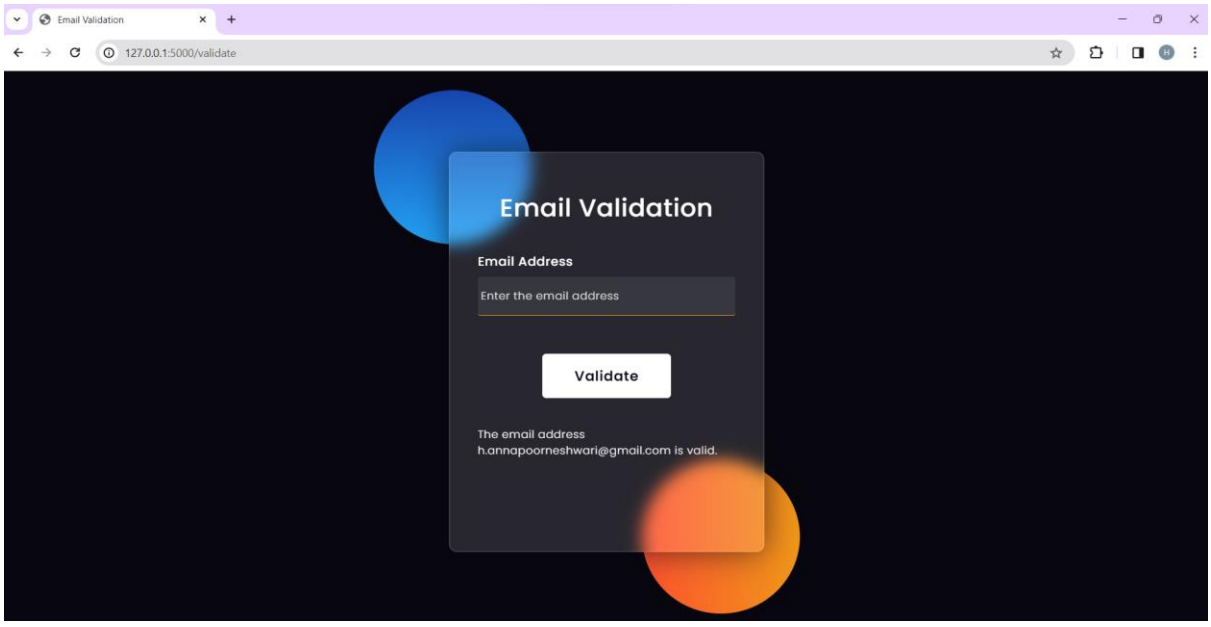
```python
@app.route('/validate', methods=['GET','POST'])
def validate():
    if request.method == 'POST':
        email = request.form.get('email')
        if re.match(r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$', email):
            return render_template('email_validation.html', email=email, valid=True)
        else:
            return render_template('email_validation.html', email=email, valid=False)
    return render_template("email_validation.html")
```

```html
<html lang="en">
<head>
    <style media="screen">
    </style>
</head>
<body>
    <div class="background">
        <div class="shape"></div>
        <div class="shape"></div>
    </div>
    <form method="POST" action="/validate" id="emailForm">
        <h3>Email Validation</h3><br><br>
        <label for="email">Email Address</label>
        <input type="email" id="email" name="email" placeholder="Enter the email address" required>
        <button type="submit">Validate</button><br><br>

        {% if email is defined and valid is defined %}
        <p>The email address {{ email }} is {% if valid %}valid{% else %}invalid{% endif %}.</p>
        {% endif %}
    </form>
</body>
</html>
```

Run the Flask application **`python app.py`** and Open a web browser and navigate to http://127.0.0.1:5000/validate to access your application.





