International Journal for Modern Trends in Science and Technology
Volume 10, Issue 01, pages 117-124.
ISSN: 2455-3778 online
Available online at: http://www.ijmtst.com/vol10issue01.html
DOI: https://doi.org/10.46501/IJMTST1001016

Generative AI–Based Test Case Synthesis for Oracle ERP Release Validation

Rama Sanjeeva Reddy Vanipenta

New Jersey, USA 08817
ramasanjeevareddy.v@gmail.com

To Cite this Article
Rama Sanjeeva Reddy Vanipenta, Generative AI–Based Test Case Synthesis for Oracle ERP Release Validation, International Journal for Modern Trends in Science and Technology, 2024, 10(01), pages. 117-124. https://doi.org/10.46501/IJMTST1001016

Article Info
Received: 12 January 2024; Accepted: 28 January 2024; Published: 02 February 2024.

ABSTRACT

Oracle releases new updates with enhancements and bug fixes and are tested thoroughly before being released as it will impact on the existing business processes. Test case generation and regression testing can be time consuming, resource intensive and difficult. The significant contribution of this work is the Generative AI–Based Test Case Synthesis Framework for Oracle ERP Release Validation, which leverages Large Language Models (LLMs) to automatically generate, prioritize, and evaluate test cases from release notes, requirements and repository of historical test cases. It also features semantic similarity analysis, AI-driven test generation, and risk-based prioritization, which all contribute to more efficient and comprehensive testing. Experimental results show that the proposed approach is superior to current testing techniques in terms of test coverage, precision, recall and F1-score with reasonable reduction in test execution time. The results indicate that Generative AI can be a valuable resource in automating Oracle ERP regression testing, software quality assurance, and release validation processes.

Keywords: Generative Artificial Intelligence (GenAI), Oracle ERP, Test Case Generation, Large Language Models (LLMs), Release Validation.

1. INTRODUCTION

Enterprise Resource Planning (ERP) systems are now integral to modern business, connecting all aspects of the enterprise, including the finance, human resource, procurement, supply chain management, and customer service functions and processes, into a single application. Oracle ERP is popular among the different ERP solutions because of its ability to handle complex business processes, reliability, and scalability. Oracle regularly updates its software to stay competitive and meet changing business needs, adding new features, improving performance, fixing bugs and security vulnerabilities.

Newly introduced changes in each Oracle ERP release must be carefully validated to ensure that there is no negative impact on the prevailing business processes. The validation of the release is mainly based on regression testing (re-running tests that were already performed to check system stability). The traditional approach to creating and running test cases, however, is largely manual and a significant testing team effort. With the increasing complexity of ERP, manual testing

becomes more and more costly, time consuming and is subject to human error. These difficulties can cause software to be delayed for release and cost to become higher for operations.

The introduction of new forms of Artificial Intelligence (AI), such as Generative Artificial Intelligence (GenAI) and Large Language Models (LLMs), has revolutionized a number of software engineering tasks including code generation, defect detection, and automated software testing. Generative AI can handle massive amounts of text data, interpret software needs, and produce meaningful text-based results from context. These features can be valuable in the context of automated generation of test cases and enhancement of software quality assurance workflows.

There are several research works that have shown that LLMs are very effective in creating regression tests, requirement-based test cases and unit tests. While there have been significant strides, there are relatively few studies that specifically address the use of Generative AI for Oracle ERP release validation. Oracle ERP environments are complex because of their workflows, number of business rules and the extensive regression testing requirements. So, an intelligent framework to generate accurate and comprehensive test cases specific to Oracle ERP systems is needed.

To address this, this paper proposes a Generative AI – Based Test Case Synthesis Framework for Oracle ERP Release Validation that uses generative AI to automatically generate test cases. The proposed framework is based on the automatic generation, evaluation and prioritization of test cases using LLM with the help of semantic requirement analysis and risk assessment techniques. The framework will use release notes, requirement documents, business rules, and historical testing repositories to enhance the overall quality of software, speed up release validation, lessen manual effort, and maximize test coverage.

