World Journal of Advanced Engineering Technology and Sciences

eISSN: 2582-8266

Cross Ref DOI: 10.30574/wjaets

Journal homepage: https://wjaets.com/

(Review Article)

Automated reporting and statistical analysis of test case results in continuous integration: A custom dashboard approach

Rahul M Salagundi * and Savitha R

Department. of Master of Computer Applications, RV College of Engineering, Bengaluru, India.

World Journal of Advanced Engineering Technology and Sciences, 2024, 13(01), 027–033

Publication history: Received on 16 July 2024; revised on 25 August 2024; accepted on 28 August 2024

Article DOI: https://doi.org/10.30574/wjaets.2024.13.1.0374

Abstract

CI has been one of the cornerstones of modern software development, where teams integrate code changes frequently and have automated testing to receive fast feedback. Running a large amount of test data, however, is overwhelming and can be fatiguing in extracting meaningful insights from traditional reporting methods like email notifications. The paper presents a bespoke dashboard, designed to support the extension of automation in test case result reporting and analysis within a CI environment by real-time data visualization, interactivity in filtering and sorting, and statistical insights that would make trends and anomalies easier for teams to detect. Implemented in a large software project, this solution brought immense improvement in decision-making, team collaboration, and efficiency, since the dashboard reduced time spent on manual report analyses and generations. The flexibility and ability to integrate with a number of CI tools make it a powerful solution for addressing traditional reporting shortcomings and let development teams focus on areas needing improvement, ultimately enhancing software quality.

Keywords: Continuous Integration; Jenkins; Automated Pipelines; Automated Reporting

1. Introduction

In the last couple of years, agile methodologies have swept through software development, boasting about their flexibility, collaboration, and satisfaction of customers. One core practice of agile development is continuous integration. It is a process in which developers regularly integrate changes, expressed in the form of code, into a common repository. Automated builds and running tests ensure that new code integrates well with existing code bases.[1][2][3]

Automated testing is an intrinsic part of CI, which gives fast feedback on the health of a codebase and ensures that new changes do not break its integrity. On the other hand, with the growing volume and complexity of software projects, test data becomes too large to process results effectively, and develop teams are faced with certain analysis for correct decisions.[4]

Our paper proposes a dedicated dashboard for test case results in a CI environment that introduces automation in reporting and statistical analyses. With the proposed dashboard, advanced data visualization would express performance intuitively and clearly for the tests. Real-time insights will be helpful for the team to locate trends, detect anomalies, and show areas of improvement. It is a method that improves not only the testing phase but general decision making in general, increasing the quality of software overall.[5]

* Corresponding author: Rahul M Salagundi

2. Literature Review

2.1. Continuous Integration Pipelines

The importance of the CI pipelines doesn't only come in when it comes to the fast-paced environment of software development today. They enable the team to routinely merge changes in code and automate the building and testing of software. This way, one is guaranteed that problems will be identified at an early stage and that the development process will run smoothly.[6]

2.1.1. Simplified Build and Deployment

CI pipelines automate the building and deployment of applications, making it easier to integrate changes in the code and deliver them seamlessly.

Quick Feedback: This means that the developer instantly gets feedback about the code, hence able to identity and fix problems at the earliest possible time, which in turn reduces headaches and solves problems in less time.[7]

2.1.2. Resource Efficiency

Automating repeated tasks allows the team members to focus more on the strategic and creative aspects of development. Automation in CI also decreases the potential for human error within the build and testing processes. The result is more reliable builds and fewer mistakes that may grossly affect the final product.[8]

2.2. Robot Framework

Robot Framework is an open, generic test automation framework. The special thing about it is its flexibility and user-friendly design, which fits very well in a wide variety of tests.*[9][10]*

2.2.1. Easy to use

The keyword-driven approach of this framework makes it very easy to use by technical and non-technical end-users, hence simplifying test case development and maintenance.

2.2.2. High Extensibility

It uses several libraries and other tools with capabilities to be highly customized and integrated with other testing solutions as per need.

2.2.3. Flexible Testing

Adaptable for 'n' number of type of testing, such as acceptance tests or end-to-end tests, hence capable enough to provide the entire solution for testing needs.

2.2.4. Platform Independence

It makes executing tests on different OS and environments possible, due to which this technology created by Robot Framework is highly compatible and flexible.

2.3. Web Development

Due to dynamism, scalability, and performance requirements of applications, web development has grown exponentially. However, modern web development does not simply mean technical robustness but means developing applications that ensure a great user experience.[11]

2.3.1. Rapid Prototyping

The modern web development tool and framework provides fast prototyping. It aids developers in fast setup of the web application structure and iteration, thus enabling one to explore new ideas or implement any change quickly. This kind of agility is going to be essential in the process of keeping up with changing user needs and market trends.