Application Deployment on AWS Cloud

1. Create an **AWS Account** → Login to **AWS Management Console** → Select **EC2** Service to create virtual servers.
2. Click **Launch instance** under EC2 Dashboard
   a. Under Name and Tags, Name as "**regex_matching_webapp**"
   b. Under Amazon Machine Image, Select **Ubuntu**
   c. Under Instances type, Select **t2.micro**
   d. Under Key pair (Login), click **Create new key pair** → Enter Key pair name and click Create key pair, **.pem file** gets downloaded automatically in your local system.
   e. Keep everything default and click on **Launch instance**
3. From Navigation bar, click **Security Groups** under Network & Security → click **Create security group**
   a. Under Basic details, provide name as "**anywhere-sg**" → description as "**anywhere**"
   b. Under Inbound rules, click **Add rule** → Select Type as "**All traffic**" → Select Source as "**Anywhere-IPv4**" → click **Create security group**
4. From Navigation bar, click **Network Interfaces** under Network & Security → Right click on Network interface ID → Select **Change security groups**
   a. Under Associated security groups, Select **anywhere-sg** from Search bar → click **Add security group** → click **Save**
5. From Navigation bar, click **Instances** under Instances → Right click on Instance ID → click **Connect** → Select **SSH client** → Copy the SSH client command
6. Copy the **.pem file** and paste the file in project location.
7. Open command prompt from project location → Paste the **SSH client** command → Press Enter
   a. Type **pwd** to check home directory
   b. Type **ls** to check list of directory and files
8. Create a new folder as "**webapp**" in project location → Upload **templates folder** and **app.py** file into **webapp folder**
9. Open another command prompt from project location
   a. Type **.env_regex\Scripts\activate**
   b. Type **pip freeze > requirements.txt**
   c. Close the command prompt
10. Upload **requirements.txt file** into webapp folder
11. Type **logout** in command prompt
12. Type **scp -r -i "keypair.pem" webapp <Remote-Server>:~/**
13. Login into Remote Server i.e., paste the **SSH client**
14. Type **ls**
15. Type **sudo apt update**
16. Type **sudo apt upgrade**
17. Type **sudo apt install python3-pip**
18. Type **ls**
19. Type **cd webapp/**
20. Type **pip install -r requirements.txt**
21. Type **nohup python3 app.py &**
22. To view all process ID → Type **top -u $USER**

Open the browser and Paste the public IP address along with port 5000

http://3.80.32.27:5000/ - For Regex Matching

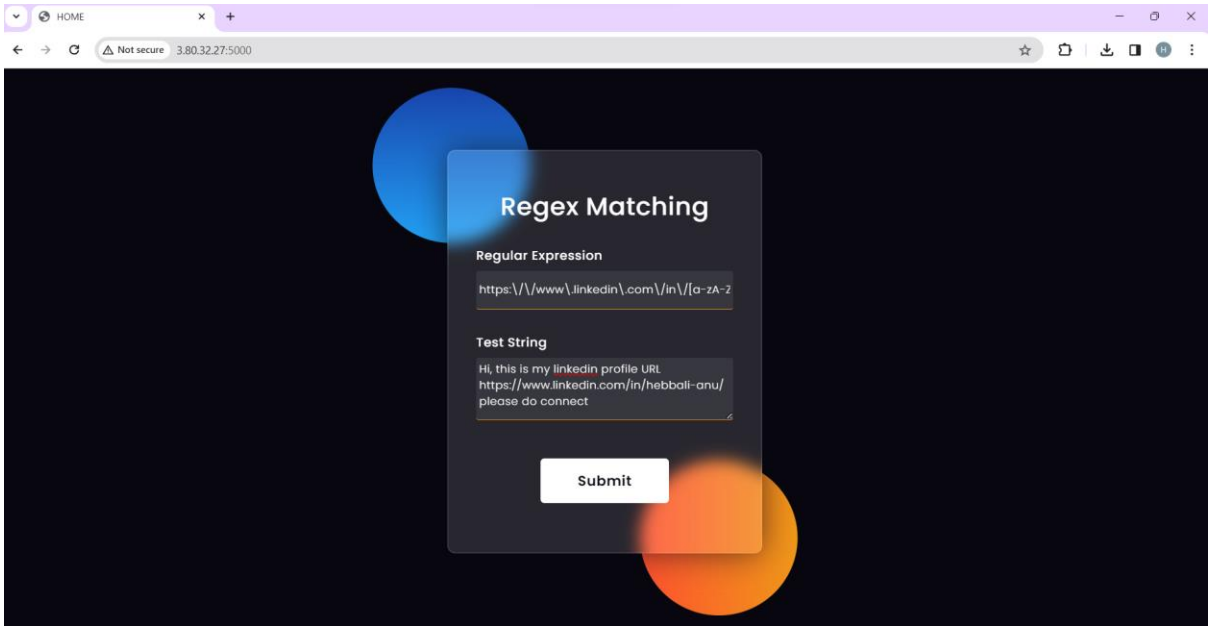
http://3.80.32.27:5000/validate - For Email Validation

