© JUN 2023 | IRE Journals | Volume 6 Issue 12 | ISSN: 2456-8880

Automating Software Development Lifecycle with Machine Learning: Enhancing Efficiency and Quality Assurance

MANOJ BHOYAR

Abstract- ***In the current world, ML deployment in the software development lifecycle (SDLC) has been scientifically adopted to boost the SVC. This paper investigates the approaches and technologies used to automate tasks within various SDLC phases, such as requirement specification, code development, testing, and installation. Through regular predictive analytics and intelligent automation, organizations can cut down time and cost and attempt to find workarounds to many challenges. In this paper, we present the role of ML algorithms in software testing, including aspects of automated bug detection and test case generation, and mention several examples of successful applications. Moreover, issues with integrating ML into current and future processes, such as data validity, model explainability, and the teams, are discussed. In conclusion, this research points out the ability of machine learning to enhance development procedures and improve the value of software solutions, thus enabling the consequent evolution of software environments.***

Index Terms- ***Software Development Lifecycle (SDLC), Machine Learning (ML), Automation, Efficiency, Quality Assurance.***

I. INTRODUCTION

Background on the software development lifecycle (SDLC)

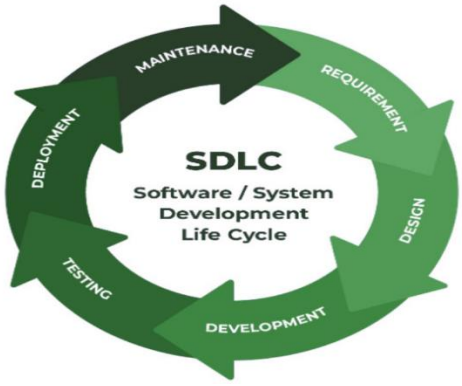

Figure 1: Pictorial Representation of Software Development Lifecycle

The specificity of the processes of creating high-quality software is described clearly by the Software Development Life Cycle (SDLC) methodology. In detail, the SDLC methodology focuses on the following phases of software development:

1. Requirement analysis
2. Planning
3. Software design, including architectural design
4. Software development
5. Testing
6. Deployment

In this article, you will learn how the SDLC model functions and further describe each phase, along with examples, to better understand all of the phases.

What is the software development life cycle?
SDLC stands for Software Development Life Cycle, and as its name suggests, the process delivers the best quality software at the least cost in the shortest time. SDLC lays down a well-defined cycle of phases that, in one way or another, enables an organization to deliver good, tested software that is ready for production.

The context in which SDLC is practiced encompasses six phases, as explained in the introduction. Several SDLC models exist, such as the waterfall, spiral, and agile.

Well, exactly how does the Software Development Life Cycle function?
How the SDLC Works
SDLC makes sense by reducing the cost of production while increasing quality and shortening development time. SDLC completes these distinctive objectives with a plan that defines a flexible and efficient approach to software development free from the most common vices of software projects. Implemented in that plan is the assessment of the weaknesses of existing systems.

This is followed by the new system specifications that any new system must meet. It then synthesizes it in the analysis, planning, design, development, testing, and deployment process that generates the software. The template activities help SLDC prevent failing to ask the end-user or client for feedback, which will result in unnecessary corrections through rework and after-the-fact solutions.

It is also good to understand that forward thinking is put into the testing phase. Since the SDLC is a cyclic process, one must check the code's quality in every cycle, which makes it very important. Some organizations dedicate little effort to testing even though a better effort will save the organization a large amount of redeployment, time, and money—smart pen and writing are the kinds of tests.

Describing the Software Development Life Cycle processes in more detail is high time.

