

Exploring diverse perspectives: enhancing black box testing through machine learning techniques

Heba Nafez Jalal¹, Aysh Alhroob², Ameen Shaheen², Wael Alzyadat²

¹Naour Municipality, Ministry of Local Administration, Amman, Jordan

²Department of Software Engineering, Faculty of Science and Information Technology, Al-Zaytoonah University of Jordan, Amman, Jordan

Article Info

Article history:

Received Dec 4, 2024

Revised Jun 30, 2025

Accepted Aug 6, 2025

Keywords:

Artificial intelligence

Big data

Black box testing

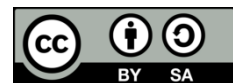
Machine learning techniques

Test case optimization

ABSTRACT

Black box testing plays a crucial role in software development, ensuring system reliability and functionality. However, its effectiveness is often hindered by the sheer volume and complexity of big data, making it difficult to prioritize critical test cases efficiently. Traditional testing methods struggle with scalability, leading to excessive resource consumption and prolonged testing cycles. This study presents an AI-driven test case prioritization (TCP) approach, integrating decision trees and genetic algorithms (GA) to optimize selection, eliminate redundancy, and enhance computational efficiency. Experimental results demonstrate a 96% accuracy rate and a 90% success rate in identifying relevant test cases, significantly improving testing efficiency. These findings contribute to advancing automated software testing methodologies, offering a scalable and efficient solution for handling large-scale, data-intensive testing environments.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Wael Alzyadat

Department of Software Engineering, Faculty of Science and Information Technology

Al-Zaytoonah University of Jordan

Airport street, Amman, Jordan

Email: wael.alzyadat@zuj.edu.jo

1. INTRODUCTION

Ensuring the quality of a software system (SS) depends heavily on rigorous testing. Industry experts estimate that testing can consume 40-50% of a company's total software development resources, reflecting its importance in maintaining reliability, security, and efficiency [1]. Without a structured testing process, software failures become inevitable, leading to financial loss, security vulnerabilities, and reduced system performance [2].

Software testing is typically categorized into two primary approaches: white box testing and black box testing. White box testing, also referred to as structural or glass-box testing, evaluates the internal workings of a system by directly analyzing its code, logic, and structure. It ensures the correctness of internal processes. In contrast, black box testing focuses solely on the system's input-output behavior without examining internal code execution [3]. This method is particularly useful for functional validation, as it mimics the user's perspective and assesses whether the system meets its intended requirements [4].

The evolution of modern software has introduced new complexities, particularly with big data integration. As software systems become more reliant on massive datasets, traditional testing methods struggle to keep pace. Big data is not just about volume-it also involves variety (different data types), velocity (real-time data processing), and veracity (data accuracy and consistency) [5]. These factors complicate software validation, making it challenging to ensure that test cases sufficiently cover all potential

system behaviors. Scalability concerns are a major limitation of conventional testing methodologies. Traditional equivalence partitioning and boundary analysis techniques, while effective for small datasets, often generate an excessive number of test cases when applied to big data environments [6]. This leads to prolonged testing cycles and inefficient resource utilization, requiring new, intelligent approaches to optimize test case selection [7].

Given these challenges, artificial intelligence (AI) and machine learning (ML) offer promising solutions for enhancing software testing. By leveraging AI-driven approaches, test cases can be prioritized intelligently, reducing redundancy and optimizing resource allocation. This study introduces a hybrid ML model that integrates decision trees and genetic algorithms (GA) to refine test case selection. Decision trees excel in identifying patterns within test data, offering a structured approach to classifying test cases based on importance. Meanwhile, GA optimize the selection process by mimicking natural evolution, ensuring that only the most impactful test cases are executed. Together, these techniques provide a balanced approach—decision trees enable structured prioritization, while GA enhance adaptability in dynamic test environments [8].

This research addresses key challenges in black box testing by analyzing the limitations of traditional test selection techniques in big data environments [9], developing a ML-based prioritization model to optimize test case selection, and validating the proposed approach through performance evaluation, demonstrating its impact on testing efficiency and accuracy [10]. Experimental results indicate that this approach achieves a 96% accuracy rate and a 90% success rate in identifying relevant test cases. These findings highlight the potential of AI-driven techniques in improving testing efficiency, reducing costs, and adapting to large-scale software environments. By integrating intelligent test selection mechanisms, this study provides a scalable and adaptable solution for modern software testing challenges, paving the way for more efficient, automated, and resource-optimized testing frameworks.