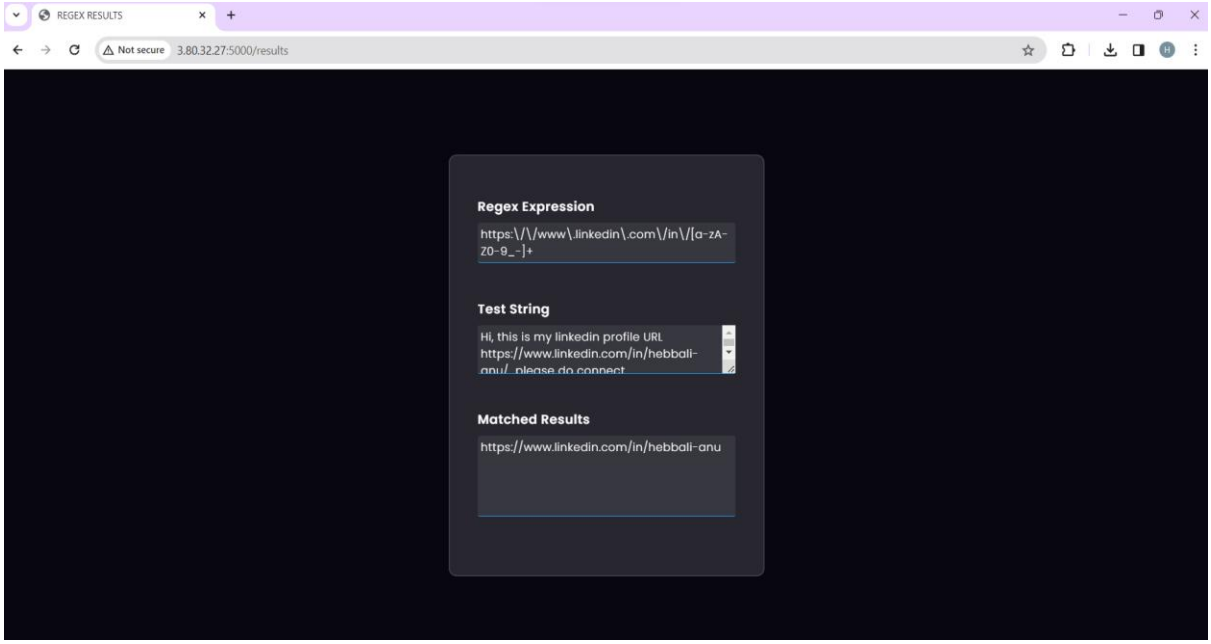
Email Pattern:
\b[A-Za-z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-z|A-Z]{2,}\b

