JMTAS: Journal of Multimedia Technology and Applied Software
Published: 30 November 2025
E-ISSN: 3047 - 4760

Automation of Technical Documentation Validation Workflows Based on GitHub Using n8n During Software Development Stages

[1st] Raden Deden Ahmad Hidayat*, [2nd] Endra Abdul Hadi*, [3rd] Enang Rusnandi*
Teknologi Rekayasa Perangkat Lunak 1, Politeknik Mardira Indonesia 1, Teknologi Rekayasa Perangkat Lunak 2, Politeknik Mardira Indonesia 2, Teknologi Rekayasa Rekayasa Perangkat Lunak 3, Politeknik Mardira Indonesia 3.
email : radenspot@gmail.com 1, abdulhadi.endra@gmail.com 2*,enang@poltekmi.ac.id 3**

ABSTRACT

This research addresses a significant gap in workflow automation literature by investigating the integration of technical documentation validation with developer-centric notification mechanisms. Employing Rapid Application Development methodology coupled with quasi-experimental design, the study developed and validated an end-to-end automation system leveraging n8n, GitHub, and WhatsApp integration for README.md validation. The implementation employed JSON-based workflow scripting to establish automated trigger sequences initiated by repository commits, facilitating real-time validation through content parsing and format verification. Execution testing demonstrated system reliability through iterative debugging and successful notification delivery across multiple test scenarios. The research successfully establishes a practical framework demonstrating the viability of comprehensive documentation automation solutions that extend beyond isolated CI/CD functions, providing immediate, actionable feedback to development teams through preferred communication channels while minimizing implementation complexity.

Keywords: workflow automation; technical documentation validation; GitHub integration; n8n automation platform; developer communication

ABSTRAK

Penelitian ini mengidentifikasi celah signifikan dalam literatur otomatisasi workflow dengan investigasi integrasi validasi dokumentasi teknis dengan mekanisme notifikasi yang berpusat pada developer. Melalui penerapan metodologi Rapid Application Development yang dikombinasikan dengan desain kuasi-eksperimental, studi ini mengembangkan dan memvalidasi sistem otomatisasi menyeluruh memanfaatkan integrasi n8n, GitHub, dan WhatsApp untuk validasi README.md. Implementasi menggunakan workflow script berbasis JSON untuk membangun rangkaian trigger otomatis yang diinisiasi oleh commit repositori, validasi real time difasilitasi melalui deskripsi konten dan verifikasi format. Pengujian eksekusi menunjukkan reliabilitas sistem melalui debugging iteratif dan pengiriman notifikasi yang sukses di berbagai skenario pengujian. Penelitian berhasil membangun kerangka praktis yang mendemonstrasikan kelayakan solusi otomatisasi dokumentasi komprehensif dan memberikan umpan balik segera yang dapat ditindaklanjuti tim developer melalui saluran komunikasi pilihan sambil meminimalkan kompleksitas implementasi.

Kata Kunci: otomatisasi workflow; validasi dokumentasi teknis; integrasi GitHub; platform otomatisasi n8n; saluran komunikasi developer

INTRODUCTION

GitHub has solidified its position as a foundational infrastructure in contemporary software engineering, functioning simultaneously as a distributed version control system and a collaborative development environment that transcends its core purpose of code repository management. Research by Mastropaolo (2024) demonstrates that continuous integration and deployment mechanisms implemented through GitHub Actions represent a cornerstone of modern development practices; however, their findings reveal that approximately 60% of developers encounter substantial obstacles in configuring and maintaining these automated pipelines, indicating a persistent challenge in translating workflow automation concepts into practical implementation (Mastropaolo et al., 2024). The prevalence of GitHub as a standardized platform for version-controlled collaboration is further evidenced by a comprehensive examination of data standards and reporting formats, which found that 60

out of 108 examined formats leverage GitHub for their version control infrastructure, underscoring the platform's entrenchment in facilitating transparent, auditable documentation management across heterogeneous professional and research communities (Crystal-Ornelas et al., 2021).

