COMPARATIVE ANALYSIS OF AUTOMATION FUNCTIONAL TESTING TOOLS PERFORMANCE FOR PLAYSTORE APPS WITH DIA METHOD

Faizal Riza [1*]; Berliyanto [2] ; Aji Nurrohman [3]; Rachmat Setiabudi [4]

Teknik Informatika[1,2,3,4]
Institut Teknologi Budi Utomo, Jakarta, Indonesia[1,2,3,4]
www.itbu.ac.id[1,2,3,4]
akuisal@gmail.com[1*], berli@itbu.ac.id[2], ajinurrohman7@gmail.com[3], raffisetiabudi@gmail.com[4]
(*) Corresponding Author

Abstract—*The complexity of smartphone applications presents challenges for developers, who must ensure flawless functionality despite limitations such as budget and time constraints. Manual testing is time-consuming, prompting a shift towards automated testing methods to ensure efficiency and reliability. In this context, researchers are evaluating the efficacy of three leading test automation frameworks—Robot Framework, Katalon Studio, and UI Path—against key performance parameters. Using the Distance to the Ideal Alternative (DIA) method on playstore apps. The main performance parameters used as a reference are automated testing progress and tools usability. Katalon Studio emerges as the top performer, securing the top rank with a remarkably close to the alternative ideal positive distance (Ri) value of 0.00001. UI Path occupies the second position with a Ri value of 0.00135, while Robot Framework trails behind with a Ri value of 0.00295. This research contributes to the understanding of the performance of different automation frameworks in the context of functional testing, providing valuable insights for developers and organizations seeking to optimize their testing processes. The findings underscore the significance of Katalon Studio's exceptional performance and highlight opportunities for improvement in UI Path and Robot Framework. Additionally, implementing a robust monitoring and evaluation framework is crucial for tracking the ongoing performance and optimizing the efficiency of these automation frameworks.*

Keywords: *automation, DIA method, playstore, testing.*

Intisari—*Kompleksitas aplikasi ponsel cerdas menghadirkan tantangan bagi pengembang perangkat lunak, yang harus memastikan fungsionalitas aplikasinya sempurna meskipun terdapat keterbatasan seperti anggaran dan waktu. Pengujian secara manual akan membutuhkan banyak waktu, sehingga mendorong pengembang beralih ke metode pengujian otomatis untuk memastikan efisiensi dan keandalan pengujian. Dalam penelitian ini, peneliti mengevaluasi kinerja tiga test automation framework yaitu Robot Framework, Katalon Studio, dan UI Path untuk pengujian pada aplikasi playstore. Parameter kinerja utama yang dijadikan acuan adalah automated testing progress dan tools usability. Metode yang digunakan pada penelitian ini adalah metode Distance to the Ideal Alternative (DIA). Penelitian menunjukkan bahwa kinerja terbaik dicapai oleh Katalon Studio yang menempati peringkat teratas karena memiliki jarak terdekat dari nilia positif ideal alternatif (Ri) yaitu 0,00001. UI Path menempati posisi kedua dengan nilai Ri sebesar 0,00135, sedangkan Robot Framework berada di posisi kedua dengan nilai Ri sebesar 0,00295. Penelitian ini berkontribusi pada pemahaman kinerja berbagai test automation framework untuk pengujian fungsional serta memberikan wawasan bagi pengembang perangkat lunak dan organisasi yang ingin mengoptimalkan proses pengujian aplikasi yang digunakan. Haisl penelitian ini menunjukkan bahwa kinerja Katalon Studio melampaui UI Path dan Robo Framework. Lebih jauh diperlukan monitoring terhadap perkembangan ketiga framework agar secara berkala dilakukan komparasi ulang agar dapat memastikan keandalan ketiga framework secara periodik.*

Kata Kunci: *otomatisasi, metode DIA, playstore, pengujian.*

INTRODUCTION

The ubiquitous nature of mobile applications (apps) in critical domains like finance, healthcare, and logistics has spurred a surge of interest in the field of automated mobile app testing. This emphasis reflects the growing recognition of the necessity for robust and efficient quality assurance methodologies in the mobile app development lifecycle (Lin et al., 2020). To synchronize development velocity with software release cycles, a paradigm shift towards test automation is essential for achieving alignment with the software development life cycle. Traditional manual testing methodologies are demonstrably inadequate in high-quality software engineering due to their inherent limitations in covering all potential bug manifestations (Salam et al., 2022). Developers face limitations like budget, time, and resources, often releasing apps quickly under market pressure. These apps need to function across various systems and user actions, requiring extensive testing – a time-consuming task. Unsurprisingly, apps often malfunction or struggle with unexpected user behavior. This not only frustrates users and reduces perceived convenience, but also damages developer reputations (Baktha, 2020).

