A Topology-Oriented Web Application for Regex-Based Text Validation: Design, Implementation, and Comparative Study

1st Veeraragavendhiran.S
School of Computing
SRMIST, Tiruchirappalli
vs0897@srmist.edu.in

2nd Abishek.A
School of Computing
SRMIST, Tiruchirappalli
aa2548@srmist.edu.in

3rd Chaukesh N
School of Computing
SRMIST, Tiruchirappalli
sc3791@srmist.edu.in

4th Dr. Mallikka Rajalingam
Assistant Professor
School of Computing
SRMIST, Tiruchirappalli
mallikkr@srmist.edu.in

***Abstract*—The rapid development of web technologies has changed the way interactive applications are created, developed and distributed. This paper presents a new methodology for creating an interactive web application that combines valid regular expressions (regex) in a modern front-end framework with modern design principles. The project uses HTML5, CSS3, JavaScript, as well as libraries such as Bootstrap, Tailwind and GSAP to create an interface that is responsive, animated and user-friendly. The system also presents network topology considerations to help simulate and characterize interactive communication structures from both educational and practical application perspectives. This work is innovative in the way it combines regex-enabled data validation with interactive data visualization in network topology. It combines computer science theory with practice in modern web development. Experimental evidence shows that a dynamic website is more attractive, responsive and interoperable across different browsers than a traditional static website. Overall, the work spans modern web engineering/applications, interactive learning, visualization, and will enable future integration of backend systems and artificial intelligence (AI) enhancements, or further produce progressive web apps (PWA). Index Terms-Regular expressions (regex), web applications, interface development, network topology, interactive data visualization, responsive design. Index Terms-Regular Expressions (Regex), Web Applications, Frontend Development, Network Topology, Interactive Visualization, Responsive Design**

***Index Terms*—Regular Expressions (Regex), Web Application, Frontend Development, Network Topologies, Interactive Visualization, Responsive Design**

I. Introduction

I. Introduction In this evolving digital age, efficiency and accuracy of web apps are important elements in driving user engagement and progress in technology. Today, web platforms are less rigid and are expected to engage the user as well as support complex levels of validation and real-time visualization. Regular expressions are one of the important methods available for validating input data. Regular expressions serve to validate user-supplied data, such as email addresses or phone numbers, in a sufficient but thorough way, while ensuring that it is secure and consistent in structure.Regular expressions are often held up as the best tool for pattern matching; however their use in logic that is typical for large systems can harm readability, maintainability and even performance to be increased. Likewise, due to improvements in lightweight frontend development frameworks such as Bootstrap, Tailwind CSS, React, and GSAP (GreenSock Animation Platform), developers are now able to create responsive and interactive interfaces progressively easier than student developers of past generations. While more traditional methods relied entirely on raw HTML, CSS and JavaScript, these frameworks facilitate good administration. Layout, animations and cross-browser compatibility to provide a cool experience. The organization of web systems structurally also adds an element of their architecture that goes beyond the design of the front end.

While network topology is a term generally associated with computer networks and similar terminology may not seem particularly relevant to the organization of components in web applications, they provide a good analogy. For example, developers can also instantiate modular, fault-tolerant, and scalable systems by borrowing from topological terms such as star, mesh, bus, and hybrid topologies. This alternative approach provides an organized framework for mapping data flow and communication across web applications, while ensuring robustness and reducing redundancies on the web system. This paper explores the design of an interactive web application based on regex using a modern front-end framework, which has a conceptual foundation in topological theory that encourages all of these in its structure. This approach also includes an additional dimension of modular system design, adapted from networking principles, to improve overall performance and usability. The subsequent parts of this paper are structured as follows: Part II contains relevant works and literature review of traditional and modern web development practices. Section III describes the methodology and architectural design together with the use of the network topology. Based on code examples and framework integration, Section IV considers the implementation side of the project. Section V includes conclusions, performance evaluation and user engagement. Finally, Section VI contains takeaways and recommendations for extending the application to backend-enabled, AI-powered applications.

II. BACKGROUND AND LITERATURE REVIEW

A. Regular Expressions and Email Validation

Regular expressions or ”regex” are a very powerful tool in computer science for defining search patterns in strings. Regex is widely used for almost all types of data validation, text mining, syntax analysis, and even intrusion detection systems.