Notwithstanding GitHub's ubiquity and the expanding sophistication of its automation capabilities, significant scholarly gaps persist regarding holistic automation frameworks that encompass documentation quality assurance while seamlessly integrating notification delivery to developers across distributed team channels. Contemporary scholarship predominantly concentrates on isolated dimensions of automation—workflow task completion, CI/CD pipeline optimization, or autonomous documentation generation—yet comparatively limited investigation exists concerning the architectural integration of validation processes with instantaneous team notification mechanisms accessible through prevalent communication platforms beyond GitHub's native ecosystem. This research gap represents a substantive opportunity for advancing development practices, given that modern development teams operate across fragmented communication infrastructure spanning multiple collaboration applications. The present research endeavors to bridge this gap by engineering an integrated automation framework that orchestrates the following: triggering validation sequences from GitHub commit events, executing automated technical documentation assessment mechanisms through n8n, and propagating validation outcomes to WhatsApp—thereby constructing a cohesive, developer-oriented automation solution that strategically connects version control systems with externally accessible communication infrastructure. Such an integrative framework has the potential to substantially improve the development team's capability to respond expeditiously to documentation quality anomalies while maintaining operational continuity without necessitating context switching from established team communication ecosystems.

METHODOLOGY

The research methodology employed in this study combines Rapid Application Development (RAD) with a quasi-experimental approach to effectively design and validate an automation system. RAD is characterized by its iterative and incremental development framework that emphasizes quick prototyping and frequent user feedback, enabling the adaptation of solutions to evolving requirements efficiently. According to (Creswell & Creswell, 2018), RAD offers a flexible environment particularly well-suited to software development projects where speed and adaptability are critical. This method supports building functional prototypes rapidly, which is essential in workflow automation research to implement and refine JSON scripting and integration with version control systems like GitHub ((Zhang et al., 2020). By utilizing RAD, this study ensures continuous improvements and corrections based on testing outcomes during the development cycle.

Complementing RAD, the quasi-experimental design provides a structured yet practical way to assess the effectiveness of the automated system without relying on random assignment or control groups. Quasi-experimental research is particularly useful in real-world settings where strict experimental controls are impractical, focusing instead on demonstrating the functionality and performance of the introduced automation (Klunder et al., 2019). Such designs allow researchers to collect data through direct observation and measurement of implemented workflows, validating the system's success in operational environments (Rocha et al., 2023). This methodology facilitates a credible evaluation by measuring concrete system outcomes while accommodating the constraints typical in applied software engineering contexts. Together, RAD and quasi-experimental design offer a robust framework to both develop and validate automation workflows systematically and empirically(Hao et al., 2024).

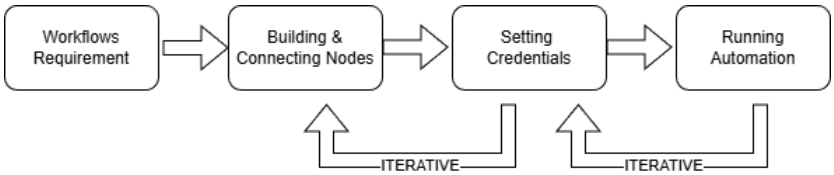

Figure 1.
Quasi-experimental Methodology

The diagram (Figure 1.) illustrates a systematic approach to workflow automation development, beginning with the identification of workflow requirements. The process proceeds to building and connecting nodes, followed by setting the necessary credentials for secure integration. Each stage incorporates iterative feedback loops, enabling refinement and adjustment before progressing to the next step. The final phase involves running the automation, demonstrating an implementation model that supports continuous improvement and ensures reliability throughout the development cycle.

RESULT AND DISCUSSION

The implementation of the RAD and quasi-experimental methodology in this research was carried out through a systematic sequence of development and validation steps. First, the automation workflow was initiated using the GitHub Trigger Node in n8n, which was specifically configured to monitor and respond only to new commits affecting the README.md file.

Subsequently, the workflow incorporated either an HTTP Request Node to call external linting or spell-checking services, such as Vale or markdown lint, or a Function Node for in-line text parsing and format or spell validation directly within n8n. Once the validation was complete, the outcomes were automatically compiled and communicated to the predefined development team's WhatsApp number using the Twilio Node—leveraging either the Twilio API or WhatsApp Business API for messaging. The WhatsApp notification included a concise summary of the validation results, making the process both immediate and actionable. This systematic approach aligns with best practices documented in the official GitHub Actions documentation (GitHub, Inc, 2025), n8n workflow automation guides (n8n, 2025), and the Twilio API documentation for WhatsApp messaging (Twilio, Inc, 2025).