The major contribution of this work is to develop an AI based framework which will automate the regression testing in Oracle ERP with high accuracy and high efficiency. The proposed solution aims to offer a scalable solution to organizations that are managing increasingly complex ERP release cycles and ensuring reliable software deployments.

2. LITERATURE SURVEY

Testing software requirements is an important aspect of software quality and reliability. Dos Santos et al. [1] have systematically reviewed the literature for software requirements testing approaches and presented a list of software requirement testing techniques that are used to validate and verify requirements throughout software development lifecycle. They focused on the need for early defect detection and identified issues with maintaining test coverage and addressing changing needs.

The application of machine learning in the software testing has been a growing phenomenon. Pan et al. [2] conducted a systematic review of test case selection and prioritization techniques based on machine learning. They discovered that by leveraging machine learning, testing efforts and testing execution time could be greatly decreased, and effectiveness of fault detection could be significantly enhanced. The effectiveness of these approaches, however, will rely on the availability of high quality historical testing data.

Large Language Models (LLMs) have been seen as potential solutions in software engineering. Fan et al. [3] provided a survey of LLMs used for different software engineering tasks, such as code generation, testing, debugging, documentation, etc. The study revealed significant potential for productivity improvements and noted issues with reliability, explainability, and hallucinated outputs.

LLMs are now being employed to generate automated unit tests. Schäfer et al. [4] empirically demonstrated the success of using LLMs to create unit tests and concluded that such tests can provide reasonable code coverage and save manual effort. Tests generated, however, can be logical incorrect and frequently need human validation prior to deployment.

ERP systems continue to play a significant role in enterprise operations management. QodeNext [5] talked about the role of ERP in optimising the business processes with various organizational functions, data visibility and better decision making abilities. However, ERP systems can be complex to implement, require customization, and have testing requirements.

Manual testing is still popular because of its flexibility and the fact that it can be used to detect usability problems. Global App Testing [6] mentioned the benefits of manual testing such as intuition and exploratory

testing. Manual testing is, however, time consuming and labor intensive, and hard to scale for large enterprise systems.

Likewise, SaiSuBha Tech Ltd. [7] have studied the advantages and disadvantages of manual testing. Their research highlighted that manual testing works well with user experience evaluation and exploratory scenarios, but has limited repeatability, inconsistencies and is more expensive in large scale projects.

In many aspects, automated testing offers a number of benefits with regard to "efficiency" and "reproducibility". Devzery [8] discussed some hidden issues with automated testing such as high cost of initial configuration, maintenance of automated testing, and requirement changes of the application rapidly. These constraints indicate that automation solutions may not be able to completely solve testing issues.

Melo and Freire [9] researched the factors that hinder using automated software testing. They identified issues for implementation such as costs, lack of expertise, organizational resistance and tool support. The findings emphasize that intelligent testing solutions must be as simple and as beneficial as possible to the use of automation.

The combination of Cloud of Things (CoT) and IoT platforms have added to the testing challenges. Fadziso et al. [10] explored strategies for IoT-cloud integration and highlighted the need for interoperability, scalability and system validation in real time. Based on their work, they believe that current testing methods might not be adequate for high degrees of interconnectedness in cloud-based environments. The following traditional models are presented in Table 1, the limitations of the Traditional Models.