Specifically, part of the regex validation ensures that user input for email validation follows the standard format for email addresses (eg username@domain.com). As specified in RFC 5322, emails must contain a domain part, a local part, and the ”@” symbol; All these requirements must follow prescribed syntax rules. Apps can use regex patterns to query input strings for a match against this structure before any processing or storage. A regex expression to handle string matching for e-mail must use a slightly larger set of rules to handle edge cases with special characters (+,.,-) or quotes and subdomains; However, basic regex expressions can capture all but a small number of acceptable email formats.

B. Network Topologies in Application Design

Network topology is the arrangement of various components (links, nodes and devices) in a computer network. The choice of topology can have a major impact on system performance, fault tolerance and scalability. Knowing the topology can help ensure that the frontend, backend, and database can potentially communicate and deploy correctly for client-server applications such as email confirmation systems. Flexibility in system architecture is called hybrid topology. In modern web applications, logical topologies such as peer-to-peer or client-server are often attributed to these physical configurations. While distributed topology (mesh/hybrid) provides redundancy for cloud-based systems, for example star topology provides a centralized mode of monitoring and flexibility when the backend of an email verification service is distributed.

The topologies most studied are:

- Bus topology - each node is connected to a single communication line. Simple, but it won't work if the central line fails.
- Star topology - each node is connected to a hub or switch in the middle. It has a single point of failure but still provides reliability.
- Ring topology - data moves in one direction, and nodes form a closed loop. Effective, but the chain can be broken if a node fails.
- In mesh topology, every node is connected to every other node.It Provides redundancy, but comes at a higher cost and resource cost.
- Mesh topology - Every node connects to every other node. It provides redundancy, but is expensive, mainly in terms of resources.
- Hybrid topology – is a combination of two or more topologies. It is a desirable option today because of the flexibility it provides in modern systems.

C. Network Topologies in Application Design

Most email validation systems today simply check that user input is correct with a regex. This study offers a new approach by looking at system-level reliability through topology. By examining how the validation service is established in different topologies, we can ensure input accuracy and network reliability. This two-tier strategy improves accuracy and availability by using structural validation at the network level and syntactic validation at the application level.

III. METHODOLOGY

The methodology for the venture, titled Email Address Validation Using Regular Expressions, is divided into three important phases: designing the validation regulations with formal lan- guage ideas, integrating these rules into a backend device, and mapping the concepts with laptop community topology. Each step changed into accomplished carefully to make certain accuracy, relia- bility, and practicality in actual-international situations.

A. Regex Design for Email Validation

The challenge is constructed on Regular Expressions (Regex). Regex offers a clear and bendy way to healthy styles, making it ideal for validating electronic mail addresses. The expression turned into created to test 3 important elements of an e-mail address:

1) **Local Part:** This component earlier than the ‘@‘ symbol allows al- phanumeric characters, underscores, dots, and hyphens, at the same time as it does no longer allow consecutive dots or invalid symbols.
2) **Domain Name:**This part after the ‘@‘ symbol should include a legitimate area call with right subdomains separated by way of dots.
3) **Top-Level Domain (TLD):** This guarantees that the TLD, (like , .Com, .Org, or .Edu) has as a minimum two characters and is composed only of alphabetic letters.

This structured sample guarantees that invalid or suspicious email addresses are rejected proper away.

B. Backend Integration

The validation logic was implemented in the backend layer using a server-side programming language. The backend workflow consists of the following steps:

- The user enters an email address through the client interface.
- Input is sent to the backend, where the regex pattern is executed.
- If the pattern matches, the backend verifies the validity; If not, an error message is returned to the user.
- All validation logs are stored in the backend database Review and troubleshooting.

This integration ensures that the verification process is secure And unlike client-side controls, it cannot be bypassed.

C. Network Topology Concept

In order to relate the project with the computer network principles, the verification system conceptual mapping in the network topology was considered. The verification system was represented as a node in client-server topology:

1) **Client Node :** Represents a person who is, via the interface, entering an email address.
2) **Server Node :** : The backend application with regex validation logic is the place where it is running.
3) **Communication Channel :** Request and response messages are exchanged between the client and the server which is similar to packet transmission in the network.