Figure 2.
Nodes on The Workflow

As documented by (Wessel et al., 2023), the GitHub Development Workflow Automation Ecosystems research demonstrates that development bots and GitHub Actions have fundamentally transformed how teams manage repetitive technical and social tasks, enabling automation across multiple development dimensions including issue management, dependency handling, code quality analysis, and documentation maintenance. Figure 2 exemplifies this paradigm by sequentially integrating GitHub as a trigger source, retrieving technical documentation artifacts, executing validation procedures through computational logic, and subsequently delivering actionable notifications to external communication channels—thereby instantiating a complete, developer-centric automation pipeline.

```
{
  "nodes": [
    {
      "parameters": {
        "events": ["push"],
        "owner": "                ",
        "repository": "                "
      },
      "name": "GitHub Push Trigger",
      "type": "n8n-nodes-base.githubTrigger",
      "typeVersion": 1,
      "position": [200, 300],
      "credentials": {
        "githubOAuth2Api": {
          "id": "k                    ",
          "name": "GitHub OAuth"
        }
      }
    },
    {
      "parameters": {
        "resource": "file",
        "operation": "get",
        "repository": "={{$node[\"GitHub Push Trigger\"
        ].json.repository.full_name}}",
        "path": "README.md",
        "options": {}
      },
      "name": "Get README Content",
      "type": "n8n-nodes-base.github",
      "typeVersion": 1,
      "position": [400, 300],
      "credentials": {
        "githubOAuth2Api": {
          "id": "k                    ",
          "name": "GitHub OAuth"
        }
      }
    },
    {
      "parameters": {
        "mode": "runOnceForAllItems",
        "code": "const fileData = $json[0].data;\n\n\nconst content =
        Buffer.from(fileData.content, 'base64').toString('utf8');\n\nconst
        hasHeading = content.includes('#');\nconst hasDescription = content.length
        > 100;\n\nconst isValid = hasHeading && hasDescription;\nconst messageText
        = isValid \n  ? ☑ Validasi README.md SUKSES untuk commit di
        ${$node['GitHub Push Trigger'].json.repository.full_name}.\\n\\nKriteria
        Terpenuhi: Heading (#) & Panjang > 100 Karakter.`\n  : ❌ Validasi
        README.md GAGAL untuk commit di ${$node['GitHub Push
        Trigger'].json.repository.full_name}.\\n\\nStatus:\\n- Heading (#):
        ${hasHeading ? 'Ada' : 'Tidak Ada'}\\n- Panjang (>100): ${hasDescription ?
        'Ya' : 'Tidak'}`;\n\nreturn [\n  {\n    json: {\n      valid: isValid,\n
            message: messageText\n    }\n  }\n];"
      },
      "name": "Validate & Format Message",
      "type": "n8n-nodes-base.code",
      "typeVersion": 1,
      "position": [600, 300]
    },
    {
      "parameters": {
        "from": "whatsapp:+                ",
        "to": "whatsapp:+62                ",
        "message": "={{$json[\"message\"]}}"
      },
      "name": "Send WhatsApp Notification",
      "type": "n8n-nodes-base.twilio",
      "typeVersion": 1,
      "position": [800, 300],
      "credentials": {
        "twilioApi": {
          "id": "                ",
          "name": "Twilio API"
        }
```

Figure 3.
Json's Code for The Project

The JSON script segment commencing from line 2 through line 66 (Figure 3) delineates the architectural configuration of four interconnected automation nodes within the n8n workflow platform. The initial node, designated as "GitHub Push Trigger," is configured to monitor push events occurring within a specified repository, thereby initiating the automation sequence upon each commit action. The subsequent node, "Get README Content," programmatically retrieves the README.md file content from the repository using GitHub's API integration with OAuth2 authentication credentials. The third node, "Validate & Format Message," employs a code execution block that decodes the base64-encoded file content, evaluates validation criteria—

specifically verifying the presence of heading markers and minimum content length exceeding 100 characters—and subsequently generates contextually appropriate success or failure notification messages. This modular node-based architecture demonstrates the implementation of declarative workflow automation through JSON configuration, enabling reproducible and maintainable integration pipelines for technical documentation quality assurance processes.