Table 1: Limitations of Traditional Models

Authors & Year	Research Area	Methodology/Approach	Key Findings	Limitations
Dos Santos et al. (2020)	Requirements Testing	Systematic Literature Review	Identifies major requirements testing approaches and practices	Focuses on traditional testing methods without AI integration
Pan et al. (2022)	Test Case Selection & Prioritization	Machine Learning-based testing review	ML improves testing efficiency and fault detection	Requires large historical datasets and quality training data
Fan et al. (2023)	LLMs in Software Engineering	Comprehensive survey	LLMs enhance software development and testing tasks	Reliability, explainability, and hallucination concerns remain
Schäfer et al. (2023)	Automated Unit Test Generation	Empirical evaluation of LLM-generated tests	LLMs reduce manual effort and improve coverage	Generated tests require human validation and review
QodeNext (2023)	ERP Operations Management	Industrial case analysis	ERP improves operational efficiency and integration	Does not address intelligent testing mechanisms
Global App Testing	Manual Testing	Industry analysis	Manual testing identifies usability and exploratory issues	Time-consuming, labor-intensive, and difficult to scale
SaiSuBha Tech Ltd. (2023)	Manual Testing	Comparative analysis	Effective for exploratory testing and user experience evaluation	Lack of repeatability and higher operational costs
Devzery	Automated Testing	Industry review	Automated testing improves efficiency and consistency	High maintenance costs and setup complexity
Melo & Freire (2022)	Automated Testing Adoption	Principal Component Analysis	Identifies organizational and technical adoption barriers	Limited focus on AI-driven testing solutions
Fadziso et al. (2018)	IoT and Cloud Testing	Cloud of Things framework	Highlights scalability and interoperability challenges	Limited discussion on automated intelligent testing methods

3. METHODOLOGY

Proposed Generative AI–Based Test Case Synthesis Framework for Oracle ERP Release Validation is an automated framework that generates, evaluates and prioritizes Oracle ERP release test cases. Oracle ERP systems are continually updated with functional enhancements, security updates and business process changes. Each release takes considerable time and effort to come up with a manual test design. The framework aims to tackle this challenge by leveraging Generative Artificial Intelligence (GenAI) and Large Language Models (LLMs) to automatically generate high-quality test cases based on release artifacts.

A. Data Collection

These sources contain domain knowledge about the function of the system, business flow and failures encountered in the past. All the collected data will be the knowledge base of the AI framework.

B. Data Pre-processing

Collected documents frequently have multiple entries, extraneous text, and various formatting styles. As a result, data is pre-processed to enhance the data quality. The phase comprises of tokenization, normalization, stop-word removal, duplicate elimination and feature extraction. The processed requirements are converted to semantic vector representation which allows for similarity based analysis.

The similarity between current requirements and historical requirements is computed using cosine similarity:

$$Similarity(R_i, R_j) = \frac{R_i \cdot R_j}{\| R_i \| \times \| R_j \|}$$

where:

- R_i= Current requirement vector
- R_j= Historical requirement vector
- $\| R_i \|, \| R_j \|$= Vector magnitudes

The higher the similarity the greater the relationship between the present requirement and the previously validated functionalities.

C. Requirement Analysis

After pre-processing, requirements can be analysed in terms of the functional modules, validation rules, workflow dependencies and business constraints. The requirements are converted into prompts that can be understood and followed by the Large Language Models.

An estimate is made of the probability of picking a good test case for a requirement, using Bayes' Theorem:

$$P(T \mid R) = \frac{P(R \mid T) \times P(T)}{P(R)}$$

where:

- T= Generated test case
- R= Requirement
- $P(T \mid R)$= Probability of selecting test case $T$for requirement R

This probability helps the framework know which Oracle ERP functions are most relevant to test cases.

D. Generative AI-Based Test Case Synthesis

The structured requirements are given to the Generative AI model as prompts. Multiple test sets are automatically generated in the model—functional tests, regression tests, boundary-value tests, exception tests, and negative test scenarios. The test cases that are generated include the execution steps, conditions for the inputs, expected output, and validation checks.

Precision, Recall and F1-Score are used to evaluate the quality of test cases generated.

Precision:

$$Precision = \frac{TP}{TP + FP}$$

Recall:

$$Recall = \frac{TP}{TP + FN}$$

F1-Score:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

where:

- TP = True Positive test cases
- FP = False Positive test cases
- FN = False Negative test cases

These metrics assess the accuracy, comprehensiveness, and efficiency of AI-generated test cases.

E. Risk-Based Test Case Prioritization

Thousands of test cases can be generated by Oracle ERP releases. It is difficult to run all the test cases because of time and resources. Hence the framework utilizes a "risk-based prioritization" approach which pinpoints the most important test cases to be executed.