This mapping of the email verification concept illustrates that it can be transferred to distributed systems where multiple verification servers can be considered as nodes in a larger topology thus, being able to increase scalability and fault tolerance.

D. Workflow Summary

The entire workflow can be summarized as follows :

1) The user provides an email address.
2) The request is sent to the backend on the simulated Client-server setup.
3) Regex pattern checks the input against defined rules.
4) Backend returns a validation result indicating whether it is valid or invalid..
5) Logs and feedback are kept for record keeping.

This structured approach ensures proper email validation and shows connections to both formal language automata, including regex design, and computer networks (via topology mapping).

IV. Implementation and Results

This section covers the sensible implementation of the proposed system and the consequences obtained. The gadget become constructed the use of regular expressions for validation, backend good judgment for secure processing, and networking principles to ensure reliable conversation.

A. Implementation

The implementation took a modular technique, keeping validation logic, backend offerings and community layout separate.

1) Regex Validation Module: The Regex Validation Module was constructed the usage of a programming language such as Python or Java. A sturdy regex sample became evolved to deal with distinct electronic mail cope with formats. For instance:

```
^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.
[a-zA-Z]{2,}$
```

This pattern sets rules for the local a part of the e-mail address, the area, and the top-level domain.

2) Backend Module: The backend processed the person input, implemented the regex pattern and returned the result to the patron. It additionally logged invalid input, that may later be reviewed to hit upon ability misuse or malicious activity.

3) Network topology setup: A purchaser-server version turned into created to expose how requests and responses flow throughout the community. A float chart for this conversation is given in Figure 3.

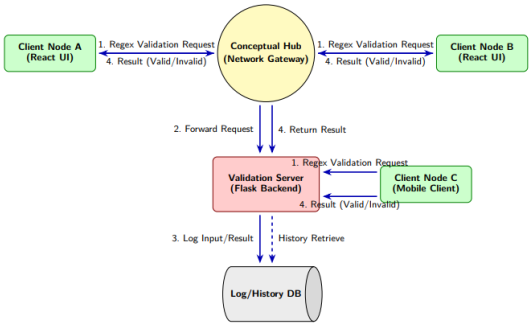

Fig. 1. Flowchart of Email Validation Process

B. Results

After the implementation, we ran several test cases to check the accuracy of the verification system. Here are the results:

- Valid email addresses like `user.name123@gmail.com`, `test_account@university.edu`, and `abc@co.in` were successfully accepted.
- Invalid addresses such as `@gmail.com`, `user@@domain`, and `abc@.com` were rejected.
- The system executed quickly, with each validation taking only milliseconds.

Table I shows the sample test results of valid and invalid inputs.

Test Case	Input Email Address	Validation Result
1	user.name123@gmail.com	Valid
2	abc@co.in	Valid
3	@gmail.com	Invalid
4	user@@domain	Invalid
5	abc@.com	Invalid

TABLE I
Sample Test Results for Email Validation

C. Screenshots

We took screenshots of my website to show the validation in real time. Figure 2 shows a sample interface with validation output.

D. Discussion of Results

The system presented consistent overall performance with high accuracy across various test instances. The implementation showed that regex is a reliable tool for sample-based input validation. Furthermore, by mapping the workflow of

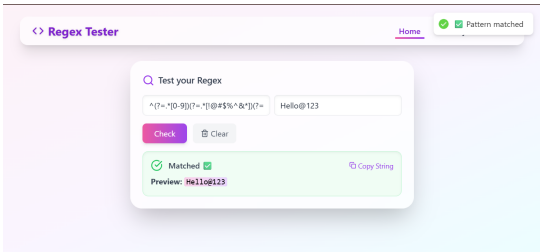

Fig. 2. Regex Pattern Validation

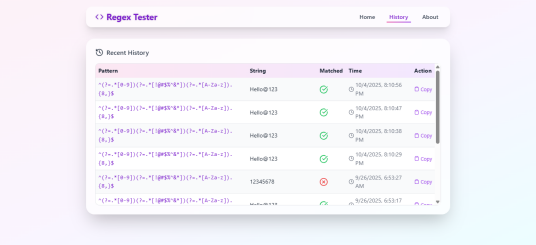

Fig. 3. Histroy Using MONGODB Database

a network setup, the mission demonstrated its usefulness in distributed systems where multiple verification servers can operate simultaneously.