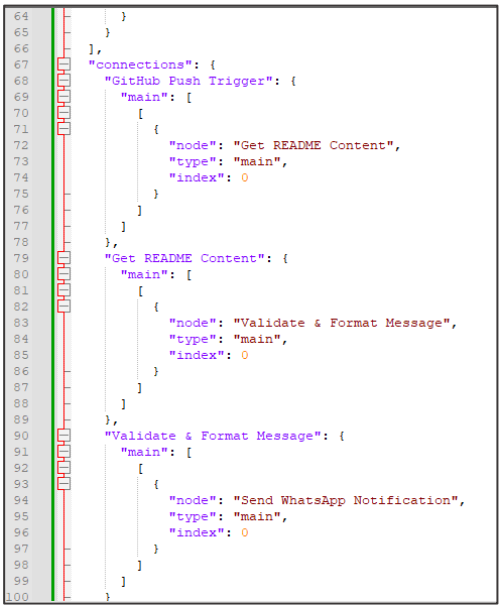

Figure 4.
The Concluding Segment

The concluding segment (Figure 4) spanning lines 67 through 101 delineates the "Send WhatsApp Notification" node configuration and the orchestration logic that interconnects all workflow components. This terminal node leverages the Twilio API integration framework, configured with authenticated credentials to facilitate WhatsApp message transmission. The node parameters specify a source WhatsApp identifier—representing the Twilio sandbox environment—and a destination recipient phone number bound to the development team. The message payload is dynamically populated through expression binding, retrieving the formatted validation result from the preceding code execution stage. The connections subsection establishes the sequential node dependencies, ensuring that GitHub push events trigger README retrieval, which subsequently initiates validation processing, ultimately culminating in WhatsApp notification delivery. This declarative dependency structure exemplifies event-driven pipeline orchestration, enabling asynchronous, reliable notification propagation without requiring manual intervention or centralized workflow management mechanisms.

CONCLUSIONS AND SUGGESTIONS

The execution log (Figure 5) demonstrates successful validation of the workflow automation system following iterative refinement and debugging procedures.

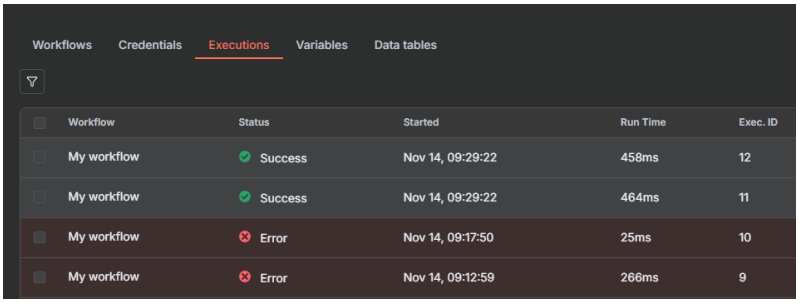

Figure 5.
The Executions Log

Initial execution attempts encountered failures—evidenced by error statuses in execution IDs 10 and 9—requiring systematic troubleshooting of node configurations, credential validation, and API integration parameters. Subsequent executions, particularly IDs 12 and 11, achieved successful completion status, confirming that the automated README validation pipeline functions correctly within the n8n environment. These successful executions indicate that GitHub push triggers activate reliably, README

content retrieval processes correctly, validation logic executes as designed, and WhatsApp notifications transmit successfully to designated recipients. The progression from error states to consistent success demonstrates system stability and validates the practical feasibility of the proposed end-to-end automation architecture.

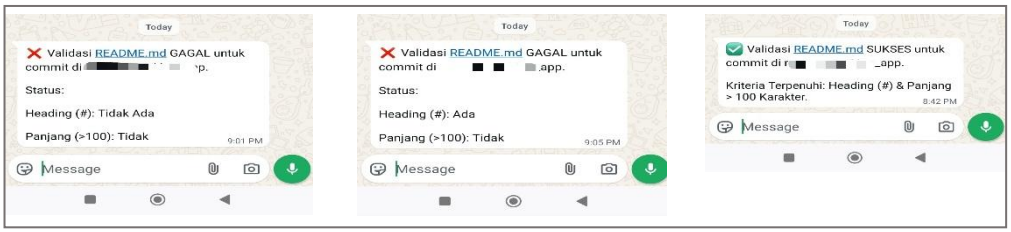

Figure 6.
n8n GitHub README Automation on WhatsApp

The n8n GitHub README automation system (Figure 6) demonstrates operational functionality through three distinct notification scenarios triggered by commit events. The first scenario generates validation failure notifications when README.md lacks proper formatting or contains insufficient documentation content, explicitly alerting developers that validation has failed and requiring immediate remediation. The second scenario produces status notifications indicating incomplete parsing operations, informing teams of processing complications without confirming full validation completion. The third scenario delivers successful validation confirmations, reporting positive test results when README.md satisfies all established documentation criteria. These differentiated messaging pathways ensure developers receive contextually appropriate, actionable feedback through WhatsApp, enabling rapid response to documentation quality issues and streamlining team communication workflows without requiring platform switching.