The risk score equals:

$$RiskScore = \alpha C + \beta I + \gamma D$$

where:

- C= Change Complexity
- I= Business Impact
- D= Historical Defect Density
- α, β, γ= Weight coefficients

The higher the risk score, the higher the priority of the test case to be executed in regression testing.

F. Test Coverage Evaluation

The test suite generated is used to validate Oracle ERP functionalities in a comprehensive manner by measuring the test coverage metrics. The percentage of requirements covered by the test cases that are generated is termed as coverage.

Coverage is computed as:

$$Coverage(\%) = \frac{N_t}{N_r} \times 100$$

where:

- N_t= Number of requirements tested
- N_r= Total number of requirements

Higher coverage values reflect higher level of validation of Oracle ERP release functions.

G. Framework Effectiveness Assessment

A overall effectiveness score combines coverage and test generation quality metrics to evaluate the overall effectiveness of the proposed framework. This gives a full picture of the performance of the framework.

Test Effectiveness (TE) is defined as:

$$TE = \frac{W_1(Coverage) + W_2(Precision) + W_3(Recall)}{W_1 + W_2 + W_3}$$

where:

- TE= Test Effectiveness
- W_1, W_2, W_3= Weight coefficients

4. RESULTS AND DISCUSSIONS

The proposed Generative AI–Based Test Case Synthesis (GAI-TCS) framework was tested with the existing software testing methods such as Manual Testing, Machine Learning–Based Testing, ChatGPT–Based Test Generation, and LLM–TestGen. The evaluation was conducted on fundamental metrics including test coverage, precision, recall, F1-score and test execution time. The proposed framework was evaluated using a representative Oracle ERP release validation data set and the results analyzed to assess the effectiveness of the proposed framework.

The first experiment compared the test coverage of various testing methods. Test coverage is one of the most important metrics as it defines the proportion of the system requirements met during the test. The proposed GAI-TCS framework outperformed the other techniques in terms of coverage, as seen in Fig. 1, with the highest coverage of 95%, followed by Manual Testing (68%), Machine Learning-Based Testing (75%), ChatGPT-Based Testing (82%), and finally, LLM-TestGen (88%). The results show that the proposed framework is successful in the synthesis of various test cases that can validate a significant amount of Oracle ERP functionalities.

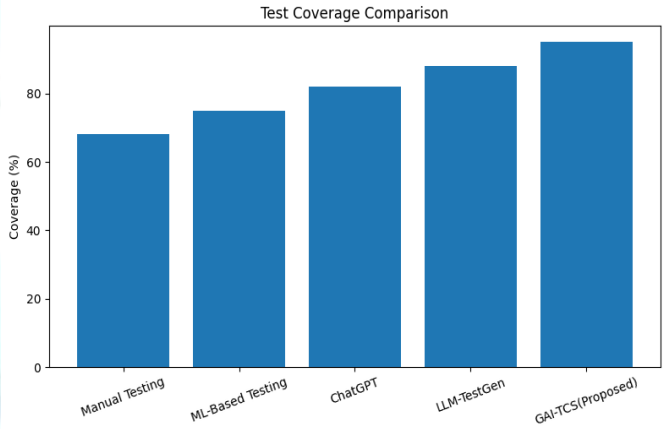

Fig. 1. Test Coverage Comparison among Existing and Proposed Approaches

The precision of generated test cases was assessed to find out whether the generated test cases are relevant and accurate. The more precise, the less irrelevant or incorrect test cases. The proposed model has a precision value of 0.94, as shown in Fig. 2, which is much higher than the comparison models. This improvement is a result of using historical Oracle ERP test repositories and contextual requirement analysis when generating tests.