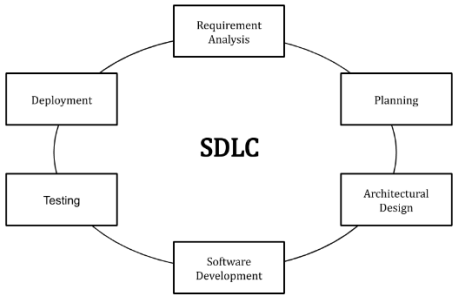

Figure 2: Different Stages of the Software Development Life Cycle

Stages and Best Practices
Stages and Best Practices
It is important to note that working according to SDLC's best practices and stages allows the process to work smoothly, effectively, and profitably.
1. Identify the Current Problems
"What is the current strategic issue?" This level of the SDLC offers input from all key stakeholder groups, including industry specialists and programmers, and introduces the Current System Characteristics and Establishment Improvement as the Objective.
2. Plan
"What do we want?" During this stage of the SDLC, the team decides the costs and resources needed to implement the analyzed requirements. It also outlines the associated risks and gives risk mitigation contingencies and sub-plans.
In other words, the team should identify whether this project can be done and how it can be done with the least risk possible on the team's part.
3. Design
What do players want, and how are they going to get it? The Software Development Life Cycle is divided into several phases; this phase follows the creation of the software specifications, followed by the Design Specification. This plan is shared with all the stakeholders, who suggest the plan outline. It is, therefore, important to have a plan for how the stakeholders will feed into this document or how such inputs will be there. Any failure at this stage will, in the best of it, propound cost implications or, at the worst, cause the total failure of the project.

Importance of efficiency and quality assurance in software development

Importance of efficiency and quality assurance in software development

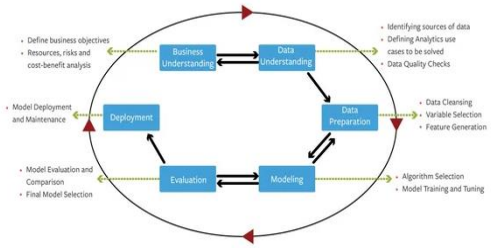

Figure 3: Quality Assurance in Software Development

Most business people are inclined to pay more attention to the quality assurance of their firms' software development and creation. However, this is an important policy that needs to be embraced and practiced by any business, regardless of size.

In its absence, it could be disastrous for software systems. The software will not run and will have breakdown problems, flaws, and kinks. All of this can, therefore, have an effect on several functions in the business:

Organizations must focus on quality assurance services to accommodate the customer's needs and expectations. We have developed this detailed guideline to discuss what quality assurance is in software development, why it is crucial, and how it may benefit companies.

What is Quality Assurance in Software Development?

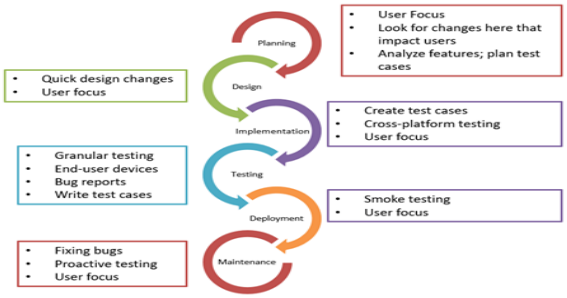

Fig 4: Quality Assurance for Software Development: Overview & Purpose

But first, we must answer what it means because many business owners might need to learn this term. Quality assurance is one of the critical activities in software development. It points to ensuring that the software being developed is up to par with the customer or company's expectations, is accredited, and is of the highest quality.

Therefore, several steps must be undertaken to meet these needs and requirements. Apart from that, quality assurance ensures that quality is assessed at each developmental phase and problems are identified beforehand. It also saves time because the details provided are more comprehensive, thus raising its quality further.

Finally, when these needs are met, well-developed software will offer numerous advantages to the company, as pointed out in the paper. For instance, it will enhance flow and make communications significantly faster.

Quality assurance is not just a simple linear step in the software development process; it performs a continuously changing function throughout the process, hunting for faults that could mushroom or become noticed by customers. However, it is correct to point to the main focus of quality assurance: having the best possible software or product.

Having defined what software quality assurance is and, therefore, distinguished it from other processes, such as software testing and quality control, it is high time to turn to the main advantages of this specific process.

Quality assurance activities are conducted at every stage of software development, and there are very many. The process is very crucial in software systems since it ushers in the best quality of the final product or system.

There are many advantages of implementing software quality assurance in an organization, and it not only covers software quality. Here are some of the key benefits of SQA:

This means that the efficiency of integrated business processes will improve customer satisfaction.

The need for getting it right must be addressed for any business with increasing reliance on software as it must be addressed. This is why you must ensure that what you offer your customers works properly, has no defects, and is as good as the audience expects. Quality assurance assists you in doing that.

High-quality software is efficient, accurate, and free most of the time from errors or failure compared to low-quality software. Its usage to increase the confidence of customers helps the organization get loyal citizens. As a result, such companies shall have a good standing with customers.

When customers are satisfied with the quality of software, they are better placed to pass that message to their counterparts, thus garnering more business and, therefore, increased shore and more revenue. Customers usually prefer products that are of high quality to those companies that offer the bottom products.

