INNOMATICS®
RESEARCH LABS

INNOVATION. AUTOMATION. ANALYTICS

Project On

Creating Web Application for Regex Pattern Matching and Email Validation

S Dinesh Kumar

4th March 2024

Objective:

The objective of this web development project is to create a user-friendly and efficient web application that serves as a tool for regex pattern matching and email validation. The application aims to provide a simple yet powerful interface, allowing users to input a test string and a custom regex pattern. The system will perform regex matching on the input, displaying all matches found. Additionally, the application will include a feature to validate email addresses, checking if they confirm to valid email formats. The primary goal is to empower users with a convenient means to validate strings against custom regex patterns and ensure the adherence of email addresses to proper formats.

DEPLOYMENT:

AWS EC2: http://51.20.5.207:5000/

HTML Sections:

I created three html file for my web application,which act as a main skeleton for my website, they are

- Base
- Index
- Results
- Framework: Flask

Base.html file:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}Regex Matcher{% endblock %}</title>
    <link rel="stylesheet" href="static\css\style.css">
</head>
<body>
    <header>
        <nav>
            <a href="webapp\Templates\index.html">Home</a>
            <a href="webapp\Templates\results.html">Results</a>
        </nav>
    </header>

    <div id="content">
        <aside class="sidebar left-sidebar">
        </aside>

        <main>
            {% block content %}{% endblock %}
        </main>

        <aside class="sidebar right-sidebar">
        </aside>
    </div>

    <footer>
        &copy; 2024 Innomatics Research Labs
    </footer>
</body>
</html>
```

Document Type Declaration (DOCTYPE):
DOCTYPE: Tells the browser which version of HTML is being used.

HTML Structure:

<html>: The beginning of the webpage.
<head>: Contains important information about the page, like its title and styles.
<body>: Where the main content of the webpage goes.
Head Section:
Contains meta tags for character encoding and viewport settings.
Sets the title of the webpage dynamically.
Links an external stylesheet for styling.
Body Section:
Contains the main content of the webpage, including header, main content area, and footer.
Header (Navbar):
Contains navigation links to different pages of the website.
Main Content Area:
Divided into left and right sidebars and the main content section.
The main content section has a placeholder for dynamic content.
Footer:
Displays copyright information.
Closing Tags:
</body> and </html>: Indicate the end of the webpage's content and structure.

Index.html file:

This Jinja template extends a base template ('base.html') and defines specific content for a page titled "Regex Matcher." The page includes an h1 heading, displays flashed messages, and features two forms—one for submitting a test string and regex pattern, and another for validating an email address. The structure and styling are inherited from the 'base.html' template.

```
{% extends 'base.html' %}
{% block title %}Regex Matcher{% endblock %}
{% block content %}
    <h1>Regex Matcher</h1>

    {% with messages = get_flashed_messages() %}
        {% if messages %}
            <ul class="messages">
                {% for message in messages %}
                    <li class="{{ message|lower }}">{{ message }}</li>
                {% endfor %}
            </ul>
        {% endif %}
    {% endwith %}

    <form action="{{ url_for('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">Submit</button>
    </form>

    <form action="{{ url_for('validate_email') }}" method="post">
        <label for="email">Validate Email:</label>
        <input type="text" id="email" name="email" required>
        <button type="submit">Validate</button>
    </form>
{% endblock %}
```

Results.html:

```
{% extends 'base.html' %}

{% block title %}Regex Matcher - Results{% endblock %}

{% block content %}
    <h1>Regex Matcher - Results</h1>

    <p><strong>Test String:</strong> {{ test_string }}</p>
    <p><strong>Regex Pattern:</strong> {{ regex_pattern }}</p>

    {% if matches %}
        <h2>Matches:</h2>
        <ul>
            {% for match in matches %}
                <li>{{ match }}</li>
            {% endfor %}
        </ul>
    {% else %}
        <p>No matches found.</p>
    {% endif %}

    {% if email_validation %}
        <p>Email Validation Status: {{ email_validation }}</p>
    {% endif %}