Mobile app development needs testing to be secure, reliable, and work smoothly. This means using special testing methods designed for mobile devices, beyond just basic functionality checks. Testers can do this manually, but with the fast-paced app market, automated testing using scripts is preferred for speed and efficiency (Menegassi & Endo, 2020). In automation testing, software errors are caught by running automated scripts, rather than having a person manually perform test cases. This approach is faster, more reliable, and yields more accurate results, ensuring the software meets quality standards (Karlsson et al., 2021). Given the abundance of test automation tools and the need for quick, reliable testing in the playstore, automation testing is the preferred method for playstore-based applications (Aslam et al., 2022).

Researchers are evaluating how well different tools automate functional testing for the popular Bizhare investment platform. Launched in 2018 and known for its user-friendly interface, Bizhare enjoys high ratings on Playstore. This study compares three top-performing test automation frameworks: Robot Framework, Katalon Studio, and UI Path. It focuses on three key performance parameters: the number of test cases covered, the time complexity of test execution, and overall execution speed (Berihun et al., 2023).

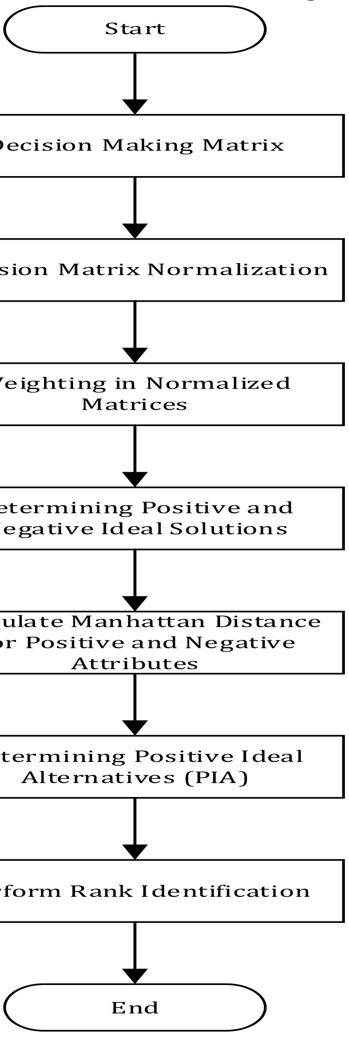
To identify the best framework based on these criteria, the researchers will employ the Distance to the Ideal Alternative (DIA) method. Previous research demonstrates that DIA outperforms other common methods like TOPSIS, SAW, and WP in its ability to accurately rank alternatives, particularly when dealing with ranking abnormalities (Abdulwareth & Al-Shargabi, 2021). This suggests DIA will provide a more nuanced and reliable comparison of the three frameworks. Following the suggestions provided in previously researched references, the researcher opted to utilize the Distance to the Ideal Alternative method in this study.

MATERIALS AND METHODS

The research design underwent multiple stages as outlined below (Arya et al., 2021):

1. Data collection
   The gathered data consists of literature concerning the DIA method, along with specifications for Robot Framework, Katalon Studio, and UI Path, aimed at evaluating their effectiveness as tools for automation functional testing.
2. Identify the application
   The Playstore application used as a test object is Bizhare, an investment platform that has been fully operational since 2018.
3. Design test scenarios (*test case*)
   The scenario design used is testing the login feature with 5 scenarios, namely 4 negative scenarios (failed login) and 1 positive scenario (successful login). The four negative scenarios are app login with an empty password, app login with an empty email, app login with an incorrectly formatted email and app login with an incorrect email and password. Meanwhile, one positive scenario is app login provided the app login is successful.
4. Emulator settings
   To carry out automatic functional testing for mobile applications, an emulator or real device is needed that is compatible with the operating system. In this research, the author chose to use an Android-based real device. By using a real device, the author only needs to connect using a data cable and set the device to Developer Mode.
5. Implementation automation functional testing
   The general steps for automation functional testing are setting up app tools, setting up scripts for test automation scenarios, running tests and generating results. General steps are taken in the three test automation frameworks, namely Robot Framework, Katalon Studio and UI Path.
6. Analysis of determining the best performance decisions for automation functional testing tools using the DIA method.