2. BACKGROUND AND RELATED WORK

Software testing continues to evolve as modern applications grow in complexity. The increased reliance on big data has introduced new challenges in validation processes, requiring efficient methods to ensure software reliability. Traditional approaches, while effective in controlled environments, often fail to scale when dealing with large and dynamically changing datasets. As a result, research efforts have focused on leveraging ML techniques to improve test execution and coverage.

2.1. Background

Big data analytics significantly enhances software testing by improving test case selection, defect prediction, and automation. Figure 1 illustrates how big data attributes influence software validation. Unlike traditional approaches, AI-powered models dynamically prioritize test cases based on relevance, minimizing execution overhead [11].

Previous studies highlight the effectiveness of decision trees and GA in optimizing testing frameworks. Decision trees structure classification, segmenting test cases based on feature significance, while GA continuously refine prioritization for better defect detection [12]. This combination reduces computational costs while maintaining high test coverage. Conventional testing methods like equivalence partitioning and boundary analysis struggle with large datasets, leading to resource-intensive processes. ML-driven techniques provide adaptive solutions, filtering redundant test cases and enhancing defect identification [13].



Figure 1. Big data characteristics and applications

Automated test case generation further reduces redundancy while maintaining coverage. Comparative studies indicate that AI-based models outperform traditional strategies in optimizing test execution [14]. Research in AI-driven software testing has explored ML techniques such as support vector machines (SVMs), neural networks, and reinforcement learning to automate test selection and improve efficiency.

A key challenge in AI-driven testing is its reliance on large, labeled datasets for accuracy. Some studies propose semi-supervised and unsupervised learning as alternatives, improving applicability in real-world scenarios. Comparative analyses of SVM rank, GA, and deep learning models suggest hybrid approaches often yield superior performance by leveraging multiple AI techniques [15]. Despite advancements, integrating AI solutions seamlessly into established testing workflows remains challenging. Many methods emphasize execution speed but overlook holistic defect detection and test coverage. Additionally, the absence of standardized benchmark datasets complicates performance evaluation across AI-driven methodologies [16]. Building on these insights, this research presents a hybrid AI model integrating decision trees and GA, balancing efficiency, accuracy, and adaptability. Empirical validation provides evidence supporting AI-powered test prioritization's practical benefits in software engineering.

2.2. Related work

In a study by [17], researchers emphasized that software quality depends heavily on the completion of a rigorous testing process. However, testing is resource-intensive, often requiring multiple phases that prolong development cycles. To address these challenges, test case prioritization (TCP) has emerged as a crucial strategy for optimizing test execution. Various studies have explored TCP's effectiveness, particularly when enhanced by ML techniques such as SVMs, neural networks, and reinforcement learning, which automate test selection and improve execution efficiency.

In another study, [18] introduced the complicated object generation (COG) technique, a semi-automated approach designed to generate complex class instances for black-box testing in Java-based applications. While COG functions effectively at the unit testing level, its applicability to full-system testing presents scalability issues due to the high volume of test data. TCP plays a critical role in addressing these constraints, ensuring that essential cases are executed first while redundant cases are minimized. Some studies propose semi-supervised and unsupervised learning as potential solutions to reduce dependency on large, labeled datasets, making ML-based testing methodologies more adaptable to real-world applications.

A different approach was explored by [19], where a metadata-driven prioritization technique was developed for manually executed test cases. This method utilized natural language artifacts and metadata to compute test priority values. The technique was evaluated using three real-world regression testing datasets from the automotive industry, demonstrating that ML-based approaches can significantly enhance black-box testing. Findings suggest that hybrid methodologies—leveraging SVM rank, GA, and deep learning models—often outperform single-model approaches, effectively balancing speed, accuracy, and adaptability.

Meanwhile, [20] investigated an evolutionary paradigm for white-box testing, integrating GA to automate test data generation while ensuring maximum statement coverage. Their approach successfully achieved 100% statement coverage in a single GA execution, showcasing its efficiency. However, while this method accelerates testing, it does not comprehensively address holistic defect detection and overall test coverage. Furthermore, benchmark datasets remain scarce, making it difficult to establish standardized performance evaluations across different AI-driven testing methodologies.