    <a href="{{ url_for('home') }}" class="back-button">Back to Home</a>
{% endblock %}
```

This Jinja template extends a base template ('base.html') and defines specific content for a page titled "Regex Matcher - Results." The page displays the test string and regex pattern used, along with the following:

Operations:

If there are matches:

- A heading "Matches" is displayed, followed by an unordered list of individual match results.

If there are no matches:

- A message "No matches found" is displayed.

If email validation is performed:

- The status of email validation is displayed.

At the end of the page, there's a link labeled "Back to Home," which redirects to the 'home' endpoint when clicked. The structure and styling are inherited from the 'base.html' template.

CSS Styling:

General Styling:

```
body {
    font-family: 'Arial', sans-serif;
    margin: 0;
    padding: 0;
    background-color: #000000;
    color: #333;
}
```

- <body> element represents the content of an HTML document, It contains all the information that is visible on the web page, including text, images, links, and other elements. I added some styles for the clear visuals.

- **font-family: 'Arial', sans-serif;**
  I used this because, this line tells the browser what kind of font to use for the text on webpage. First,It prefers to use 'Arial' if it's available,If not, it will use a generic sans-serif font

- **margin: 0:**
  This property removes any default margin around the <body> element. Margins are spacing outside the border of an element.

- **padding: 0:**
  Similar to the margin, this property removes any default padding around the <body> element. Padding is the spacing inside the border of an element.

- **background-color: #000000:**
  This sets the background color of the <body> element to black (#000000 in hexadecimal). You can replace this with any other valid color code or name as your wish.

- **color: #333:**
  This sets the text color of the <body> element to a dark gray (#333 in hexadecimal).

Content Styling:

```
#content {
    max-width: 800px;
    margin: 50px auto;
    padding: 20px;
    background-color: #333;
    color: #fff;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    border-radius: 8px;
    animation: fadeIn 0.5s ease-in-out;
}
```

- **Content:**
  "content" generally refers to the substance of information on a webpage. It includes text, images, videos, or any other elements that convey the primary information or message on the page.

- **max-width: 800px;**
  This sets the maximum width of the #content element to 800 pixels. It ensures that the content doesn't stretch too wide on larger screens.

- **margin: 50px auto;**

  This property sets the margin around the #content element.
  50px is the top and bottom margin, and auto horizontally centers the element by automatically distributing the remaining space on the left and right.

- **padding: 20px;**

  Padding adds space inside the #content element.
  This line specifies 20 pixels of padding on all sides.
  background-color: #333;

- **background-color: #333**

  This sets the background color of the #content element to a dark gray color (#333 in hexadecimal).

- **color: #fff;**

  This sets the text color inside the #content element to white (#fff in hexadecimal).

- **box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);**

  This creates a subtitle box shadow effect around the #content element.It's a shadow that is 10 pixels in size, with a slight transparency (0.1) and a black color.

- **border-radius: 8px;**

  This gives the #content element rounded corners.
  The value 8px determines the radius of the corners.
  animation: fadeIn 0.5s ease-in-out;
  This applies an animation to the #content element.
  It uses an animation called fadeIn that lasts for 0.5 seconds, easing in and out for a smoother effect.

- **Header (h1) Styling:**

```
h1 {
    color: #ffc107;
}
```

  Giving headers with a standout yellow color.

Form Styling:

```css
form {
    margin-top: 20px;
}
```

form {margin-top: 20px;}:

I Added a top margin of 20 pixels to all <form> elements on the page.

Label Styling:

```css
label {
    display: block;
    margin-bottom: 5px;
    font-weight: bold;
    color: #fff;
}
```

Styles for <label> elements, making them block-level, adding bottom margin, making the text bold, and setting the text color to white (#fff).

Input Styling:

```css
input[type="text"] {
    width: 100%;
    padding: 8px;
    margin-bottom: 15px;
    box-sizing: border-box;
    border: 1px solid #ccc;
    border-radius: 4px;
    background-color: #444;
    color: #fff;
}
```

Styles for <input> elements with the type attribute set to "text".

Button Styling:

```
button {
    background-color: #ffc107;
    color: #333;
    padding: 10px 15px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
```

- Setting the background color to yellow (#ffc107) and text color to dark gray.
- Adding padding inside the button.
- Removing the default button border and adds rounded corners.
- Changing the cursor to a pointer on hover.
- Styles for buttons with the class "back-button".Similar to the main button but with slight differences.