The calculation flow of The Distance To The Ideal Alternative (DIA) method for selecting the best test automation frameworks is shown in Figure 1.


Start
Decision Making Matrix
Decision Matrix Normalization
Weighting in Normalized Matrices
Determining Positive and Negative Ideal Solutions
Calculate Manhattan Distance For Positive and Negative Attributes
Determining Positive Ideal Alternatives (PIA)
Perform Rank Identification
End
Source : (Aslam et al., 2022)
Figure 1. Calculation Process Using The Distance To The Ideal Alternative Method

The Distance To The Ideal Alternative (DIA) method is one of the MADM methods, MADM or Multiple Attribute Decision Making itself is a method used to find the optimal alternative from a number of alternatives with certain criteria . The essence of MADM is to determine the weight value for each attribute, then proceed with a ranking process that will select the alternatives that have been given (Chakraborty, 2022). In general, it can be said that MADM selects the best alternative from a number of alternatives. There are several MADM methods such as the SAW, WP, ELECTRE, TOPSIS, and AHP methods. The DIA method itself is a method based on principles such as the TOPSIS method. This method was developed to improve the previous method, namely the TOPSIS method. Metode DIA, a recent addition to the MADM toolbox, shares similarities with TOPSIS in identifying ideal scenarios for each attribute (Zayat et al., 2023). However, it differs in three key ways: 1) DIA uses the Manhattan distance instead of Euclidean distance to measure closeness to the ideal. 2) It defines the Positive Ideal Alternative (PIA) as having at least the highest positive value *(Dj^+)* for each attribute, not just maximizing the sum of weighted values. 3) It ranks alternatives based on the order of their distance values *(Ri)* from both the positive and negative ideal, offering a potentially more robust ranking approach (Al-Gharabally et al., 2021), following are the steps of the DIA method.

1. Determine the decision matrix with assigned weight.

$$X = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \text{.....................} (1)$$

2. Decision matrix normalization.
   Each element in the matrix is normalized to obtain the normalization matrix ***rij*** which can be calculated as follows:

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \text{..} (2)$$

   So that the normalized R matrix is obtained

$$R = \begin{bmatrix} r_{11} & r_{21} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{bmatrix} \text{..........................} (3)$$

3. Weighting on a normalized matrix.
   After the matrix normalization process then determines the matrix *V*, where each element of the matrix *V* is obtained by calculation:

$$V = W_{ij} * X_{ij} \text{..} (4)$$

   Given the weight *W* = *w₁, w₂, … , wₙ* so that the weight normalized matrix *V* can be produced as follows:

$$V = \begin{bmatrix} w_1 r_{11} & w_2 r_{21} & \cdots & w_n r_{1n} \\ w_1 r_{21} & w_2 r_{22} & \cdots & w_n r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_1 r_{m1} & w_2 r_{m2} & \cdots & w_n r_{mn} \end{bmatrix} \text{...........} (5)$$

4. Determine positive ideal and negative ideal solutions.
   Similar to TOPSIS, DIA establishes the positive and negative ideal attribute values for each attribute. These values represent the maximum and minimum values of the attributes within each column of the MADM matrix. The positive ideal solution is labeled as A^+ and the negative is labeled as A^- :

$$A^+ = max_j v_{ij} = [v_1^+, v_2^+, v_3^+, \dots, v_m^+] \text{..........} (6)$$
$$A^+ = min_j v_{ij} = [v_1^-, v_2^-, v_3^-, \dots, v_m^-] \text{...........} (7)$$

5. Calculate the Manhattan distance for positive and negative attributes.
   While the TOPSIS method employs the positive ideal solution value to compute the Euclidean distance in m-dimensional space between the

solution and the ideal solution, DIA utilizes the Manhattan distance to calculate the distance between the attribute value and both the positive and negative ideal values of each attribute.