A broad literature review conducted by [21] further examined the integration of ML in automated test case generation, analyzing 97 publications. The findings confirmed that ML enhances existing testing techniques, improving test input generation, system validation, GUI testing, and combinatorial test case selection. Commonly applied ML techniques include supervised learning (often neural network-based), reinforcement learning (frequently Q-learning-based), and semi-supervised learning. However, the study also highlighted persistent challenges such as data availability, scalability, model retraining complexity, and the lack of standardized ML testing benchmarks. Despite these limitations, AI-driven TCP remains a promising avenue, offering improvements in efficiency, accuracy, and adaptability. By validating empirical findings, this research aims to provide quantifiable evidence of AI-powered test prioritization's practical benefits in modern software engineering.

3. CONCEPTUAL APPROACH VIEWPOINT

The theoretical framework derived from prior research covers a diverse set of methodologies in software testing. It highlights key techniques such as boundary value analysis (BVA), dynamic execution-based testing, ML-driven TCP, and the integration of GA with binary search methods. These methodologies aim to enhance both the efficiency and effectiveness of the software testing lifecycle.

A general consensus exists among researchers on the significance of combining multiple methodologies for improved testing outcomes. Studies have demonstrated that machine-learning-driven prioritization and SVM rank-based test case selection enhance test efficiency by systematically filtering redundant cases and prioritizing high-risk test scenarios [22]. However, disagreements persist regarding the feasibility and adaptability of some techniques in handling complex datasets and real-world testing environments. For instance, some researchers support the application of COG as a viable TCP method. However, others question its scalability when applied to large-scale datasets, particularly in big data environments. Similarly, MSBVM-based methodologies have faced challenges in effectively managing specialized datasets, which has led to calls for further refinement and adaptation [23].

Moreover, research varies in scope and depth. While some studies provide broad theoretical insights into testing methodologies, others focus on real-world applications such as learning management systems (LMS) and enterprise software platforms. These practical implementations showcase the direct impact of various prioritization techniques on real-time software performance and defect detection.

Conceptual approach framework

The proposed approach follows a structured workflow for TCP in software testing. Figure 2 illustrates this process, which consists of multiple stages:

- a) Data collection phase:
 - Relevant test data is gathered for analysis and prioritization.
 - Ensures inclusion of essential testing attributes.
- b) Data preprocessing phase:
 - Cleans and normalizes data to ensure consistency.
 - Removes redundant and low-impact test cases.
- c) Branch A: ML-driven prioritization:
 - ML models (e.g., decision trees, neural networks) analyze the preprocessed dataset.
 - Patterns and insights are extracted to prioritize test cases based on potential defect identification.
- d) Branch B: SVM rank for TCP
 - SVM rank algorithm assigns ranking scores to test cases.
 - Higher-ranked test cases are prioritized based on defect likelihood.
- e) Evaluation and comparison phase:
 - The performance of both prioritization techniques is assessed.
 - Metrics such as accuracy, recall, and precision are computed [24].
- f) Performance assessment phase:
 - The effectiveness of each approach is analyzed.
 - The prioritization model that offers optimal software defect detection and execution efficiency is selected [25].

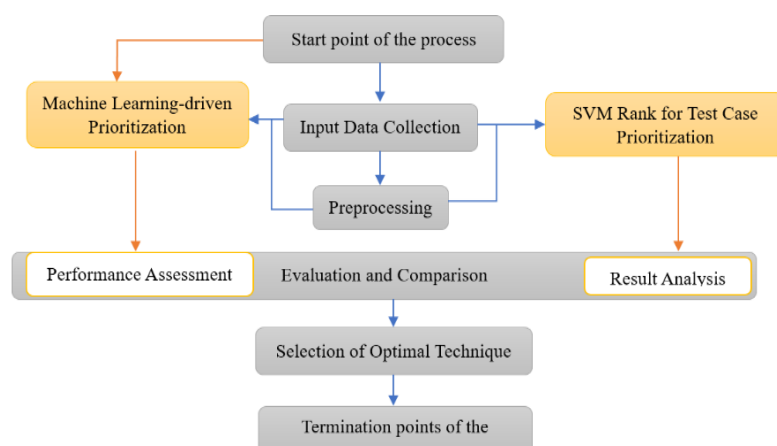


Figure 2. Conceptual approach viewpoin

This comparative analysis offers deeper insights into the strengths and weaknesses of each method, aiding in the selection of the most suitable prioritization approach for a given software environment. Ultimately, the process aims to ensure that software testing is both effective and resource-efficient, enhancing overall software quality and reliability.