Reduced Costs

Many people may view quality assurance, particularly in software development, to be costly, but this is not the truth. Also, it can help to decrease costs in the following ways: First, it is easy to correct the defects by solving them from the root before they develop to a stage that will cost more to solve. Defects detected later in the development cycle are likely to take longer and are expected to be expensive.

Second, guaranteeing quality reduces the chances of the software developing faults or costly repairs or maintenance. This can lead to a fair amount of cost savings over the long run. It can also make the product very lucrative. Because it is of exceptional quality, more people would have a desire for it, and hence its sales would improve.

If you launch your product without software quality assurance, you may be in for many problems, such as bugs and malfunctioning. Such can be very tough and time-consuming when they are being repaired. It would also cause a decrease in sales. The profit you could have accrued with a successful product would have also been a loss.

Increased Efficiency

SQA also has the potential to improve efficiency because various defects may be found, resulting in changes being made early in the development. This often means that if a defect is discovered after the development cycle is underway or completed, time and money must be invested to correct the problem.

When defects are discovered early, the developer can correct the error and be able to perform other development activities. Therefore, development time is reduced. Consequently, it can help them concentrate on their expertise rather than worry about different aspects.

Even during such a period, you will not be lowering the quality of the products since quality assurance will have been done. You will thus enhance the efficiency of your organization and ensure that you give out quality products or software.

Table 1 shows the Importance of efficiency and quality assurance in software development.

Benefit	Explanation	Outcome
Improved Customer Satisfaction	High-quality software is reliable, user-friendly, and has fewer errors or failures. Smooth performance meets customer expectations.	High-quality software is reliable, user-friendly, and has fewer errors or failures. Ensuring smooth performance meets customer expectations.
Reduced Costs	Early identification of defects minimizes expensive fixes later in development. Quality software requires less	This results in higher profitability through lower repair costs, increased sales, and fewer malfunctions or bugs affecting

	repair or maintenance, reducing long-term costs.	product performance.
Increased Efficiency	Detecting defects early allows developers to address them quickly, preventing delays and reducing additional resource needs.	Improves organizational efficiency by enabling developers to focus on core tasks, ensuring timely delivery without compromising product quality.

Introduction to machine learning in the context of SDLC

Applying ML in the SDLC environment and other workflows contributes to the emergence of new challenges that go beyond simple software creation. Again, whereas the technically-based software development life cycle normally comprises the following phases: planning, design, development, testing, deployment, and maintenance, the ML projects emphasize data processes. The emphasis now goes toward data gathering, processing, and pre-processing, as data quality is critical for teaching effective models. At the model selection stage, algorithms are trained and tested to choose the best one. As you incorporate it into several software systems, they must be supervised and retrained periodically for better results with new data. Due to this iterative approach, model maintenance is an important aspect that embraces updating models and bringing them in compliance with current business objectives.

II. LITERATURE REVIEW

This paper synthesizes existing literature about different phases of SDLC and ML applications in software engineering. Due to the adoption of ML in the SDLC, several prior researchers have performed empirical studies in software engineering for data science. The first work fundamentally compared the state of affairs of developing ML or non-ML systems in many aspects of software engineering. In that study, interviews were used to explore what might be uncomfortable from a more general tooling perspective to surface the issues and concerns regarding using visual analytic tools in SLDM, where none of the cycle phases has tooling to support it. This study also aimed to describe the professional roles and practices related to DA. Testing and verifying the ML software systems is always a difficult task. The ML model could be wrong even if the learning algorithm is done correctly because training data could have been better. Combinatorial testing by traditional methods is needed for such systems. This engine software must study and create sophisticated means of reporting these issues. Software engineers have been concerned with validating those systems developed using ML and AI models or testing such systems. Literature reviews have been done previously on using ML algorithms in each developmental phase of the software development life cycle. Our study found a publication that gives a relative view of mapping the existing ML techniques between the tools and tools' scope in the stages of SDLC. However, their review was done for the period between 1991 and 2021. However, our paper employs a systematic review to investigate the influence of ML in the most recent papers (2015–2021) across all the phases of SDLC rather than just one phase, as has been the focus of most works. Our work also reveals that four of the most widely adopted algorithms in overall SDLC are linear regression. To see the global trend for each of those four techniques, we evaluated them individually in every life cycle phase.