$D_j^+ = \sum_{i=1}^{m}[V_{ij} - a_i^+], for\ i = 1,2,3,\ldots,m$ .... (8)

$D_j^- = \sum_{i=1}^{m}[V_{ij} - a_i^-], for\ i = 1,2,3,\ldots,m$ .... (9)

Where V_{ij} is an element of matrix V and a_i is an element of matrix A.

6. Determine Positif Ideal Alternatif (PIA).
   Than, DIA considers the minimum value D^+ and the maximum value D^-

   $\min D^+ = \min D_j^+ = min_j \sum_{i=1}^{m}[V_{ij} - a_i^+]$ .... (10)

   $\max D^- = \max D_j^- = min_j \sum_{i=1}^{m}[V_{ij} - a_i^-]$ .... (11)

   Determine Positif Ideal Alternatif (PIA) which has a minimum D_j^+ , and a maximum of D_j^-, as follows:

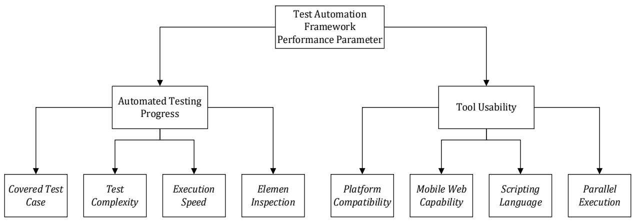
   $PIA = \min(D_j^+), \max(D_j^-)$ .... (12)
7. Perform Rank Identification.
   The ranking can be determined by looking at the *Ri* obtained from the PIA alternative distance as follows:

   $R_i = \sqrt{(D_i^+ - \min(D_i^+)^2 + (D_i^- - \min(D_i^-)^2}$ .... (13)

RESULTS AND DISCUSSION

The analysis process in this study commences with data collection through a literature review, followed by the application of an implementation methodology to identify the target application for automated functional testing. Subsequently, this leads to the formulation of test cases and configuring emulator settings for test preparation. The author utilizes a physical device as the application's working environment, with the Appium server serving as a bridge between the tools and real devices. Through the Appium server, the application ID is obtained, and the application elements utilized in the test case are inspected (Tran et al., 2023). Furthermore, for the implementation of automated functional testing, the Robot Framework, Katalon Studio, and UI Path employ distinct methodologies. The Robot Framework utilizes a console script, while Katalon Studio and UI Path employ a graphical interface.

The complete equation for each stage in The Distance to the Ideal Alternative (DIA) method has been presented in the Literature Review. The author defines parameters and subparameters based on literature studies written by (Aslam et al., 2022; Kozak & Berko, 2022; Prasad et al., 2021). The automatic testing progress parameter has a weight of 0.65, while the tool usability parameter has a weight of 0.35. Figure 2 explains the parameters and sub-parameters as a benchmark for comparison between frameworks.



Source : (Research Result, 2024)

Figure 2. The Parameters And Sub-Parameters As A Criteria

During the assessment of automation software testing, attention is directed towards two primary parameters: Automated Testing Progress and Tool Usability. Below is an explanation of each parameter and sub-parameter.

Table 1. The Parameters And Sub-Parameters Definition

Parameter Sub-Parameter	Definition
Automated Testing Progress	
Covered Test Case	The framework's accuracy in finding elements, so that more tests are passed, thereby minimizing manual testing.
Time Complexity	The total duration of test execution determines whether the framework swiftly executes the multitude of created tests.
Execution Speed	The speed at which the framework performs a command to execute an action. For instance, in a command to click a button.
Elemen Inspector	How each framework carries out inspections to determine the identity of elements, manually or automatically.
Tool Usability	
Platform Compatibility	Framework compatibility for testing applications with various platforms such as Android, IOS, Windows or others.
Mobile Web Capability	The framework possesses the capability to test Web View on application pages.
Scripting Language	Any scripting language applicable for test scripting enhances the flexibility of the framework; the broader the range of programming languages supported within a single framework, the greater its flexibility.
Parallel Execution	Whether the framework has the capability to conduct multiple tests concurrently

Source : (Aslam et al., 2022; Prasad et al., 2021)

Based on the rules of the DIA method, the sum of all weights must be 1, so the weights of each sub-parameter are presented in table 1.