2.3.2. Responsive Design

Responsive design gives room to the fact that the application looks and works great across all devices—from desktops to devices like smartphones and tablets. Some of the techniques currently in practice to offer a wonderful user experience, regardless of screen size, would be media queries and flexible grid layouts. It thus lays a base for maintaining consistency and accessibility, which becomes crucial in a mobile-first world.

2.3.3. Scalability and Performance

Very critical in modern web development is scalability. Applications are designed to scale to traffic and a growing user base, handling the demanding job with both speed and reliability. Load balancing, caching, and asynchronous processing are techniques often put in place to ensure performance optimization—enabling everything to run ultra-smooth even under very heavy loads.

2.3.4. User-Centric Design

The new web development enforces the idea of ease and accessibility of the interface. It puts into action the principles of user experience design, hence making the application easier and more user-friendly to use. This would involve understanding user behavior and running usability tests to further fine-tune the interface in order to keep it in line with the user's needs and expectations.[12]

2.4. Automated Reporting

Jenkins pipelines allow for seamless integration of automated reporting into the CI/CD process. By using Jenkins, teams can configure automated reporting tasks as part of the pipeline, ensuring that reports are generated consistently whenever a build is executed.

2.4.1. Automated Test Results Collection

Jenkins can automatically collect test results from various testing frameworks and tools. It aggregates the data from unit tests, integration tests, and other types of tests, providing a comprehensive overview of the application's health.

2.4.2. Real-Time Reporting

Jenkins supports real-time reporting by generating and updating reports as new builds and tests are executed. This allows developers and stakeholders to access up-to-date information about the status and quality of the software.

2.5. Key Tools

2.5.1. Jenkins

It’s the most powerful automation system explaining one's software builds and deployment. It keeps the team continuously integrating and delivering changes in code, automates the tasks of building, testing, and software deployment. Jenkins makes it so things run smooth so the developer can actually just focus on building code instead of managing repetitive chores.[14][15]

2.5.2. Python Robot Framework

It is an easy way to do automatic assessment and processes because it employs simple keyword-driven approaches to help write test cases and understand them. This framework can be easily extended with a number of libraries so as to fit various needs, from acceptance testing to automating repetitive tasks.

2.5.3. Elasticsearch

Elasticsearch is among the most powerful search engines dealing with big data volumes at a considerably good speed. It's good for things that require rapid searching or real-time analytics. When it comes to building out a search feature or searching through log data, Elasticsearch helps you find and make sense of your data fast.[13]

2.5.4. Flask

Flask is a flexible web framework for Python that you can start using very easily. It makes the development of the web application very rapid, without much of a headache, and this is because it is simple and has small setup. With Flask, you can write anything in small projects or in large applications, making sure your app remains lightweight and simple.

2.6. Proposed Framework

To address the challenges of reporting and analyzing test case results, a comprehensive solution has been developed that integrates a robust backend, an intuitive frontend, and seamless log integration.

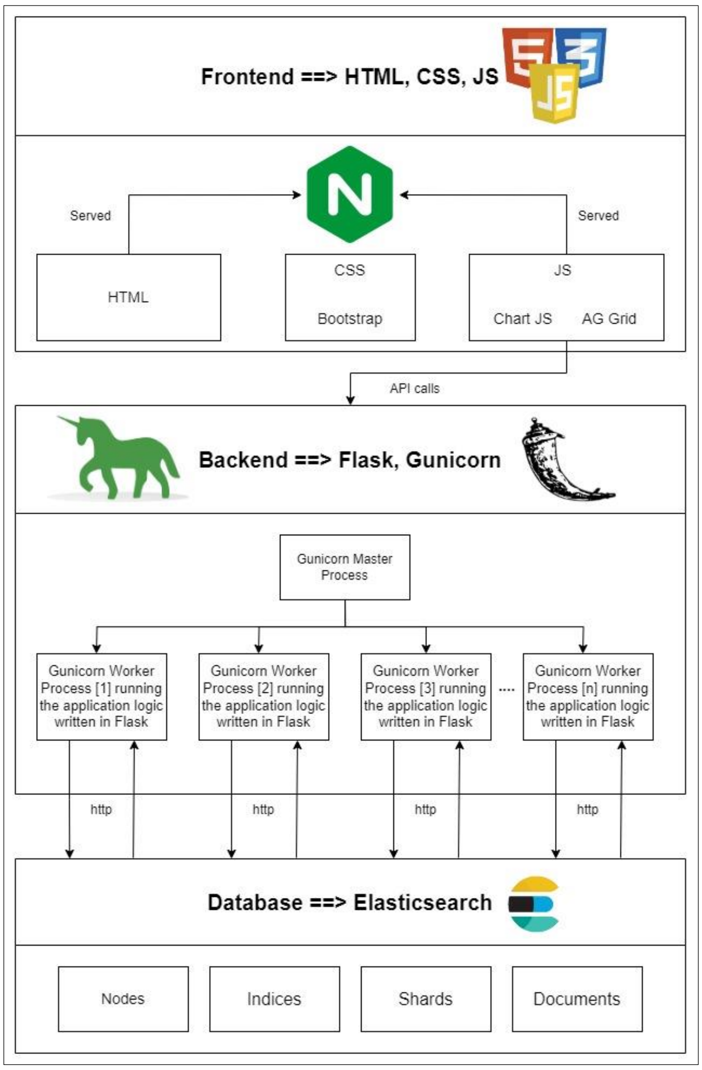
Frontend ==> HTML, CSS, JS
Served
Served
HTML
CSS
Bootstrap
JS
Chart JS
AG Grid
API calls
Backend ==> Flask, Gunicorn
Gunicorn Master Process
Gunicorn Worker Process [1] running the application logic written in Flask
Gunicorn Worker Process [2] running the application logic written in Flask
Gunicorn Worker Process [3] running the application logic written in Flask
Gunicorn Worker Process [n] running the application logic written in Flask
http
http
http
http
Database ==> Elasticsearch
Nodes
Indices
Shards
Documents