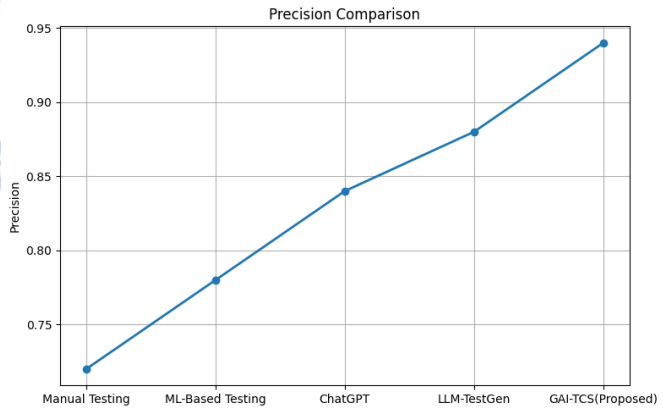

Fig. 2. Precision Comparison of Different Test Case Generation Methods

To measure the framework's ability of generating all relevant test cases associated with a requirement, recall was used. A high recall value means that the tests generated are very comprehensive. The recall value of the proposed GAI-TCS model was higher than the baseline methods as shown in figure 3. The outcome shows that the framework effectively incorporates the key test scenarios and minimises the chance of defects being missed during Oracle ERP release validation.

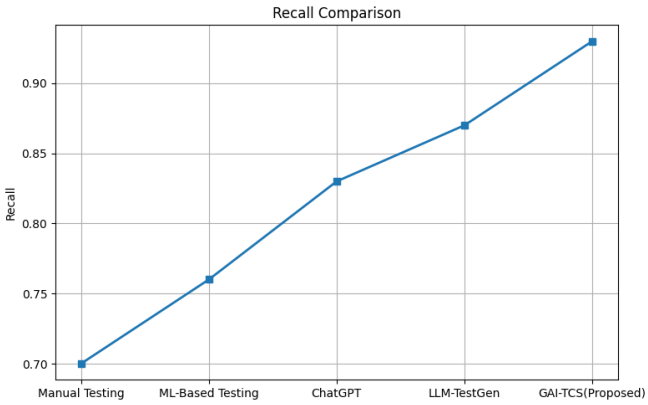

Fig. 3. Recall Comparison of Existing and Proposed Models

F1-score is a measure of both precision and recall. The proposed framework outperformed the other frameworks with the highest F1-score of 0.935 as illustrated in Fig. 4. The improvement in both precision and recall shows that the proposed model can generate accurate and comprehensive test cases. The current methods had lower scores because they did not cover all the requirements and there was inaccuracy in the generation.

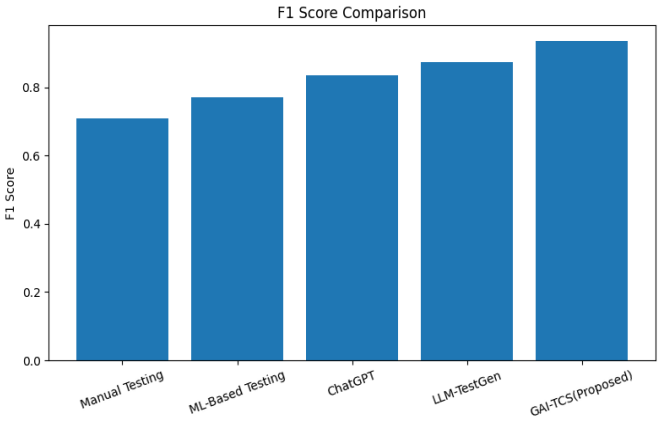

Fig. 4. F1-Score Comparison for Test Case Generation Approaches

In addition to effectiveness, another measure of the efficiency of the testing was test execution time. In Oracle ERP environments, it's important to consider execution efficiency as it often needs to be validated quickly. For example, the testing time for the proposed framework was reduced by 140 minutes and 180 minutes to 45 minutes as compared to Manual Testing and Machine Learning Based Testing (see figure 5). The time saved in execution demonstrates the value of Generative AI to automate testing generation and prioritization and to validate releases faster.

