Regex Matching Web Application Development Project Report

Project Overview

The primary objective of this project was to develop a web application that replicates the core functionality of the website regex101.com. The application allows users to input a test string and a regular expression (regex) and displays all matches found within the test string. Additionally, the application includes a feature to validate email addresses. As a bonus, the project was deployed on AWS Cloud to demonstrate the deployment process of a Flask web application in a cloud environment.

Project Steps and Execution

Step 1: Project Setup and Environment Preparation

The project began with creating a dedicated directory to organize all files related to the web application. This directory was created to maintain a clean and structured environment conducive to development.

- **Action Taken**: Created a new directory named PROJECT and navigated into it.
- **Objective**: To establish a dedicated workspace for the project files and virtual environment.

Step 2: Virtual Environment and Flask Installation

To ensure a controlled development environment and avoid potential conflicts with other projects, a Python virtual environment was set up. Flask, a lightweight Python web framework, was installed within this virtual environment to build the web application.

- **Action Taken**:
  - Set up a virtual environment using the command python3 -m venv venv.
  - Activated the virtual environment with source venv/bin/activate.
  - Installed Flask using pip: pip install Flask.
- **Objective**: To create a virtual environment for the project and install the necessary dependencies without affecting the global Python installation.

Step 3: Initialization of Flask Application

The core of the application was built using Flask. A new Python file named app.py was created, where the Flask application instance was initialized. The initial route was set up to render a home page that allows users to input a regex pattern and a test string.

- **Action Taken**:
  - Created app.py and initialized a Flask app instance with app = Flask(__name__).
  - Defined the home route (/) to render the main input form.
- **Objective**: To set up the foundation of the web application and provide users with a starting point to input data for regex matching.

Step 4: Designing the HTML Template
An essential part of the project was creating an intuitive user interface. A directory named templates was created to store HTML files. The main HTML file, index.html, was designed with a form that included input fields for the regex pattern and the test string, along with a submit button.

- **Action Taken**:
  - Created a templates directory.
  - Developed index.html with a user-friendly form layout.
- **Objective**: To provide an interactive front-end interface for users to input data and view results.

Step 5: Handling Form Submissions and Performing Regex Matching
A new route (/results) was defined in app.py to handle form submissions. This route captured the user input, performed regex matching using Python's re module, and stored the matched strings.

- **Action Taken**:
  - Defined the /results route to process POST requests.
  - Extracted input data and performed regex matching with re.findall().
  - Stored matched strings in a list.
- **Objective**: To process user input, perform regex operations, and store results for display.

Step 6: Rendering Results to the User
The matched results were passed back to the HTML template (index.html) and displayed to the user. The template was modified to show the matched strings below the input form, allowing users to see the outcome of their regex patterns.

- **Action Taken**:
  - Modified index.html to display results dynamically.
  - Integrated result display logic into the Flask app.
- **Objective**: To provide immediate feedback to users on their regex inputs by displaying all matches found.

Step 7: Application Testing
The web application was tested locally to ensure all functionalities worked as expected. Various test strings and regex patterns were input to validate the application's core functionality and robustness.

- **Action Taken**:
  - Ran the Flask application locally using python app.py.
  - Tested various regex patterns and strings at http://localhost:port.
- **Objective**: To verify the correct implementation of regex matching and validate user inputs.

Step 8: Email Validation Feature Implementation

An additional route (/validate_email) was implemented to validate whether a given email address is formatted correctly. This feature was designed to demonstrate the application's flexibility in handling different types of input validations.

- **Action Taken**:
  - Added a new route /validate_email in app.py.
  - Used a regex pattern to validate email addresses.
  - Rendered validation results in validate_email.html.
- **Objective**: To enhance the application with an email validation feature and showcase the versatility of regex in different contexts.