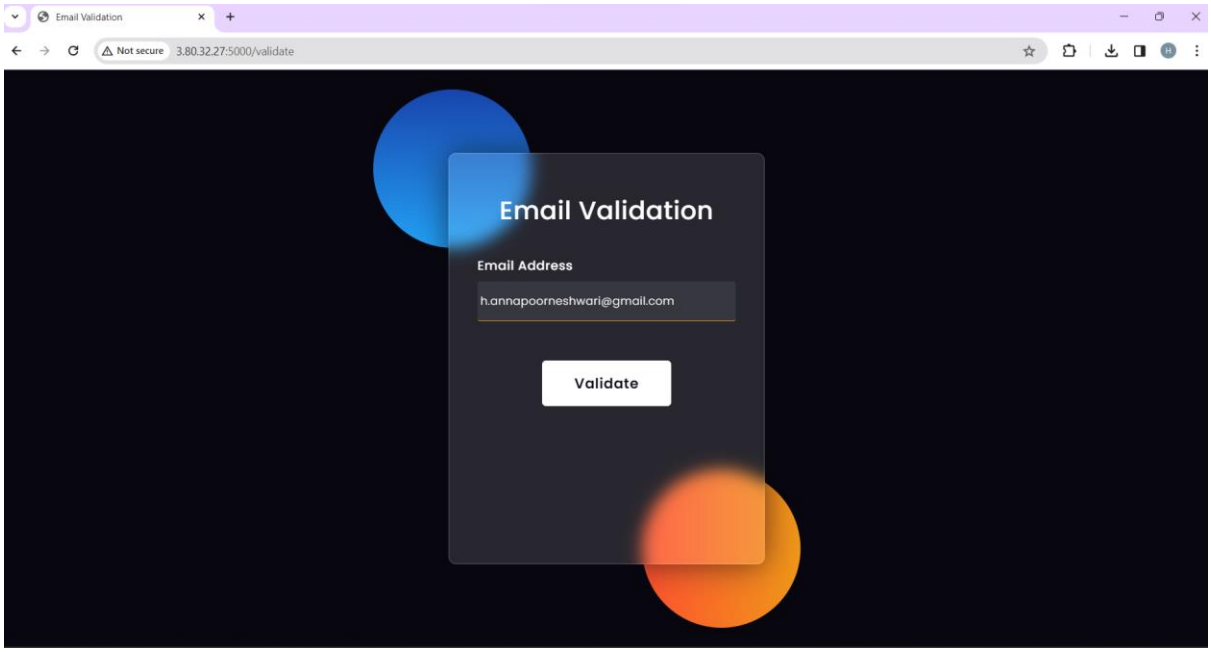
Digits Pattern:
\d+

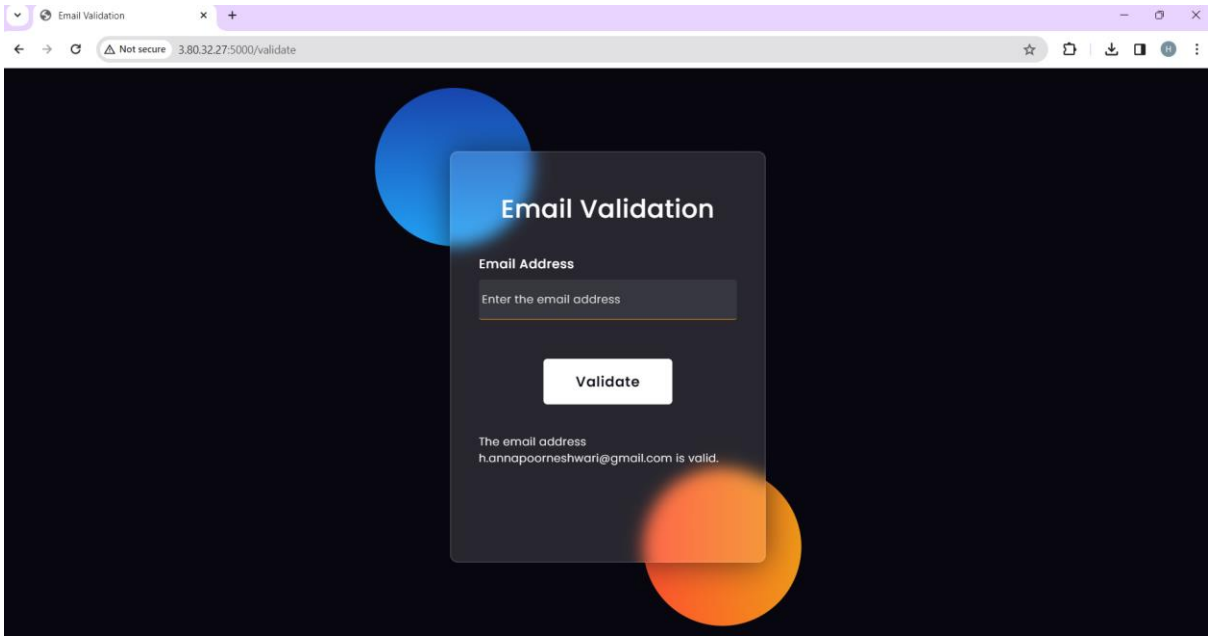
Capital characters Pattern:
([A-Z])\w+

LinkedIn Profile URL Pattern:
https:\/\/www\.linkedin\.com\/in\/[a-zA-Z0-9_-]+









NOTE: If want to stop the server permanently → Type **kill <process_ID>**