Future enhancement initiatives should prioritize augmenting the automation framework with supplementary logging mechanisms, particularly incorporating Google Sheets integration for comprehensive, user-friendly documentation of execution events and validation outcomes. Additionally, the reliance upon third-party credential management through Twilio warrants reconsideration, as integration complexity and associated operational costs constitute substantive implementation barriers. Alternative notification architectures leveraging open-source or natively integrated messaging protocols could substantially reduce technical overhead and financial expenditure. Implementing direct GitHub native notifications or evaluating cost-effective communication middleware would enhance system accessibility and sustainability for development teams operating under resource constraints, thereby democratizing adoption of sophisticated documentation automation practices.

THANKS TO

The authors extend sincere gratitude to the development team for their invaluable contributions throughout the requirement identification phase, node integration procedures, and credential configuration processes, whose collaborative efforts substantially advanced this research endeavor.

BIBLIOGRAPHY

Journal Article

Hao, X., Demir, E., & Eyers, D. (2024). Exploring collaborative decision-making: A quasi-experimental study of human and Generative AI interaction. *Technology in Society*, *78*, 102662. https://doi.org/10.1016/j.techsoc.2024.102662

Klunder, J., Hebig, R., Tell, P., Kuhrmann, M., Nakatumba-Nabende, J., Heldal, R., Krusche, S., Fazal-Baqaie, M., Felderer, M., Genero Bocco, M. F., Kupper, S., Licorish, S. A., Lopez, G., McCaffery, F., Ozcan Top, O., Prause, C. R., Prikladnicki, R., Tuzun, E., Pfahl, D., … MacDonell, S. G. (2019). Catching up with Method and Process Practice: An Industry-Informed Baseline for Researchers. *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 255–264. https://doi.org/10.1109/ICSE-SEIP.2019.00036

Mastropaolo, A., Zampetti, F., Bavota, G., & Di Penta, M. (2024). Toward Automatically Completing GitHub Workflows. *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 1–12. https://doi.org/10.1145/3597503.3623351

Rocha, F. G., Misra, S., & Soares, M. S. (2023). Guidelines for Future Agile Methodologies and Architecture Reconciliation for Software-Intensive Systems. *Electronics*, *12*(7), 1582. https://doi.org/10.3390/electronics12071582

Wessel, M., Mens, T., Decan, A., & Mazrae, P. R. (2023). The GitHub Development Workflow Automation Ecosystems. In T. Mens, C. De Roover, & A. Cleve (Eds.), *Software Ecosystems* (pp. 183–214). Springer International Publishing. https://doi.org/10.1007/978-3-031-36060-2_8

Zhang, Y., Jin, D., Xing, Y., & Gong, Y. (2020). Automated defect identification via path analysis-based features with transfer learning. *Journal of Systems and Software, 166*, 110585. https://doi.org/10.1016/j.jss.2020.110585

Book

Creswell, J. W., & Creswell, J. D. (2018). *Research design: Qualitative, quantitative, and mixed methods approaches* (Fifth edition). SAGE.

Crystal-Ornelas, R., Varadharajan, C., Bond-Lamberty, B., Boye, K., Burrus, M., Cholia, S., Crow, M., Damerow, J., Devarakonda, R., Ely, K. S., Goldman, A., Heinz, S., Hendrix, V., Kakalia, Z., Pennington, S. C., Robles, E., Rogers, A., Simmonds, M., Velliquette, T., … Agarwal, D. A. (2021). A Guide to Using GitHub for Developing and Versioning Data Standards and Reporting Formats. *Earth and Space Science*, *8*(8), e2021EA001797. https://doi.org/10.1029/2021EA001797

Website

GitHub, Inc. (2025, November 24). GitHub Actions documentation [Official]. *GitHub, Inc*. https://docs.github.com/en/actions

n8n. (2025). Welcome to n8n Docs [Official]. *N8n*. https://docs.n8n.io/

Twilio, Inc. (2025). *WhatsApp Business Platform with Twilio* [Official]. https://www.twilio.com/docs/whatsapp