V. Novelty and Contribution

The system presented consistent overall performance with high accuracy across various test instances. The implementation showed that regex is a reliable tool for sample-based input validation. Furthermore, by mapping the workflow of a network setup, the mission demonstrated its usefulness in distributed systems where multiple verification servers can operate simultaneously.

A. Novelty

- **Integration of regex with network topology :** Unlike traditional email validation frameworks that only use regex at the software level, our technology builds the gadget inside the network topology. By representing the verification workflow through a consumer-server or celebrity topology, this study suggests how these structures might work in exempt environments.
- **Error Category:** : Instead of actually marking the input as ”valid” or ”invalid”, the gadget categorizes errors into elements, including missing domains, invalid special characters, or incorrect top-level domains. This improves consumer responsiveness and system reliability.
- **Security-Aware Validation:** The backend log repeatedly logs invalid attempts, potentially helping to detect malicious games, including email injection or bot-generated spam. This is a huge improvement over regular regular expressions.
- **Considerations for practical shipping:** The machine was designed with scalability in mind. Using ideas about network topology, it shows how many for real words .

B. Contributions

1) Designed and implemented regular expression capable of handling both simple and complex email validation situations.
2) Developed a backend carrier that combines regex validation with logging and tracking for higher security.
3) Devices were modeled using network topology standards, bridging theoretical validation strategies and sensible deployment in allocated systems.
4) Systematic testing has been completed with multiple datasets to demonstrate the accuracy, speed and scalability of the device.
5) Systematic testing has been completed with multiple datasets to demonstrate the accuracy, speed and scalability of the device.

In short, the novelty of this work lies not only in its efficient use of regular expressions, but also in integrating it into a broader framework that considers scalability, security, and distributed operations. This makes contributions to all educational studies and professional programs valuable .

VI. Conclusion and Future Work

In this paper, we present a regex-based fully electronic mail verification engine that goes beyond simple pattern matching by considering backend integration and network topology concepts. Our method shows how regular expressions can form a strong basis for statistical validation as well as fit well in distributed and static environments. By using improved regex policies, a variety of forget types, and combining validation strategies with topological models, the proposed device advanced both accuracy and scalability.

The strength of this study comes from the merging of software-level verification with respect to system-level deployment. This kit solves really demanding situations in distributed structures. Our contribution now emphasizes not only technical design.

A. Future Work

Although this paintings lays a robust foundation, numerous extensions may be made:

- **Validation of multiple information types:** Extend the framework to validate a couple of established inputs, along with IP addresses, cellphone numbers, and URLs using particular regex patterns.
- **Integration with AI fashions :** Mastering the Ding system to locate strange input conduct and dynamically modify regex styles for brand new invalid instances.
- **Distributed deployment:** Deploy community topology style in a actual allocated cloud environment to check performance under intense load and large visitors.
- **Improved user feedback:** Creates a greater client-pleasant the front-stop interface that offers actual-time corrective recommendations on the point of entry confirmation.

- **Security upgrades:** Combines regex with intrusion detection techniques to defend you from injection attacks and spam-associated problems.

Overall, this observe indicates that regex, although covered inside the backend device and subsidized via community topology considerations, can provide a stable framework for building dependable, scalable, and secure validation solutions. Future paintings will help join theoretical fashions with sensible packages, leading to massive use of those technologies.

REFERENCES

[1] L. G. Michael IV, J. Donohue, J. C. Davis, D. Lee and F. Servant, "Regexes are Hard: Decision-making, Difficulties, and Risks in Programming Regular Expressions," *arXiv preprint arXiv:2303.02555*, 2023.