Step 9: Deployment on AWS Cloud

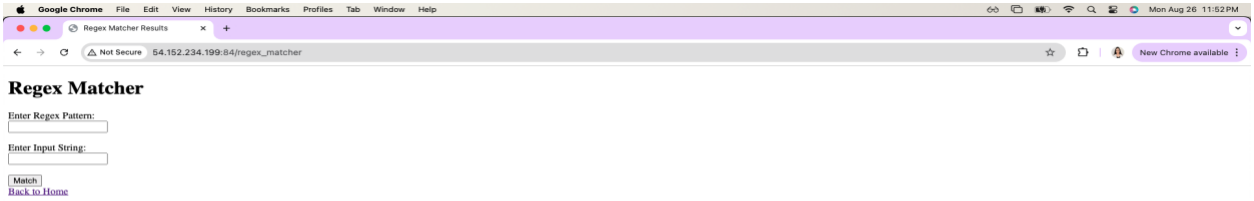
To make the application accessible over the internet, it was deployed on an AWS EC2 instance. This step involved setting up the EC2 environment, configuring security groups, and running the Flask app on a publicly accessible IP address.

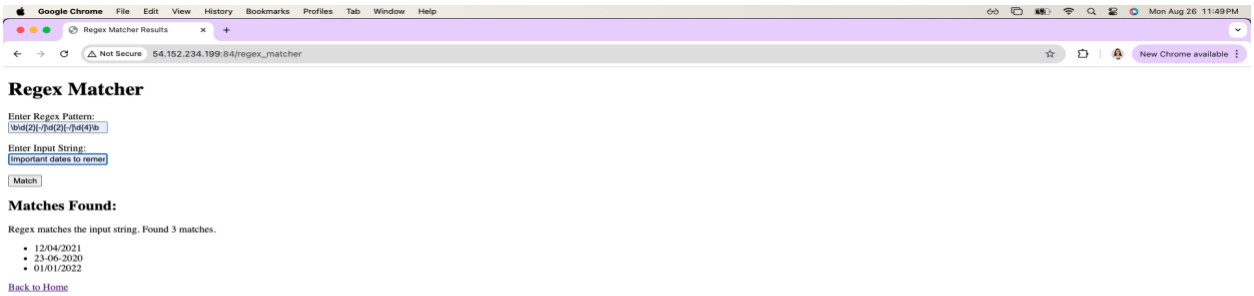
- **Action Taken**:
  - Launched an AWS EC2 instance and configured it with the necessary security settings.
  - Installed required software (Python, Flask) on the instance.
  - Deployed the Flask application using a production-ready WSGI server.
- **Objective**: To demonstrate the deployment process of a Flask application on a cloud platform, ensuring accessibility and scalability.

Conclusion

The Regex Matching Web Application project successfully achieved its objectives by creating a functional and user-friendly web application that mimics the core functionality of regex101.com. The added email validation feature demonstrated the versatility of the app, while the AWS deployment showcased the practical steps involved in deploying a web application in a cloud environment. The project provided valuable hands-on experience in web development, regex, and cloud deployment, making it a comprehensive exercise in building and deploying a real-world web application.

Outputs for Above:





Regex Matcher

Enter Regex Pattern:
//d

Enter Input String:
This number is 123

Match

Matches Found:

Regex matches the input string. Found 0 matches.

Back to Home

Email Validator

Enter Email:

Validate Email

Back to Home

Email Validator

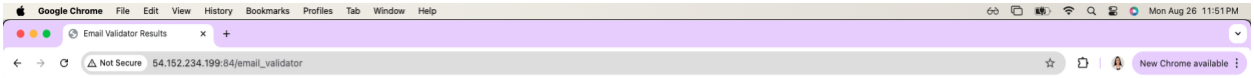
Enter Email:
kiranbele.1110@gmail.cc

Validate Email

Validation Result:

'kiranbele.1110@gmail.com' is a valid email address.

Back to Home



Email Validator

Enter Email:
unmeshmali.gmailcom

Validate Email

Validation Result:

'unmeshmali.gmailcom' is not a valid email address.

Back to Home