III. METHODOLOGY

Firstly, our planned research aimed to consider only articles from the most recent ten years (from 2010 to 2020). Because of the sheer volume of articles produced in recent years, we only limited our review to journal articles and conference papers published in the last five years. We found that it grew much faster in the last decade, starting from 2010 to 2015. We then steadily increased 2015 the number of publications because recent advancements in ML helped SDLS a lot, as ML techniques can reshape and update a software system. We used several keywords and queries to get our results from ACM, IEEE, and Springer databases. Yes, some of the queries worked well in IEEE and Springer, but changing the keywords

and queries was necessary to find the publications in ACM.

Furthermore, we observed that ACM would retrieve many unrelated works when directly querying the two other libraries by their names. As a result, it was necessary to use additional limitations to access the Database. For example, when developing the initial query, it is essential to add 'Artificial Intelligence' into the keywords. In some cases, our queries provided us with similar or better work in multiple publications; we handled each as a distinct publication because they centered on different aspects. Furthermore, when one ML technique was used in a few phases of the model, we distinguished these while counting them. The latter scenario compelled us to investigate four popular ML methods in each phase while providing a more targeted viewpoint.

The current application of machine learning for SLDC is as follows;
As we can see, ML is implemented throughout the entire SDLC process. In this section, we discuss the impact of applying ML in each phase of the SDLC:
This paper is devoted particularly to the problem of Software Requirements Analysis.
Software Architecture Design
Software Implementation
Software Testing
Software Maintenance

Software Requirement Specification
This phase concerns the collection and documentation of the requirement facets of that software system with a focus on functional and nonfunctional aspects. It enables one to ensure that stakeholders' needs are established and communicated as prerequisites of the development team.

Software Architecture Design
In this stage, what needs to be done first and the framework of the overall system is decided, i.e., what technologies, frameworks, and design patterns are to be incorporated. This means there is always a need to design a system that can be easily scaled; additional functionality means that as the requirement of the project demands in the future, so shall the system.

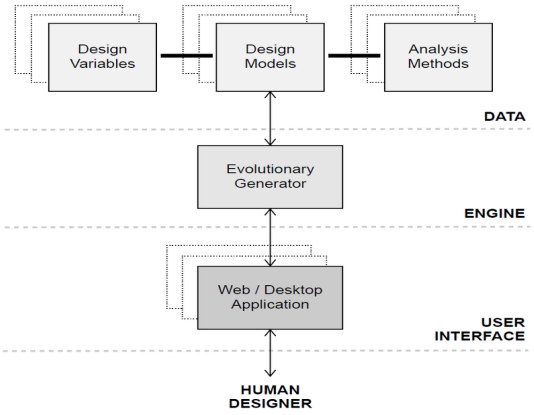

Figure 5: Software Architectural Design

Software Implementation
The term implementation of a new design means the action taken to write the design in a programming language and tools. By following the above design parameters, developers design, integrate, and ensure that the related software components work the desired way.

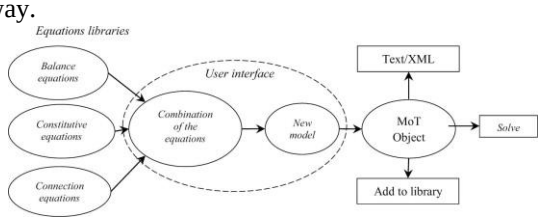

Figure 6: Software Implementation Design

Software Testing
It is important to see that the developed software is operational and meets the required needs by eradicating them. These are unit, integration, system, and acceptance tests performed to ascertain that they are functional, perform the expected tasks, and are secure.

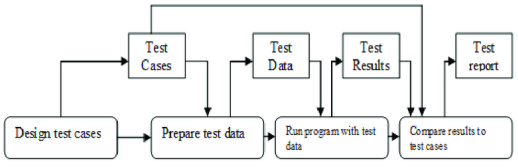

Figure 7: Design of Software Testing

Software Maintenance
software maintenance is well associated with modifying the deployed software; there has to be a way to see it as having to be changed to fit new needs, solve some problems, or even see it as needing

improvement in certain aspects. It applies that creates surety of the software in operations and serves the purpose in the lifecycle.

IV. RESULT AND DISCUSSIONS

The integration (ML) for the Software Development Lifecycle (SDLC) has effectively provided enhanced productivity and quality assurance. This means that with the help of ML techniques, it is possible to automate SDLC cycles, such as requirements analysis, system architecture, programming, verification, installation, and updates. Consequently, the function of repeating work and excluding individuals minimizes human mistakes and, thus, enriches the caliber of software systems.