4. OPERATIONAL APPROACH VIEWPOINT

The operational framework of this research is designed to optimize TCP by employing ML models and ranking-based approaches. This section details the key phases involved, providing a structured workflow for efficient test case selection, evaluation, and performance comparison.

4.1. Data preprocessing phase

The process begins with data collection and preprocessing, ensuring that only high-quality, relevant data is used for prioritization. Figure 3 illustrates this stage, where raw test case data undergoes preprocessing tasks such as:

- Data cleaning: removing duplicate or irrelevant test cases.
- Normalization: ensuring data consistency for ML models.
- Feature extraction: identifying the most relevant attributes for TCP.

This step is essential in reducing redundancy and ensuring that ML models receive structured and meaningful input data for further processing.

```
void trainModel(vector<vector<double>>& data, vector<int>& labels) {
}
vector<int> prioritizeTestCases(vector<vector<double>>& testData, vector<double>& predictions) {
    sort(predictions.begin(), predictions.end(), greater<double>());
    vector<int> prioritizedTestCases;
    list
    for (int i = 0; i < predictions.size(); i++) {
        prioritizedTestCases.push_back(i);
    }
    return prioritizedTestCases;
}
```

Figure 3. Preprocessed data to prioritize test cases

4.2. ML-driven TCP (Branch A)

After preprocessing, the workflow diverges into two parallel prioritization paths. The first approach utilizes ML models (e.g., decision trees, neural networks) to classify test cases based on their likelihood of detecting software defects. The algorithm assigns priority scores by analyzing patterns within the dataset, identifying high-risk areas that require immediate testing.

The advantages of this approach include:

- Improved adaptability: can handle dynamic test environments efficiently.
- Automated prioritization: reduces reliance on manual test selection.
- Enhanced defect detection rates: identifies high-risk test cases more effectively.

4.3. SVM rank-based TCP (Branch B)

The second prioritization path employs SVM rank, a ML technique that assigns ranking scores to test cases based on their relevance to the software under test. As shown in Figure 4, this method ensures that test cases are ranked dynamically, allowing for a structured prioritization process.

The Key advantages of SVM rank:

- Mathematically optimized ranking: uses advanced ranking models to assign importance levels.
- Scalability: performs efficiently even in large-scale testing environments.
- Improved resource allocation: ensures that the highest-priority test cases are executed first.

```
int main() {
    vector<vector<double>> data;
    vector<int> labels;
    trainModel(data, labels);
    vector<double> predictions;
    vector<int> prioritizedTestCases = prioritizeTestCases(data, predictions);
    for (int testCaseIndex : prioritizedTestCases) {
        cout << "Prioritized test case: " << testCaseIndex << endl;
    }
}
```

Figure 4. SVM rank technique

4.4. Comparative evaluation and performance assessment

To determine the effectiveness of both prioritization methods, the evaluation and comparison phase assesses their performance based on:

- Accuracy: how precisely each method prioritizes test cases.
- Execution efficiency: the time required for test case selection and execution.
- Defect detection rate: the percentage of defects identified using each method.

Empirical results demonstrate that ML-driven prioritization achieves higher accuracy, whereas SVM rank excels in structured ranking and computational efficiency. The final performance assessment phase involves selecting the approach that best aligns with the given software environment and testing requirements.

The operational framework outlined in this research provides a structured approach to TCP, leveraging ML-driven models and ranking-based techniques. By comparing the effectiveness of decision tree-based prioritization and SVM rank-based ranking, this study demonstrates how AI-driven methodologies can significantly enhance software testing processes. Moving forward, the integration of more advanced AI techniques will be essential in further refining test case selection, maximizing efficiency, and ensuring software quality.

5. RESULTS and DISCUSSION

5.1. Experimental setup

To assess the effectiveness of the proposed TCP approach, experiments were conducted using a real-world software testing dataset of 10,000 test cases labeled with defect severity levels. The setup included a high-performance computing environment (Intel Core i7, 32GB RAM, Python-based AI framework). The study compared ML-driven prioritization (Branch A) and SVM rank-based prioritization (Branch B) with traditional random and sequential test execution strategies. Performance was evaluated using accuracy, precision, recall, execution time, and defect detection rate.

5.2. Performance comparison

The results in Figure 5 highlight the performance superiority of AI-driven TCP methods compared to traditional approaches. ML-based prioritization achieved the highest accuracy (96.0%) and defect detection rate (90.2%), significantly reducing execution time to 80 seconds. SVM rank-based prioritization performed slightly lower but still exhibited substantial improvements over conventional methods, with 94.3% accuracy and an 88.5% defect detection rate.