Figure 1 Proposed Solution

2.6.1. Execution and Data Ingestion

Jenkins Pipeline

Test cases are executed through a Jenkins pipeline, which automates the build and testing process. As tests run, listeners generate JSON files containing the results of each test case.

Elasticsearch Integration

These JSON files are then pushed to Elasticsearch, where they are indexed and stored. Elasticsearch's powerful search capabilities allow for efficient querying and retrieval of test case data.

2.6.2. Backend System

Elasticsearch

Serves as the primary database for storing and managing test case execution data. Its powerful indexing and search capabilities enable efficient data retrieval and analysis, handling large volumes of test results with ease.

Flask

Used to build the backend API that interfaces with Elasticsearch. This lightweight Python framework simplifies the development of a server that can query and interact with the data, facilitating smooth data operations.

Gunicorn

Employed to serve the Flask application, Gunicorn manages multiple worker processes to handle concurrent requests. This ensures that the backend remains responsive and performs well even under high load.

2.6.3. Frontend Interface

Prototype

Developed using HTML, CSS, and JavaScript. This initial version provides a functional and user-friendly interface for interacting with the data and generating reports.

Deployment

The frontend is deployed using Nginx, a reliable web server that efficiently serves static content and manages traffic, ensuring a stable and smooth user experience.

2.6.4. Integration with Logstore

Well, integration with a log management system grants easy access to logs created by the Robot Framework. The interface hyperlinks open log entries for the related test suites, helping in easy troubleshooting and review of detailed execution data.

2.6.5. Reporting and Analysis

Test View

This view is designed for generating comprehensive reports, features for entering failure analysis details, and requesting additional traces. This page is designed to be a central location for test results compilation and analysis.

Historical View

This provides a history of the runs for both test suites and test cases, which is useful for monitoring trends in performance over time.

Statistics view

This provides a run, release, and team-based breakup of the results. Such statistics-oriented insights provide data-driven decision-making and process improvements

3. Results and discussion

3.1. Increased Efficiency and Accessibility

By automating test result ingestion and providing an intuitive interface, the system reduces the time and effort needed to access and analyze test data. Developers and testers can quickly obtain the information they need, enhancing productivity and streamlining the workflow.

3.2. Enhanced Search Capabilities and Historical Tracking

Elasticsearch's powerful search capabilities enable efficient querying and filtering of test results. Users can easily track the history of test executions and identify trends over time, especially across different releases, helping teams understand software quality evolution and pinpoint improvement areas.

3.3. Real-Time Test Report Updates

The system provides real-time updates of test reports, allowing stakeholders to access the latest results as soon as they are available. This real-time visibility supports prompt decision-making and quick identification of issues, maintaining a high level of code quality.

3.4. Comprehensive Dashboard for CI Activities

The dashboard offers a unified view of all continuous integration activities, consolidating information from various pipeline stages. Users can view test results, analyze failure patterns, and request traces, simplifying CI process management and providing a clear overview of development efforts.

3.5. Centralized Report Data Storage

Centralizing report data within Elasticsearch ensures secure storage and easy access from a single location. This approach simplifies data management, enhances integrity, and provides a single source of truth for all test data, ensuring consistency across the team.

4. Conclusion and Future Work

The proposed solution for automated reporting and statistical analysis of test case results has significantly enhanced the continuous integration process by providing increased efficiency, improved accessibility, and comprehensive insights into test data. By leveraging technologies such as Jenkins, Elasticsearch, and Flask, along with a user-friendly dashboard, the system offers real-time visibility into test results, robust search capabilities, and a centralized storage solution. These advancements enable development teams to quickly identify issues, track historical performance, and make data-driven decisions, ultimately leading to higher software quality and productivity.