Machine learning-based tools collect and compare large data sets in requirement analysis to predict project requirements better. In the design phase, ML models help to provide approaches to create efficient system designs and to recognize design issues throughout the process. Code generation has also advanced in the coding and development of applications to minimize development time through intelligent programming tools and automatic recommenders that enhance code writing and auto-correct syntax.

Testing, which is otherwise a very manual process, is a prime area that positively impacts the use of ML for automation. Any model trained for pattern detection in bugs or errors can automate test cases and defect predictions, leading to better test coverage. Automated testing also helps reduce re-testing time since faulty requirements can be immediately spotted and fixed in any phase of the testing process, which optimizes the testing cycle. In addition, such ML tools in the CI/CD pipeline, which integrate code and deploy systems, also manage and monitor the interface's real-time performance concerns.

However, the offered approach still reveals that the main issue is integrating ML into SDLC processes. Using the ML models in software projects is an advantage and a disadvantage since the models necessitate frequent training using newly developed or updated datasets, which brings extra overhead to software projects. Moreover, some of the algorithms used in ML approaches are black-boxed, which means that debugging and mesmerizing the algorithms could be problematic during their creation and testing. To handle these problems, organizations must implement strategies for retraining models, increasing transparency, and applying process automation using AI and human-in-the-loop where needed.

The use of ML in automating SDLC holds a lot of advantages since it eases the process and minimizes the occurrence of errors while at the same time enhancing the quality of the results. However, successful implementation of ML-based automation needs rigorous planning and, simultaneously, understanding that automation can be a tool rather than a solution in every phase of the SDLC and needs to be integrated with human intelligence.

CONCLUSION

The incorporation of machine learning into SDLC is, therefore, a revolutionary way of improving S/W production processes and improving the quality of the products. In the different phases, including requirement analysis, design, coding, testing, and deployment, the ML technique cuts the amount of work done manually, speeds up the procedures, and decreases the chance of human mistakes. It also helps in better identification and tracking of defects, predictive action, and even making recommendations, providing enhanced project results recommendations, providing enhanced project results. The implementation of ML in SDLC comes with its challenges, including the issue of model updating, interpretability, and the role of humans. Hence, organizations need to implement sound long-term model management frameworks, reporting frameworks, and proper people strategies that require cooperation between automated model maintenance and software engineering for AI to fully unlock its automation capability potential. With time and the advancement in various ML technologies, the need for software development, enhancement of innovations, and the ability to develop quality software within the shortest time possible should be emphasized.

REFERENCES

[1] Altvater, A. (2024, August 27). What Is SDLC? Understand the Software Development Life Cycle. Stackify. https://stackify.com/what-is-sdlc/

[2] Navaei, M., & Tabrizi, N. (2022). Machine Learning in Software Development Life Cycle: A Comprehensive Review. *https://www.scitepress.org/PublishedPapers/2022/110406/110406.pdf*, 344–354. https://doi.org/10.5220/0011040600003176

[3] Admin-Coolsnail. (2023, August 16). The Importance of Quality Assurance in Software Development. CoolSnail Technologies. https://coolsnail.com/the-importance-of-quality-assurance-in-software-development/

[4] Singh, A. (2023, June 27). Leveraging Generative AI in SDLC: Unleashing unprecedented efficiency and innovation in software development. *Medium*.

[5] https://medium.com/@anand94523/leveraging-generative-ai-in-sdlc-unleashing-unprecedented-efficiency-and-innovation-in-software-3613f84a6877

[6] https://ideamaker.agency/automation-in-software-development/. (n.d.). https://ideamaker.agency/automation-in-software-development/.

[7] quality-assurance-for-software-development-overview-purpose. (n.d.). https://study.com/academy/lesson/quality-assurance-for-software-development-overview-purpose.html.

[8] *Software architecture diagram, showing data, engine, and user. . .* (n.d.). ResearchGate.https://www.researchgate.net/figure/Software-architecture-diagram-showing-data-engine-and-user-interface-layers-The-types_fig11_273641993

[9] software-implementation. (n.d.). https://www.sciencedirect.com/topics/computer-science/software-implementation.

[10] A-model-of-the-software-testing-process. (n.d.). https://www.researchgate.net/figure/A-model-of-the-software-testing-process_fig1_332247218.

