INNOMATICS®
RESEARCH LABS

INNOVATION. AUTOMATION. ANALYTICS

PROJECT ON

REGEX MATCHES & EMAIL VERIFICATION

About me

- I did Complete Bachelor of Computer Science.
- The motivations for learning data science can vary from person to person, but it often involves a combination of intellectual curiosity, career aspirations, and the desire to make a positive impact .
- I am currently working as a data science intern .
- GitHub Link : https://github.com/sravanthiveerla/internship/upload/main/task_in2
- LinkedIn : www.linkedin.com/in/sravanthi-veerla

Scenario

Develop a web application similar to regex101.com, aimed at providing users with a platform to test regular expressions (regex) against a given input string. Users can input a test string and a regular expression into the application. The application then processes the input, identifies and displays all matches found between the test string and the provided regular expression. This functionality empowers users to efficiently test and validate their regex patterns, aiding in various text processing and pattern matching tasks.

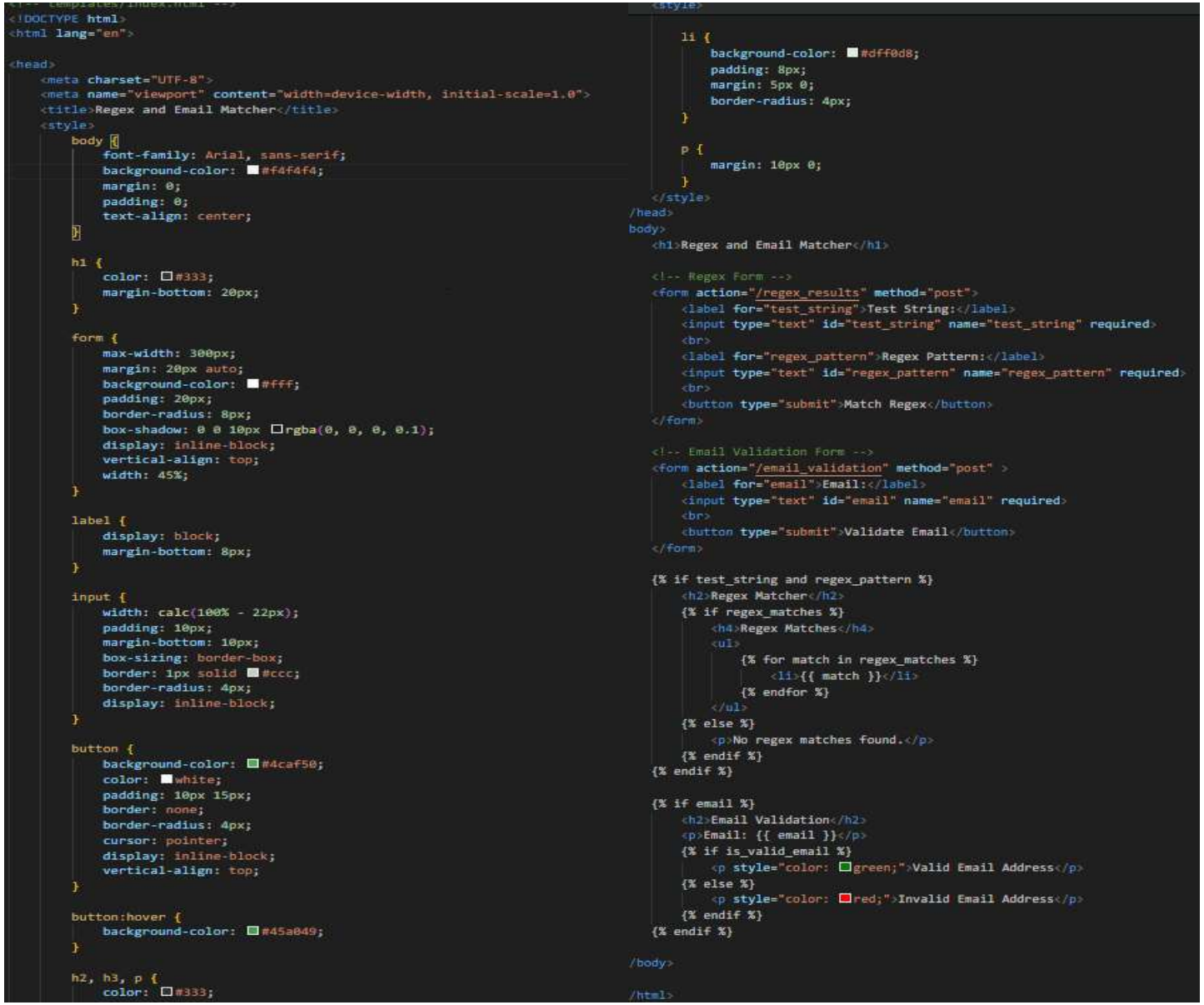
Initialize a Flask application:

This Flask application provides a platform for testing regular expressions and validating email addresses. Users input test strings and regex patterns to find matches, while also verifying email validity through a simple regex pattern.

```python
from flask import Flask, render_template, request
import re

app = Flask(__name__)

def regex_matcher(test_string, regex_pattern):
    matches = re.findall(regex_pattern, test_string)
    return matches

def is_valid_email(email):
    # Regex pattern for a simple email validation
    email_pattern = r'^[\w\.-]+@[a-zA-Z\d\.-]+\.[a-zA-Z]{2,}$'
    return re.match(email_pattern, email)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/regex_results', methods=['POST'])
def regex_results():
    test_string = request.form.get('test_string')
    regex_pattern = request.form.get('regex_pattern')

    regex_matches = regex_matcher(test_string, regex_pattern)

    return render_template('index.html', test_string=test_string, regex_pattern=regex_pattern,
                           regex_matches=regex_matches)



@app.route('/email_validation', methods=['POST'])
def email_validation():
    email_to_validate = request.form.get('email')

    is_valid_email_result = is_valid_email(email_to_validate)

    return render_template('index.html', email=email_to_validate,
                           is_valid_email=is_valid_email_result)

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

Create the HTML template

This HTML template creates a user-friendly interface for a Regex and Email Matcher web application. It includes forms for inputting test strings and regex patterns, as well as for validating email addresses. The design features a clean layout with CSS styling for enhanced readability. Additionally, it dynamically displays regex matches and email validation results based on user inputs, providing a seamless user experience.



```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Regex and Email Matcher</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 0;
            text-align: center;
        }

        h1 {
            color: #333;
            margin-bottom: 20px;
        }

        form {
            max-width: 300px;
            margin: 20px auto;
            background-color: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            display: inline-block;
            vertical-align: top;
            width: 45%;
        }

        label {
            display: block;
            margin-bottom: 8px;
        }

        input {
            width: calc(100% - 22px);
            padding: 10px;
            margin-bottom: 10px;
            box-sizing: border-box;
            border: 1px solid #ccc;
            border-radius: 4px;
            display: inline-block;
        }

        button {
            background-color: #4caf50;
            color: white;
            padding: 10px 15px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            display: inline-block;
            vertical-align: top;
        }

        button:hover {
            background-color: #45a049;
        }

        h2, h3, p {
            color: #333;
        }

        li {
            background-color: #dff0d8;
            padding: 8px;
            margin: 5px 0;
            border-radius: 4px;
        }

        p {
            margin: 10px 0;
        }
    </style>
</head>
<body>
    <h1>Regex and Email Matcher</h1>

    <!-- Regex Form -->
    <form action="/regex_results" method="post">
        <label for="test_string">Test String:</label>
        <input type="text" id="test_string" name="test_string" required>
        <br>
        <label for="regex_pattern">Regex Pattern:</label>
        <input type="text" id="regex_pattern" name="regex_pattern" required>
        <br>
        <button type="submit">Match Regex</button>
    </form>

    <!-- Email Validation Form -->
    <form action="/email_validation" method="post" >
        <label for="email">Email:</label>
        <input type="text" id="email" name="email" required>
        <br>
        <button type="submit">Validate Email</button>
    </form>

    {% if test_string and regex_pattern %}
        <h2>Regex Matcher</h2>
        {% if regex_matches %}
            <h4>Regex Matches</h4>
            <ul>
                {% for match in regex_matches %}
                    <li>{{ match }}</li>
                {% endfor %}
            </ul>
        {% else %}
            <p>No regex matches found.</p>
        {% endif %}
    {% endif %}

    {% if email %}
        <h2>Email Validation</h2>
        <p>Email: {{ email }}</p>
        {% if is_valid_email %}
            <p style="color: green;">Valid Email Address</p>
        {% else %}
            <p style="color: red;">Invalid Email Address</p>
        {% endif %}
    {% endif %}

</body>

</html>
```

INNOMATICS RESEARCH LABS

Define a route to handle form submission

Handling form submission involves defining a route in the web application backend to receive form data sent by users. In this process, data such as input fields' values is extracted from the form using request methods provided by the web framework (e.g., Flask's request.form.get()). The extracted data is then processed as required, such as performing regex matching or other operations. After processing, the results may be stored or manipulated as needed before rendering a response, typically by rendering a template with the processed data for display back to the user.