Table 2. Assessment Criteria and Weights

Code	Criteria	Weight	Type
C1	Covered Test Case	0,20	Benefit
C2	Time Complexity	0,15	Cost
C3	Execution Speed	0,10	Cost
C4	Elemen Inspector	0,05	Benefit
C5	Platform Compatibility	0,20	Benefit
C6	Mobile Web Capability	0,15	Benefit
C7	Scripting Language	0,10	Benefit
C8	Parallel Execution	0,05	Benefit

Source : (Research Result, 2024)

After obtaining the *Ri* value and ranking of the two parameters that have been determined, the author will carry out another analysis to compare the values of the two parameters using The Distance To The Ideal Alternative (DIA) method to produce a final value that is more valid for measuring the performance of the test automation framework as a whole. . Furthermore, the ranking results obtained from the comparative analysis of test automation framework performance for functional testing on Playstore-based applications using the DIA method are presented in table 3.

Table 3. Final Ranking Results

Automated Testing Framework	R_i	Rank
Robot Framework	0,00295	3
Katalon Studio	0,00001	1
UI Path	0.00135	2

Source : (Research Result, 2024)

The alternative distance to the Alternative Positive Ideal is called *Ri* . A set of alternatives can be ranked according to the increasing order of *Ri*. The minimum *Ri* value indicates that alternative *Ai* is more selected (Abdulwareth & Al-Shargabi, 2021). Table 3 illustrates the ranking results, with Katalon Studio achieving the top rank with a *Ri* value of 0.00001. This exceptional performance underscores Katalon Studio's dominance in the ranking, as it exhibits the smallest alternative ideal positive distance (*Ri*) value, indicating its proximity to the studied criteria. Katalon Studio's ability to offer comprehensive features for test automation, encompassing test recording, Web, mobile, and API automation, along with robust CI/CD integration, greatly enhances modern software testing practices (Gota et al., 2020). These advantages can be attributed scientifically to a structured approach in software development and provision that caters to the diverse needs of users.

UI Path secures the second position with a *Ri* value of 0.00135, indicating a greater distance from the positive ideal alternative compared to Katalon Studio. UI Path's superiority in GUI-based process automation and robust integration with various testing platforms and environments can be attributed scientifically to a meticulous and adaptive technical approach to evolving test environments and recent technological advancements.

Robot Framework occupies the third position in the ranking with a *Ri* value of 0.00295. Despite being ranked last, Robot Framework's performance can be interpreted scientifically as a result of its focus on flexibility and robust support for various scripting languages, as well as active engagement in a developer community that aids users in overcoming diverse challenges in automated testing.

CONCLUSION

The study conducted by the authors demonstrates a rigorous approach to test case design and functional testing across three prominent automation frameworks. Katalon Studio emerges as the clear frontrunner, securing the top rank with a remarkably low *Ri* value of 0.00001. This near-zero value suggests an exceedingly close proximity to an ideal alternative, indicating Katalon Studio's exceptional performance across various metrics, including automated testing progress and tool usability parameters. UI Path occupies the second position with a *Ri* value of 0.00135, indicating a significant contribution to the testing landscape. Lastly, Robot Framework trails behind with a *Ri* value of 0.00295. Therefore, it can be concluded that Katalon Studio represents an ideal alternative test automation framework regarding performance, considering parameters such as automated testing progress and tool usability.

The authors suggest comparing different test automation frameworks, especially concerning the evaluation of mobile applications, performing functional testing across various platforms, and carrying out functional testing using test automation frameworks that are smoothly integrated into a continuous integration tool like Jenkins. Moreover, they propose exploring additional research paths, such as examining the scalability and adaptability of test automation frameworks across different software development environments. Furthermore, investigating the integration of artificial intelligence and machine learning techniques to improve the effectiveness and precision of automated testing processes would be advantageous.

Additionally, this study should be extended to encompass more intricate test case scenarios while maintaining the same criteria. This approach is essential for yielding innovative findings in the identification of the optimal automated testing framework through the utilization of The Distance To The Ideal Alternative method.