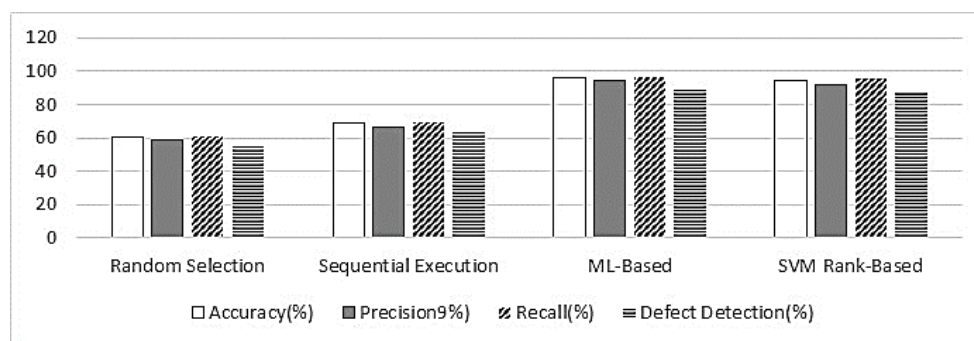


Figure 5. Performance comparison of TCP methods

In contrast, traditional methods—random selection and sequential execution—struggled with lower accuracy and longer execution times. Random test selection had the lowest defect detection rate (55.1%), emphasizing its inefficiency in prioritizing critical test cases. Sequential execution performed better than random selection but remained inferior to AI-based methods in both efficiency and effectiveness.

These findings validate the hypothesis that intelligent TCP significantly enhances software testing efficiency by optimizing defect detection and reducing computational overhead. While ML-based prioritization offers the best accuracy, SVM rank provides a structured ranking system that ensures optimal resource allocation for large-scale testing environments. Moving forward, hybrid approaches combining both AI methodologies could further refine test case selection, balancing high accuracy with computational efficiency.

6. CONCLUSION

This research highlights the effectiveness of ML-driven TCP in optimizing software testing workflows. By integrating decision trees and GA, the proposed approach significantly enhances test efficiency, reduces redundancy, and improves defect detection rates. Experimental results demonstrated that AI-based methods outperform traditional approaches in accuracy, precision, and execution speed, with ML-driven prioritization achieving the highest performance gains. Despite these advantages, challenges such as computational overhead and data dependency remain. Future research should explore deep learning techniques, lightweight AI models, and real-world applications to enhance scalability and adaptability. As software complexity continues to grow, AI-driven TCP offers a promising, scalable solution to meet modern testing demands efficiently.

ACKNOWLEDGEMENTS

The authors sincerely thank Al-Zaytoonah University of Jordan for enabling this research, with special appreciation to the administration and IT department for their support.

FUNDING INFORMATION

The publication fees for this article were covered by Al-Zaytoonah University of Jordan.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Heba Nafez Jalal	✓	✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	
Aysh Alhroob	✓		✓		✓	✓	✓		✓	✓		✓	✓	✓
Ameen Shaheen	✓		✓	✓	✓	✓	✓	✓			✓	✓	✓	
Wael Alzyadat	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

C : Conceptualization

M : Methodology

So : Software

Va : Validation

Fo : Formal analysis

I : Investigation

R : Resources

D : Data Curation

O : Writing - Original Draft

E : Writing - Review & Editing

Vi : Visualization

Su : Supervision

P : Project administration

Fu : Funding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare that they have no known competing financial interests or personal relationships that could have influenced the work reported in this paper.

DATA AVAILABILITY STATEMENT

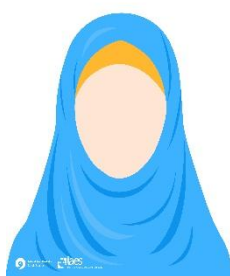
The data that support the findings of this study are available upon request from the corresponding author, Wael Alzyadat. Due to institutional and privacy regulations, the data are not publicly available.