```python
@app.route('/regex_results', methods=['POST'])
def regex_results():
    test_string = request.form.get('test_string')
    regex_pattern = request.form.get('regex_pattern')

    regex_matches = regex_matcher(test_string, regex_pattern)

    return render_template('index.html', test_string=test_string, regex_pattern=regex_pattern,
                           regex_matches=regex_matches)
```

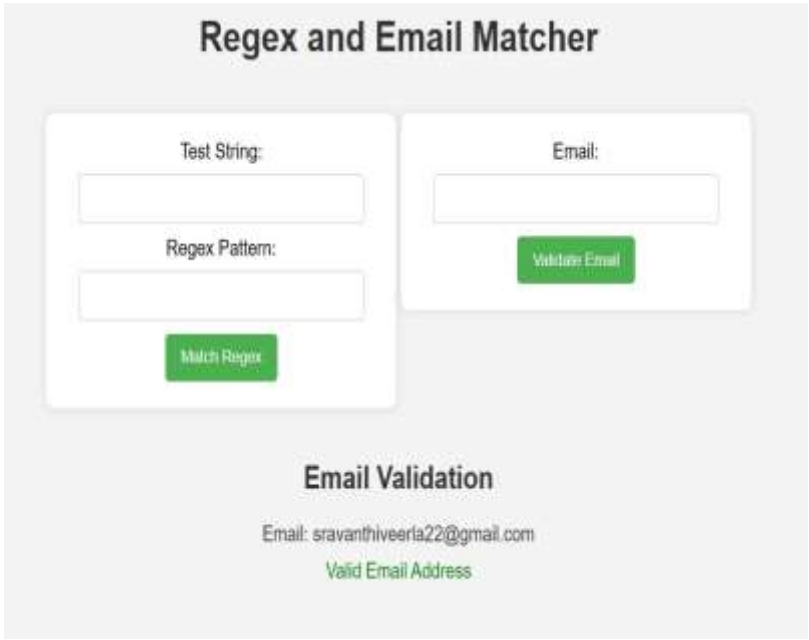
Render the result

This code defines a route to handle form submission in a Flask application, extracting the test string and regex pattern from the submitted form data. It then performs regex matching on the test string using the provided pattern and passes the matched strings to an HTML template. The HTML template, ‘index.html’, displays the test string, regex pattern, and matched strings in a structured format below the input form. If no matches are found, it provides a message indicating so.

```
<form action="/regex_results" method="post">
    <label for="test_string">Test String:</label>
    <input type="text" id="test_string" name="test_string" required>
    <br>
    <label for="regex_pattern">Regex Pattern:</label>
    <input type="text" id="regex_pattern" name="regex_pattern" required>
    <br>
    <button type="submit">Match Regex</button>
</form>


<form action="/email_validation" method="post" >
    <label for="email">Email:</label>
    <input type="text" id="email" name="email" required>
    <br>
    <button type="submit">Validate Email</button>
</form>

{% if test_string and regex_pattern %}
    <h2>Regex Matcher</h2>
    {% if regex_matches %}
        <h4>Regex Matches</h4>
        <ul>
            {% for match in regex_matches %}
                <li>{{ match }}</li>
            {% endfor %}
        </ul>
    {% else %}
        <p>No regex matches found.</p>
    {% endif %}
{% endif %}

{% if email %}
    <h2>Email Validation</h2>
    <p>Email: {{ email }}</p>
    {% if is_valid_email %}
        <p style="color: green;">Valid Email Address</p>
    {% else %}
        <p style="color: red;">Invalid Email Address</p>
    {% endif %}
{% endif %}
```

Render the result

The result of this application is a web interface where users can input test strings and regular expressions (regex) to perform matching. Upon submission, the application displays the matches found between the test string and the provided regex pattern. Users can also validate email addresses using a simple regex pattern, with the result indicating whether the email is valid or not. This intuitive interface aids users in efficiently testing and validating regex patterns and email addresses, enhancing text processing and pattern matching tasks.



INNOMATICS RESEARCH LABS

Conclusion

n summary, I've replicated regex101.com's functionality, enabling users to input test strings and regexes to find matches. Utilizing Flask, we created routes for the home page and form submission, leveraging Python's re module for regex matching. We also added an email validation feature and deployed the app on AWS. Testing validated its accuracy, offering users a dependable regex testing tool, enhancing our Flask development skills and regex proficiency.

THANK YOU

INNOMATICS RESEARCH LABS