```
.back-button {
    display: inline-block;
    margin-top: 15px;
    padding: 10px 15px;
    background-color: #ffc107;
    color: #333;
    text-decoration: none;
    border-radius: 4px;
    cursor: pointer;
}

.back-button:hover {
    background-color: #ffdb58;
}
button:hover {
    background-color: #ffdb58;
}
```

List Styling:

```
ul {
    list-style: none;
    padding: 0;
}
li {
    margin-bottom: 5px;
}
```

Removed default list styles and padding for unordered lists.

Added a bottom margin to list items.

Messages Styling:

```
.messages {
    margin-top: 15px;
}
.messages li {
    padding: 10px;
    border-radius: 4px;
}
.messages .error {
    background-color: #f8d7da;
    color: #721c24;
}
.messages .success {
    background-color: #d4edda;
    color: #155724;
}
```

Added top margin to a container with the class "messages".

Styles for list items within the messages container.
padding and rounded corners to list items.
Added Styles for success messages.

Fade-in Animation:

```css
/* Simple fade-in animation */
@keyframes fadeIn {
    from {
        opacity: 0;
    }
    to {
        opacity: 1;
    }
}
nav {
    background-color: #444;
    color: #fff;
    padding: 10px;
}
nav a {
    color: #fff;
    text-decoration: none;
    margin: 0 10px;
}
```

- Defining a simple fade-in animation using keyframes.
- Navigation Bar Styling
- Styling for navigation links.

Footer Styling:

```css
footer {
    background-color: #444;
    color: #fbfbfb;
    text-align: center;
    padding: 10px;
    position: fixed;
    bottom: 0;
    width: 100%;
}
```

Fixed the footer at the bottom of the viewport and added some styles.

Framework:

Flask

```
from flask import Flask, render_template, request, redirect, url_for, flash
import re

app = Flask(__name__)
app.secret_key = ‘Your - Key’
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/results', methods=['POST'])
def results():
    if request.method == 'POST':
        test_string = request.form['test_string']
        regex_pattern = request.form['regex_pattern']
        try:
            re.compile(regex_pattern)
        except re.error:
            flash('Invalid regex pattern. Please enter a valid regex.',
'error')
            return redirect(url_for('home'))
        matched_strings = re.findall(regex_pattern, test_string)
        email_validation = 'Valid' if re.match(r'^[\w\.-]+@[a-zA-Z\d\.-]+\.[a-
zA-Z]{2,}$', test_string) else 'Invalid'
        return render_template('results.html', test_string=test_string,
regex_pattern=regex_pattern, matches=matched_strings,
email_validation=email_validation)
@app.route('/validate-email', methods=['POST'])
def validate_email():
    if request.method == 'POST':
        email = request.form['email']

        email_regex = r'^[\w\.-]+@[a-zA-Z\d\.-]+\.[a-zA-Z]{2,}$'