Building on the current system, future work will focus on incorporating advanced features to further enhance test case analysis and reporting:

4.1. Automatic Fix Suggestions

By analyzing previously entered data from testers, the system can suggest potential fixes for test cases that fail with similar error messages. This feature will leverage machine learning algorithms to identify patterns and recommend solutions, reducing the time spent troubleshooting recurrent issues.

4.2. Error Classification

Implementing a classification system for errors will allow the categorization of test case failures based on error types and patterns. This classification will help in understanding the root causes of failures and facilitate targeted improvements to the testing process.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] B. Choudhary and S. K. Rakesh, "An approach using agile method for software development," 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH), Greater Noida, India, 2016, pp. 155-158, doi: 10.1109/ICICCS.2016.7542304.

[2] M. Kuhrmann et al., "What Makes Agile Software Development Agile?," in IEEE Transactions on Software Engineering, vol. 48, no. 9, pp. 3523-3539, 1 Sept. 2022, doi: 10.1109/TSE.2021.3099532.

[3] A. R. Chaudhari and S. D. Joshi, "Study of effect of Agile software development Methodology on Software Development Process," 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2021, pp. 1-4, doi: 10.1109/ICESC51422.2021.9532842.

[4] J. Bosas, "Automated Testing Importance and Impact," 2018 IEEE AUTOTESTCON, National Harbor, MD, USA, 2018, pp. 1-4, doi: 10.1109/AUTEST.2018.8532522.

[5] L. Gota, D. Gota and L. Miclea, "Continuous Integration in Automation Testing," 2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), Cluj-Napoca, Romania, 2020, pp. 1-6, doi: 10.1109/AQTR49680.2020.9129990.

[6] M. R. Pratama and D. Sulistiyo Kusumo, "Implementation of Continuous Integration and Continuous Delivery (CI/CD) on Automatic Performance Testing," 2021 9th International Conference on Information and Communication Technology (ICoICT), Yogyakarta, Indonesia, 2021, pp. 230-235, doi: 10.1109/ICoICT52021.2021.9527496.

[7] A. M. Mowad, H. Fawareh and M. A. Hassan, "Effect of Using Continuous Integration (CI) and Continuous Delivery (CD) Deployment in DevOps to reduce the Gap between Developer and Operation," 2022 International Arab Conference on Information Technology (ACIT), Abu Dhabi, United Arab Emirates, 2022, pp. 1-8, doi: 10.1109/ACIT57182.2022.9994139.

[8] F. Zampetti, S. Geremia, G. Bavota and M. Di Penta, "CI/CD Pipelines Evolution and Restructuring: A Qualitative and Quantitative Study," 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME), Luxembourg, 2021, pp. 471-482, doi: 10.1109/ICSME52107.2021.00048.

[9] Shetty V, Swetha S, Bindu Ashwini C (2020) An Empirical Study on Robot Test Automation Framework. J Comput Eng Inf Technol 9:3. doi: 10.37532/jceit.2020.9(3).227

[10] L. Jian-Ping, L. Juan-Juan and W. Dong-Long, "Application Analysis of Automated Testing Framework Based on Robot," 2012 Third International Conference on Networking and Distributed Computing, Hangzhou, China, 2012, pp. 194-197, doi: 10.1109/ICNDC.2012.53.

[11] T. Uppal, S. Srivastava and K. Saini, "Web Development Framework : Future Trends," 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2022, pp. 2181-2184, doi: 10.1109/ICAC3N56670.2022.10074105.

[12] H. Božiković and M. Štula, "Web design — Past, present and future," 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 2018, pp. 1476-1481, doi: 10.23919/MIPRO.2018.8400266.

[13] F. Ahmed, U. Jahangir, H. Rahim, K. Ali and D. -e. -S. Agha, "Centralized Log Management Using Elasticsearch, Logstash and Kibana," 2020 International Conference on Information Science and Communication Technology (ICISCT), Karachi, Pakistan, 2020, pp. 1-7, doi: 10.1109/ICISCT49550.2020.9080053.

[14] P. A. G. Permana, E. Triandini and N. L. G. P. Suwirmayanti, "Implementation Jenkins Automation Deployment with Scheduler and Notification," 2021 3rd International Conference on Cybernetics and Intelligent System (ICORIS), Makasar, Indonesia, 2021, pp. 1-5, doi: 10.1109/ICORIS52787.2021.9649555.

[15] S. Mysari and V. Bejgam, "Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible," 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), Vellore, India, 2020, pp. 1-4, doi: 10.1109/ic-ETITE47903.2020.239.