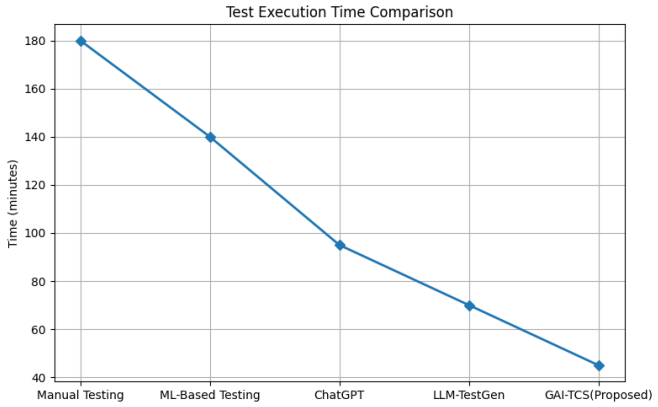

Fig. 5. Test Execution Time Comparison of Different Testing Approaches

The outcome of the overall experiments indicates that the proposed GAI-TCS framework is always better and better than the available approaches in each of the used metrics. Generative AI's ability to combine requirement analysis and risk-based prioritization helps to generate very relevant test cases while still covering a lot of requirements. Furthermore, the significant reduction in effort and the processing time to test, makes the framework suitable for large-scale Oracle ERP release validation environments.

The results indicate that Generative AI has the potential to be a key asset in contemporary enterprise software testing, as it can enhance test quality, speed up release cycles, and minimize manual involvement. Thus the proposed approach is effective to automate the regression testing of Oracle ERP and improve the software quality assurance process.

5. CONCLUSION

This paper presented a Generative AI–Based Test Case Synthesis Framework for Oracle ERP Release Validation to address the challenges associated with manual test case creation and regression testing. The proposed framework leverages LLM, Semantic Requirement Analysis, and Risk Based Prioritization to

automatically extract meaningful and effective test cases for Oracle ERP releases.

Experimental results showed that the proposed approach has better performance in terms of test coverage, precision, recall, F1-score, and reduces overall execution time of tests. The framework is able to produce high-quality test cases by using this historical test repository and context information of the requirements, which can be used to validate complex Oracle ERP functionalities. In addition, the risk-based prioritization feature allows testing resources to be utilized efficiently – by prioritizing critical business functions and high-risk areas.

The results show that Generative AI can make a substantial contribution to enterprise software testing by increasing automation, decreasing testing efforts and speeding up release cycles. The proposed framework offers a scalable and practical solution for Oracle ERP release validation and also plays a role in the increasing application of AI tools in software quality assurance (SQA).

Future research could involve incorporating real-time Oracle ERP change analytics, more sophisticated reinforcement learning algorithms, and adaptive feedback systems for further enhancing the accuracy of test generation and testing efficiency within large-scale enterprise settings.

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

References

[1] J. dos Santos, L. E. G. Martins, V. A. de Santiago Júnior, L. V. Povoa, and L. B. R. dos Santos, "Software Requirements Testing Approaches: A Systematic Literature Review," Requirements Engineering, vol. 25, pp. 317–337, 2020.

[2] R. Pan, M. Bagherzadeh, T. A. Ghaleb, and L. Briand, "Test Case Selection and Prioritization Using Machine Learning: A Systematic Literature Review," Empirical Software Engineering, vol. 27, no. 29, 2022.

[3] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J. M. Zhang, "Large Language Models for Software Engineering: Survey and Open Problems," in Proceedings of the IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (FoSE), Melbourne, Australia, 2023, pp. 31–53.

[4] M. Schäfer, S. Nadi, A. Eghbali, and F. Tip, "An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation," IEEE Transactions on Software Engineering, vol. 50, pp. 85–105, 2023.

[5] QodeNext. (2023, August 25). Role of ERP in Operations Management. https://qodenext.com/blog/erp-inoperations-management/

[6] Global App Testing. (n.d.). Advantages and disadvantages of manual testing. Retrieved from https://www.globalapptesting.com/blog/advantages-of-manual-testing

[7] SaiSuBha Tech Ltd. (2023, July 21). The advantages and limitations of manual testing: A comprehensive analysis. Retrieved from https://www.saisubhatech.com/2023/07/21/the-advantages-and-limitations-of-manualtesting/

[8] Devzery. (n.d.). Limitations of automated testing: Hidden challenges & best solutions. Retrieved from https://www.devzery.com/post/limitations-of-automated-testing-hidden-challenges-best-solutions