[11] Krishna, K. (2020). Towards Autonomous AI: Unifying Reinforcement Learning, Generative Models, and Explainable AI for Next-Generation Systems. Journal of Emerging Technologies and Innovative Research, 7(4), 60-61.

[12] Murthy, P. (2020). Optimizing cloud resource allocation using advanced AI techniques: A comparative study of reinforcement learning and genetic algorithms in multi-cloud environments. World Journal of Advanced Research and Reviews. https://doi. org/10.30574/wjarr, 2.

[13] MURTHY, P., & BOBBA, S. (2021). AI-Powered Predictive Scaling in Cloud Computing: Enhancing Efficiency through Real-Time Workload Forecasting.

[14] Mehra, A. D. (2020). UNIFYING ADVERSARIAL ROBUSTNESS AND INTERPRETABILITY IN DEEP NEURAL NETWORKS: A COMPREHENSIVE FRAMEWORK FOR EXPLAINABLE AND SECURE MACHINE LEARNING MODELS. International Research Journal of Modernization in Engineering Technology and Science, 2.

[15] Mehra, A. (2021). Uncertainty quantification in deep neural networks: Techniques and applications in autonomous decision-making systems. World Journal of Advanced Research and Reviews, 11(3), 482-490.

[16] Thakur, D. (2020). Optimizing Query Performance in Distributed Databases Using Machine Learning Techniques: A Comprehensive Analysis and Implementation. Iconic Research And Engineering Journals, 3, 12.

[17] Krishna, K. (2022). Optimizing query performance in distributed NoSQL databases through adaptive indexing and data partitioning techniques. International Journal of Creative Research Thoughts (IJCRT). https://ijcrt. org/viewfulltext. php.

[18] Krishna, K., & Thakur, D. (2021). Automated Machine Learning (AutoML) for Real-Time Data Streams: Challenges and Innovations in Online Learning Algorithms. Journal of Emerging Technologies and Innovative Research (JETIR), 8(12).

[19] Murthy, P., & Mehra, A. (2021). Exploring Neuromorphic Computing for Ultra-Low Latency Transaction Processing in Edge Database Architectures. Journal of Emerging Technologies and Innovative Research, 8(1), 25-26.

[20] Thakur, D. (2021). Federated Learning and Privacy-Preserving AI: Challenges and Solutions in Distributed Machine Learning. International Journal of All Research Education and Scientific Methods (IJARESM), 9(6), 3763-3764.

[21] KRISHNA, K., MEHRA, A., SARKER, M., & MISHRA, L. (2023). Cloud-Based Reinforcement Learning for Autonomous Systems: Implementing Generative AI for Real-time Decision Making and Adaptation.

[22] THAKUR, D., MEHRA, A., CHOUDHARY, R., & SARKER, M. (2023). Generative AI in Software Engineering: Revolutionizing Test Case Generation and Validation Techniques.

[23] Krishna, K., & Murthy, P. (2022). AIENHANCED EDGE COMPUTING: BRIDGING THE GAP BETWEEN CLOUD AND EDGE WITH DISTRIBUTED INTELLIGENCE. TIJER-INTERNATIONAL RESEARCH JOURNAL, 9 (2).

[24] Murthy, P., & Thakur, D. (2022). Cross-Layer Optimization Techniques for Enhancing Consistency and Performance in Distributed NoSQL Database. International Journal of Enhanced Research in Management & Computer Applications, 35.

[25] MURTHY, P., MEHRA, A., & MISHRA, L. (2023). Resource Allocation for Generative AI Workloads: Advanced Cloud Resource Management Strategies for Optimized Model Performance.

[26] Alahari, J., Thakur, D., Goel, P., Chintha, V. R., & Kolli, R. K. (2022). Enhancing iOS Application Performance through Swift UI: Transitioning from Objective-C to Swift. In International Journal for Research Publication & Seminar, 13 (5): 312. https://doi. org/10.36676/jrps. v13. i5. 15 (Vol. 4).

[27] Salunkhe, V., Thakur, D., Krishna, K., Goel, O., & Jain, A. (2023). Optimizing Cloud-Based Clinical Platforms: Best Practices for HIPAA and HITRUST Compliance. Innovative Research Thoughts, 9 (5): 247. https://doi. org/10.36676/irt. v9. i5, 1486.

[28] Agrawal, S., Thakur, D., Krishna, K., & Singh, S. P. Enhancing Supply Chain Resilience through Digital Transformation.