REFERENCE

Abdulwareth, A. J., & Al-Shargabi, A. A. (2021). Toward a Multi-Criteria Framework for Selecting Software Testing Tools. *IEEE Access*, *9*, 158872–158891. https://doi.org/10.1109/ACCESS.2021.3128071

Al-Gharabally, M., Almutairi, A. F., & Salman, A. A. (2021). Particle swarm optimization application for multiple attribute decision making in vertical handover in heterogenous wireless networks. *Journal of Engineering Research (Kuwait)*, *9*(1), 176–187. https://doi.org/10.36909/JER.V9I1.10331

Arya, S., Chitranshi, M., & Singh, Y. (2021). Analysing Distance Measures in Topsis: A Python-Based Tool. 275–292. https://doi.org/10.1007/978-981-16-1528-3_24

Aslam, Z., Ayub, N., Ali, M., Zubair, S., & Naz, A. (2022). Performance-Based Analysis Of Test Automation Tools For Android Applications. *Researchgate.Net*. https://doi.org/10.17605/OSF.IO/D3BHQ

Baktha, K. (2020). Evaluating the Performance and Capabilities of Popular Android Mobile Application Testing Automation Frameworks in Agile/DevOps Environment. https://www.diva-portal.org/smash/record.jsf?pid=diva2:1471376

Berihun, N. G., Dongmo, C., & Van der Poll, J. A. (2023). The Applicability of Automated Testing Frameworks for Mobile Application Testing: A Systematic Literature Review. *Computers*, *12*(5). https://doi.org/10.3390/computers12050097

Chakraborty, S. (2022). TOPSIS and Modified TOPSIS: A comparative analysis. *Decision Analytics Journal*, *2*, 100021. https://doi.org/10.1016/j.dajour.2021.100021

Gota, L., Gota, D., & Miclea, L. (2020, May). Continuous Integration in Automation Testing. In 2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR) (pp. 1-6). IEEE. https://doi.org/10.1109/AQTR49680.2020.9129990

Karlsson, S., Čaušević, A., Sundmark, D., & Larsson, M. (2021, April). Model-based automated testing of mobile applications: an industrial case study. In 2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) (pp. 130-137). IEEE. https://doi.org/10.1109/ICSTW52544.2021.00033

Kozak, I., & Berko, A. (2022, November). Three-module framework for automated software testing. In 2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT) (pp. 454-457). IEEE. https://doi.org/10.1109/CSIT56902.2022.10000806

Lin, J. W., Salehnamadi, N., & Malek, S. (2020, December). Test automation in open-source android apps: A large-scale empirical study. In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (pp. 1078-1089). https://doi.org/10.1145/3324884.3416623

Menegassi, A. A., & Endo, A. T. (2020). Automated tests for cross-platform mobile apps in multiple configurations. *IET Software*, *14*(1), 27–38. https://doi.org/10.1049/iet-sen.2018.5445

Prasad, L., Yadav, R., & Vore, N. (2021). A Systematic Literature Review of Automated Software Testing Tool. *Lecture Notes in Networks and Systems*, *167*, 101–123. https://doi.org/10.1007/978-981-15-9712-1_10

Salam, M. A., Taha, S., & Hamed, M. G. (2022, October). Advanced Framework for Automated Testing of Mobile Applications. In 2022 4th Novel Intelligent and Leading Emerging Sciences Conference (NILES) (pp. 233-238). IEEE. https://doi.org/10.1109/NILES56402.2022.9942374

Tran, H. M., Ninh, T. D., Tran, T. D., Van Ngo, V., & Nguyen, L. D. (2023, October). Automation Testing with Appium Framework in IP Multimedia Subsystem. In 2023 14th International Conference on Information and Communication Technology Convergence (ICTC) (pp. 579-582). IEEE. https://doi.org/10.1109/ICTC58733.2023.10392322

Zayat, W., Kilic, H. S., Yalcin, A. S., Zaim, S., & Delen, D. (2023). Application of MADM methods in Industry 4.0: A literature review. *Computers & Industrial Engineering*, *177*, 109075. https://doi.org/10.1016/J.CIE.2023.109075