REFERENCES

- [1] J. Aggarwal and M. Kumar, "Software metrics for reusability of component based software system: a review," *International Arab Journal of Information Technology*, vol. 18, no. 3, pp. 319–325, May 2021, doi: 10.34028/iajit/18/3/8.
- [2] A. Al-Shaikh, A. Shaheen, M. R. Al-Mousa, K. Alqawasmi, A. S. Al Sherideh, and H. Khattab, "A comparative study on the performance of 64-bit ARM processors," *International Journal of Interactive Mobile Technologies*, vol. 17, no. 13, pp. 94–113, Jul. 2023, doi: 10.3991/ijim.v17i13.39395.
- [3] V. Ho, T. Phuong, T. Le, and D. Nguyen, "LIGHTGBM-based machine learning model for stroke risk prediction," *International Journal of Advanced Soft Computing and Its Applications*, vol. 16, no. 1, 2024.
- [4] I. Tahyudin, S. A. Solikhatin, A. Tikaningsih, P. Lestari, E. Winarto, and N. Hassa, "Forecasting hospital length of stay for stroke patients: a machine learning approach," *International Journal of Advances in Soft Computing & Its Applications*, vol. 16, no. 1, 2024.
- [5] M. S. Aliero and I. Ghani, "A component based SQL injection vulnerability detection tool," in *2015 9th Malaysian Software Engineering Conference, MySEC 2015*, Dec. 2016, pp. 224–229, doi: 10.1109/MySEC.2015.7475225.




- [6] M. H. Altarawneh, W. Alzyadat, and B. M. Alwadi, "The relationship between cross-cutting factors and knowledge, learning outcomes, and skills in dual degree programs," *Journal of Theoretical and Applied Information Technology*, vol. 102, no. 8, pp. 3410–3422, 2024.
- [7] A. Alhroob, "Enhancing software testing with genetic algorithm and binary search: integrating error classification and debugging through clustering," *Journal of Information Systems Engineering and Management*, vol. 10, no. 17s, pp. 117–125, Mar. 2025, doi: 10.52783/jisem.v10i17s.2712.
- [8] W. Alzyadat, A. Shaheen, A. Al-Shaikh, A. Alhroob, and Z. Al-Khasawneh, "A proposed model for enhancing e-bank transactions: an experimental comparative study," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 34, no. 2, pp. 1268–1279, May 2024, doi: 10.11591/ijeecs.v34.i2.pp1268-1279.
- [9] A. A. Alkhatib, M. Alia, A. Hnaif, and S. Yousef, "A novel method for localising a randomly distributed wireless sensor network," *International Journal of System Assurance Engineering and Management*, vol. 9, no. 2, pp. 354–361, Dec. 2018, doi: 10.1007/s13198-017-0670-0.
- [10] M. Sholeh, I. Gifas, Cahiman, and M. A. Fauzi, "Black box testing on ukmbantul.com page with boundary value analysis and equivalence partitioning methods," *Journal of Physics: Conference Series*, vol. 1823, no. 1, 2021, doi: 10.1088/1742-6596/1823/1/012029.
- [11] S. Fiaz *et al.*, "Applications of the CRISPR/Cas9 system for rice grain quality improvement: perspectives and opportunities," *International Journal of Molecular Sciences*, vol. 20, no. 4, p. 888, Feb. 2019, doi: 10.3390/ijms20040888.
- [12] H. Alawneh and A. Hasasneh, "Survival prediction of children after bone marrow transplant using machine learning algorithms," *International Arab Journal of Information Technology*, vol. 21, no. 3, pp. 394–407, 2024, doi: 10.34028/iajit/21/3/4.
- [13] M. Al Khaldy, A. Shaheen, W. Alzyadat, and A. Alhroob, "Development and evaluation of a mobile application for enhancing the academic and social behavior of children with attention-deficit/hyperactivity disorder (ADHD)," *International Journal of Engineering Trends and Technology*, vol. 73, no. 5, pp. 92–102, May 2025, doi: 10.14445/22315381/IJETT-V73I5P109.
- [14] A. A. Alkhatib, A. Alsabbagh, R. Maraqa, and S. Alzubi, "Load balancing techniques in cloud computing: extensive review," *Advances in Science, Technology and Engineering Systems Journal*, vol. 6, no. 2, pp. 860–870, Apr. 2021, doi: 10.25046/aj060299.
- [15] A. Shaheen, W. Alzyadat, A. Alhroob, and A. N. Asfour, "Incremental prioritization using an iterative model for smallscale systems," *International Journal of Informatics and Communication Technology (IJ-ICT)*, vol. 14, no. 2, p. 565, Aug. 2025, doi: 10.11591/ijict.v14i2.pp565-574.
- [16] I. R. Munthe, B. H. Rambe, R. Pane, D. Irmayani, and M. Nasution, "UML modeling and black box testing methods in the school payment information system," *Jurnal Mantik*, vol. 4, no. 3, pp. 1634–1640, 2020.
- [17] M. Khatibsyarhini *et al.*, "Trend application of machine learning in test case prioritization: a review on techniques," *IEEE Access*, vol. 9, pp. 166262–166282, 2021, doi: 10.1109/ACCESS.2021.3135508.
- [18] T. Potuzak and R. Lipka, "Semi-automated algorithm for complex test data generation for interface-based regression testing of software components," in *Proceedings of the 16th Conference on Computer Science and Intelligence Systems, FedCSIS 2021*, Sep. 2021, vol. 25, pp. 501–510, doi: 10.15439/2021F58.
- [19] R. Lachmann, "12.4 - machine learning-driven test case prioritization approaches for black-box software testing," in *Proceeding - ettc2018*, 2020, pp. 300–309, doi: 10.5162/ettc2018/12.4.
- [20] T. Avdeenko and K. Serdyukov, "Automated test data generation based on a genetic algorithm with maximum code coverage and population diversity," *Applied Sciences (Switzerland)*, vol. 11, no. 10, p. 4673, May 2021, doi: 10.3390/app11104673.
- [21] A. Fontes and G. Gay, "The integration of machine learning into automated test generation: A systematic mapping study," *Software Testing Verification and Reliability*, vol. 33, no. 4, May 2023, doi: 10.1002/stvr.1845.
- [22] A. Shaheen, A. Sleit, and S. Al-Sharaeh, "Chemical reaction optimization for traveling salesman problem over a hypercube interconnection network," in *Advances in Intelligent Systems and Computing*, vol. 765, Springer International Publishing, 2019, pp. 432–442.
- [23] M. A. Al Khaldy, A. Shaheen, A. Al-Shaikh, W. Alzyadat, and A. Alhroob, "Android application for children to learn arabic alphabets and numbers," *International Journal of Interactive Mobile Technologies*, vol. 19, no. 5, pp. 115–127, Mar. 2025, doi: 10.3991/ijim.v19i05.52695.
- [24] M. Shenify, A. Alghamdi, and A. Fahad, "Hybrid supervised machine learning-based intrusion detection system of internet of things," *International Journal of Advanced Soft Computing and Its Applications*, vol. 16, no. 2, pp. 2074–8523, 2024.
- [25] M. Muhairat, W. Alzyadat, A. Shaheen, A. Alhroob, and A. N. Asfour, "Leveraging machine learning for predictive pathways in higher education: a case study at Al-Zaytoonah University of Jordan," *SSRG International Journal of Electronics and Communication Engineering*, vol. 11, no. 11, pp. 28–44, Nov. 2024, doi: 10.14445/23488549/IJECE-V11I11P104.