        if re.match(email_regex, email):
            flash('Valid email address!', 'success')
        else:
            flash('Invalid email address. Please enter a valid email.',
'error')
        return redirect(url_for('home'))
if __name__ == '__main__':
    app.run(debug=True,host='0.0.0.0',port=5000)
```

This Flask app has two main functionalities: regex matching and email validation. It uses two HTML templates ('index.html' and 'results.html') to display content, and it incorporates the Flask flash feature for displaying temporary messages. The app runs in debug mode on localhost.

This Flask web application enables users to test regular expressions and validate email addresses. It features two routes:

Home Route (/):

Renders the 'index.html' template for the home page.

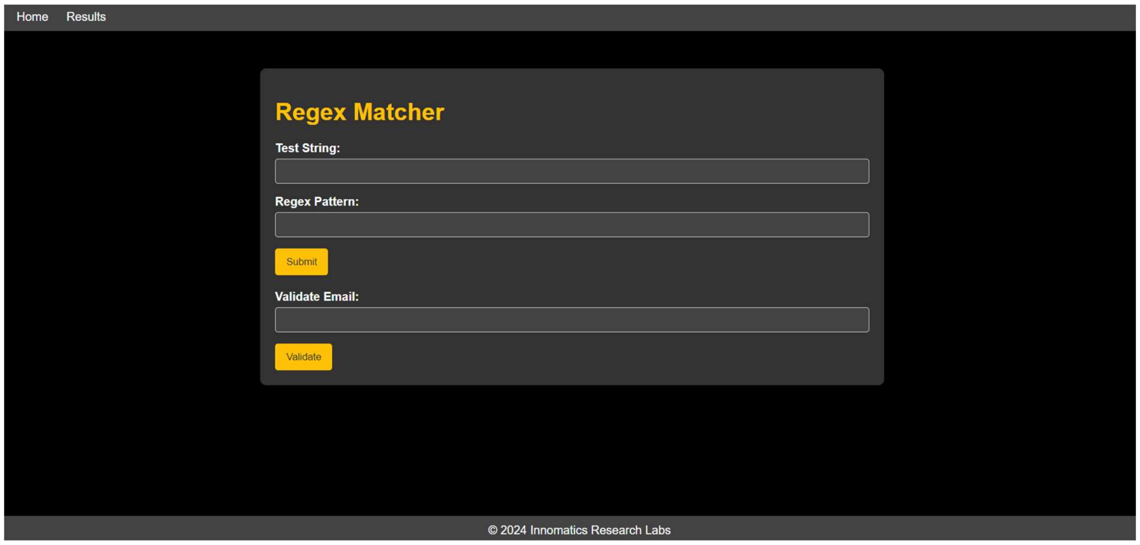
Results Route (/results):

Handles form submissions from the 'results' page.
Retrieves and validates a test string and regex pattern.
Finds matches in the test string using the regex pattern.
Performs email validation.
Renders the 'results.html' template with relevant information.

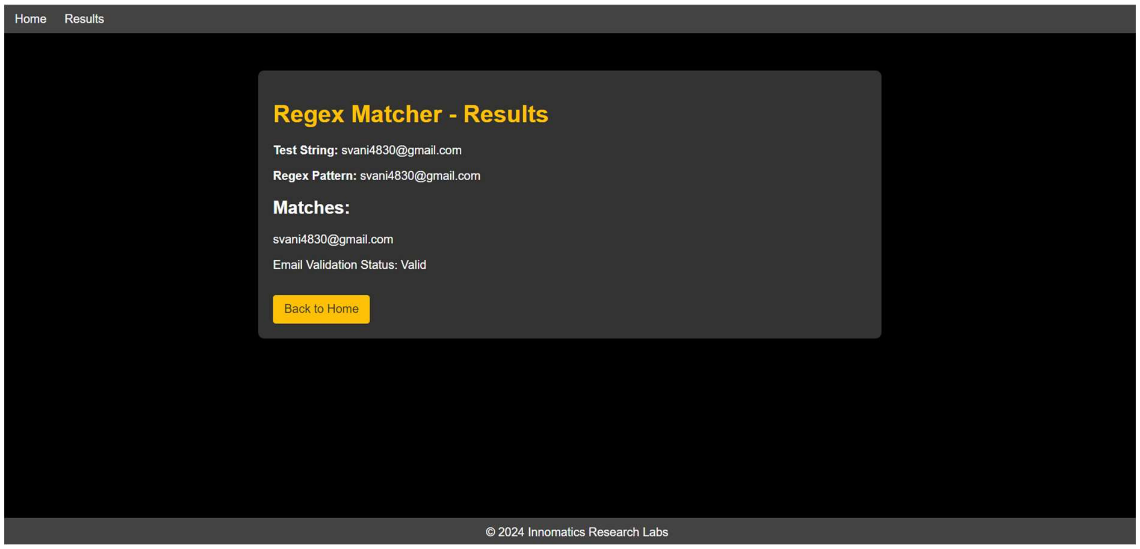
Validate Email Route (/validate-email):

Handles form submissions from the 'validate-email' page.
Validates an entered email address using a regex pattern.
Flashes a success or error message based on the validation result.
Redirects back to the home page.
The application runs in debug mode on localhost and utilizes two HTML templates for content rendering
('index.html' and 'results.html').

UI:



Results:



Test String: svani4830@gmail.com

Regex Pattern: svani4830@gmail.com

Matches:

svani4830@gmail.com

Email Validation Status: Valid

Thank You