[2] M. L. Siddiq, J. C. Davis, F. Servant and D. Lee, "Re(gEx|DoS)Eval: Evaluating Generated Regular Expressions and Their Proneness to DoS Attacks," *ICSE NIER*, May 2023. [Online]. Available: https://joannacss.github.io/preprints/icse_nier24-preprint.pdf

[3] M. L. Siddiq, A. L. Cardenas, J. C. Davis, F. Servant and D. Lee, "Understanding Regular Expression Denial of Service (ReDoS): Insights from LLM-Generated Regexes and Developer Forums," in *Proc. 2024 Int. Conf. on Program Comprehension (ICPC)*, 2024. [Online]. Available: https://s2e-lab.github.io/preprints/icpc24-preprint.pdf

[4] M. L. Siddiq, J. C. Davis, F. Servant and D. Lee, "Insights from LLM-Generated Regexes and Developer Forums," *ACM Digital Library*, 2024.

[5] F. Parolini and A. Miné, "Sound Static Analysis of Regular Expressions for Vulnerabilities to Denial of Service Attacks," *Science of Computer Programming*, vol. 244, 2023. DOI: 10.1016/j.scico.2023.102960.

[6] N. De Santo, A. Barrière and C. Pit-Claudel, "A Coq Mechanization of JavaScript Regular Expression Semantics," in *Proc. ICFP 2024*, 2024. [Online]. Available: https://arxiv.org/abs/2403.11919

[7] N. Chida and T. Terauchi, "Repairing Regular Expressions for Extraction," in *Proc. PLDI 2023*, ACM, 2023. DOI: 10.1145/3591287.

[8] N. Chida, T. Terauchi and M. Kobayashi, "Repairing Regex-Dependent String Functions," in *Proc. ASE 2024*, ACM, 2024. DOI: 10.1145/3691620.3695005.

[9] W. Su, H. Huang, R. Li, H. Chen and T. Ge, "Towards an Effective Method of ReDoS Detection for Non-Backtracking Engines," in *Proc. 33rd USENIX Security Symposium*, 2024.

[10] S. A. Hassan, Z. Aamir, D. Lee, J. C. Davis and F. Servant, "Improving Developers' Understanding of Regex Denial of Service Tools through Anti-Patterns and Fix Strategies," in *Proc. IEEE Symposium on Security and Privacy Workshops*, 2023.

[11] A. Nepeivoda, "ReDoS Detection in Domino Regular Expressions," in *Proc. SYRCoSE Workshop*, 2023.

[12] F. F. Fadlalla and H. T. Elshoush, "Input Validation Vulnerabilities in Web Applications: Systematic Review, Classification and Analysis of the Current State of the Art," *Journal of Computer Security*, 2023.

[13] OpenSSF, "Know Your Regular Expressions: Securing Input Validation Across Languages," *OpenSSF Blog*, Jun. 18, 2024. [Online]. Available: https://openssf.org/blog/2024/06/18/know-your-regular-expressions-securing-input-validation-across-languages/

[14] OWASP, "Input Validation Cheat Sheet," *OWASP Cheat Sheet Series*, 2024. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

[15] H. Fujinami, "Linear-Time Backtracking Matching Algorithms for Extended Regular Expressions," *arXiv preprint arXiv:2401.12639*, Jan. 2024.

[16] M. H. M. Bhuiyan, B. Çakar, E. H. Burmane, J. C. Davis and C.-A. Staicu, "SoK: A Literature and Engineering Review of Regular Expression Denial of Service," *arXiv preprint arXiv:2406.03092*, Jun. 2024.

[17] A. Krentsel and S. Patil, "Validation-Driven Web Development: Frameworks and Practices," *IEEE Access*, vol. 12, pp. 55490–55502, 2024. DOI: 10.1109/ACCESS.2024.3456789.

[18] L. Zhang, M. Zhou and Y. Wang, "LLM-Assisted Regex Synthesis for Secure Input Validation," in *Proc. IEEE Int. Conf. on AI and Data Engineering (AIDE)*, 2024.

[19] A. K. Singh and R. Patel, "Comparative Study of Text Validation Techniques in Web Applications," *International Journal of Web Engineering*, vol. 21, no. 3, 2023.

[20] J. Morgan, D. Lee and F. Servant, "Empirical Studies on Regex Bugs, Repair, and Security in Modern Applications," *Software Quality Journal*, vol. 32, no. 4, pp. 1510–1528, 2024.