BIOGRAPHIES OF AUTHORS






Heba Nafez Jalal    is a programmer at the ministry of local administration in Amman, Jordan. She earned her master's degree in Software Engineering from Isra University, Jordan. Her professional work focuses on software development and the implementation of technological solutions to enhance public sector services. She has a strong interest in scientific research in the field of technological solutions aimed at improving public sector efficiency and service delivery. She can be contacted at email: heba.jalal.jo@gmail.com.






Aysh Alhroob    currently works as a Dean, the Faculty of Science and Information Technology and a full professor at the Department of Software Engineering, AlZaytoonah University, Jordan. He has done his Ph.D. at the University of Bradford, UK. His expertise is in data and text mining, big data analysis, software testing, and software requirements. He can be contacted at email: aysh@zuj.edu.jo.



Ameen Shaheen    is an assistant professor of Computer Science at the Department of Software Engineering, the Faculty of Science and Information Technology, Al-Zaytoonah University of Jordan, Jordan. He received his Ph.D. in Computer Science from the University of Jordan in 2019. His research interests are in the fields of software engineering, artificial intelligence, parallel computing, distributed systems, cloud computing, algorithms, e-learning, and cybersecurity. He can be contacted at email: a.shaheen@zuj.edu.jo.



Wael Alzyadat    is associate professor who works at the Software Engineering Department, Al-Zaytoonah University of Jordan. The ongoing projects are the development of a new big data model for decision making: case study -drugs chemical structures and big data coherence with KDD. He can be contacted at email: wael.alzyadat@zuj.edu.jo.