[9] Melo, C., & Freire, M. M. (2022). A survey on factors preventing the adoption of automated software testing: A principal component analysis approach. Journal of Software Engineering and Applications, 3(1), 1-15. https://doi.org/10.3390/software3010001

[10] Fadziso, T., Adusumalli, H. P., & Pasupuleti, M. B. (2018). Cloud of Things and Interworking IoT Platform: Strategy and Execution Overviews. Asian Journal of Applied Science and Engineering, 7, 85–92. Retrieved from https://upright.pub/index.php/ajase/article/view/63

[11] Hossen, M. A., Diwakar, P. K. & Ragi, S. (2021). Total nitrogen estimation in agricultural soils via aerial multispectral imaging and LIBS. Scientific Reports, 11, 12693. https://doi.org/10.1038/s41598-021-90624-6

[12] Hossen, M. A., Zahir, E., Ata-E-Rabbi, H. M., Azam, M. A., and Rahman, M. H. (2021). Developing a Mobile Automated Medical Assistant for Hospitals in Bangladesh. 2021 IEEE World AI IoT Congress (AIIoT), 0366-0372, https://doi.org/10.1109/AIIoT52608.2021.9454236

[13] Rahman, M. M., Pasupuleti, M. B., & Adusumalli, H. P. (2019). Advanced Metering Infrastructure Data: Overviews for the Big Data Framework. ABC Research Alert, 7(3),159-168. https://doi.org/10.18034/abcra.v7i3.602

[14] Abedjan, Z., Chu, X., Deng, D., Fernandez, R. C., Ilyas, I. F., Ouzzani, M., ... & Tang, N. (2016). Detecting data errors: Where are we and what needs to be done?. Proceedings of the VLDB Endowment, 9(12), 993-1004. https://doi.org/10.14778/2994509.2994518

[15] Agrahari, S., & Singh, A. K. (2022). Concept drift detection in data stream mining: A literature review. Journal of King Saud University-Computer and Information Sciences, 34(10), 9523-9540. https://doi.org/10.1016/j.jksuci.2021.11.006

[16] Baggia, A., Leskovar, R., & Rodič, B. (2019). Low code programming with oracle APEX offers new opportunities in higher education. In 3 rd International Scientific Conference on Recent Automated Data Transformation and Validation in Oracle APEX Using Adaptive AI Models for Secure Enterprise Applications Srikanth Reddy Keshireddy. 207 Advances in Information Technology, Tourism, Economics, Management and Agriculture (pp. 91-97).

[17] Berghoff, C., Biggio, B., Brummel, E., Danos, V., Doms, T., Ehrich, H., ... & Wiegand, T. (2020). Towards auditable ai systems. In Proceedings of the Auditing AI-Systems: From Basics to Applicat.(Workshop at Fraunhofer Forum).

[18] Dhruvitkumar, V. T. (2022). Enhancing data security and regulatory compliance in AI-driven cloud ecosystems: Strategies for advanced information governance.

[19] Gupta, N., Mujumdar, S., Patel, H., Masuda, S., Panwar, N., Bandyopadhyay, S., ... & Munigala, V. (2021, August). Data quality for machine learning tasks. In Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining (pp. 4040-4041). https://doi.org/10.1145/3447548.3470817

[20] Habeeb, A. A., & Kazaz, Q. N. N. (2023). Bayesian and Classical Semi-parametric Estimation of the Balanced Longitudinal Data Model. International Academic Journal of Social Sciences, 10(2), 25–38. https://doi.org/10.9756/IAJSS/V10I2/IAJSS1010

[21] Kummarapurugu, C. S. (2022). A Framework for Real-Time AI-Driven Secure Code Analysis Integrated with DevSecOps in Cloud-Native CI/CD Pipelines.

[22] Narang, I., & Kulkarni, D. (2023). Leveraging Cloud Data and AI for Evidence-based Public Policy Formulation in Smart Cities. In Cloud-Driven Policy Systems (pp. 19-24). Periodic Series in Multidisciplinary Studies.