Project plan for degree project

Version 1.3 – October 12, 2016

PA2534: MASTER THESIS IN SOFTWARE ENGINEERING

March 15, 2025

Thesis	Tentative title	Analyzing the Impact of Generative AI on Low-Code/No-Code Development: A Mixed-Methods Research Approach.
	Classification	Gen AI, Low Code/No Code Development, Bias, Security
Student 1	Name	BURRI VAMSHI KRISHNA
	e-Mail	vabr24@student.bth.se
	Social security nr	20030315-T195
Student 2	Name	BARURI ASHRITH SHARMA
	e-Mail	asba24@student.bth.se
	Social security nr	20040122-T98
Student 3	Name	GANDLA VARSHA PATEL
	e-Mail	vaga24@student.bth.se
	Social security nr	20031213-T204
Supervisor	Name and title	DR.AHMAD NAUMAN GHAZI
	e-Mail	nauman.ghazi@bth.se
	Department	Software Engineering

**2012 ACM Computing Classification System: www.acm.org/about/class/2012.*
Co-advisor from industry or a higher education institution (HEI).

1 Introduction:-

Creating software is becoming simpler and more accessible than ever before. People who previously required extensive programming skills can now build software solutions efficiently, thanks to the growing popularity of Low-Code/No-Code (LCNC) platforms. Platforms like Microsoft Power Apps, Salesforce Lightning, Bubble.io, and Airtable provide graphical interfaces, drag-and-drop capabilities, and pre-built templates, enabling users from diverse backgrounds—including business analysts, marketers, educators, and entrepreneurs—to rapidly create functional software applications [7]. These platforms are extensively utilized across industries to accelerate digital transformation, reduce development costs, and empower non-technical users to actively participate in software development[8].

The rapid expansion of software development has led to the development of Low-Code/No-Code (LCNC) platforms that allow users to create applications with little or no coding ability [3]. LCNC platforms provide graphic interfaces, drag-and-drop capabilities, and pre-built templates to make software development faster and easier [7]. LCNC platforms are being utilized on a large scale in industries to accelerate digital transformation, reduce development costs, and allow non-technical users to create functional applications [8]. The addition of Generative

AI (GenAI) in LCNC platforms has also revolutionized this field [4]. AI models such as OpenAI Codex and GPT-4 are now capable of generating code, optimizing workflows, improving UI/UX, and assisting with debugging [5]. This combination significantly reduces software development time while enhancing functionality and usability [10].

Despite these developments, several problems remain unaddressed. Security risks, AI bias, long-term maintainability, and scalability issues are significant concerns when applying AI-generated code [9]. The explainability and reliability of AI-supported development also need further investigation to guarantee software quality and compliance with industry standards [6].

This study aims to analyze the role of Generative AI in LCNC development, focusing on its benefits, drawbacks, and challenges [2]. Security risks, integration problems, and ethical concerns will be analyzed, providing best practices for secure and effective AI-enhanced software development [1]. By overcoming these challenges, this study will contribute to the responsible adoption of AI in LCNC platforms and provide recommendations for developers, organizations, and researchers in the field [3].

This research integrates qualitative and quantitative research in a special manner to show a broader perspective of AI-driven LCNC platforms. Unlike other studies, which are focused on either efficiency or security, this research also evaluates ethical concerns and scalability, thus giving a broader picture. This research also introduces an AI-enhanced security model, a subject not thoroughly discussed in current literature.

1.1 Aim:-

The aim of this study is to analyze the role of Generative AI in Low-Code/No-Code applications in terms of its benefits, drawbacks, security risks, and ethical issues. The study aims to explore the role of AI-based automation towards software development and how it addresses significant issues such as scalability, maintainability, and AI bias.

1.2 Objectives:-

1-To analyze how Generative AI helps to enhance Low-Code/No-Code (LCNC) development via better automation, software efficiency, and ease of use for users.
2- Defining the key issues and risks associated with AI-based LCNC platforms, including security risk, AI bias, scalability, and long-lived software maintainability.
3- Suggest guidelines and best practices for the secure, ethical, and scalable adoption of Generative AI in the development of LCNC software.

1.3 Motivation:-

Low-Code/No-Code (LCNC) platforms are making app development simple, especially for people with having little coding experience. With the increase of Generative AI, these platforms have become even more powerful, helping to automate coding and speed up the development.

However, AI-generated applications come with risks like security vulnerabilities, bias, and difficulties in scaling and maintaining them over time.

Despite AI's growing role in LCNC, there is no clear way to ensure the security and reliability of AI-generated apps. Also, there is small research comparing AI-powered LCNC platforms with traditional LCNC development, making it unclear if AI truly improves the process.

This research aims to bridge these gaps by identifying security risks, studying AI's real impact on LCNC development, and proposing solutions to make AI-generated applications safer, more scalable, and more reliable for future use.

2 Related work:-

Earlier research has explored the application of Generative AI (GenAI) on Low-Code/No-Code (LCNC) platforms, i.e., its scope to extend automation and reduce reliance on traditional programming. LCNC tools infused with AI have demonstrated the potential to make software development more productive, workflows better optimized, and debugging better assisted[2]. Advanced AI models such as GPT-4 and OpenAI Codex allow automatic code generation, UI/UX design refinement, and application development simplification[5]. Even with these advantages, several challenges remain that require further investigation.

Previous Research studies have looked up how the LCNC platform has grown over the last decades and its effect on software Development. In the early research, they highlighted how these platforms help users with minimal technical knowledge to develop different kinds of applications using visual programming interfaces, drag-and-drop tools, and pre-installed templates[4]. LCNC platform provides greatly improved software accessibility and becomes more efficient. However in traditional systems, we find limitations in customizing and enterprise integration, and researchers have studied how Artificial Intelligence will improve the software quality and optimize workflows and debugging mechanisms [1]. However, it creates some issues like security vulnerabilities, bias, and maintenance issues.

One of the greatest concerns in AI-based LCNC development is security vulnerabilities. Since AI code can upgrade performance and efficiency, it may also introduce hidden bugs, and applications are displayed to unauthorized access, data theft, and cyber-attacks. It has been set up that AI-generated applications are open to have poor authentication schemes, open APIs, and injection attacks without having required security measures taken [9]. Some solutions have been proposed, including AI-based security audits, vulnerability scanners, and compliance requirements. No uniform process for guaranteeing AI-generated LCNC application security currently exists and is at the top of the priority list for developers and researchers.

A major challenge in AI-powered LCNC development is bias and ethical issues. Since generative AI learns from large data sets, which include different kinds of biases. This can cause inequitable decisions and unintentional discriminatory outcomes. These issues particularly can be seen in different kinds of sectors like finance, healthcare, and recruitment, where biased AI-generated applications could highly impact organizations [6]. Researchers are working on a way to fix this with techniques like explainable AI, bias detection algorithms, and teaching AI to be fairer. However, it is a tough problem to make AI systems more transparent and fair [7].

Different kinds of real-world strategies to decrease bias in AI are still being developed and fair rules are not ready yet.

AI-generated applications face challenges in scalability and maintainability. Artificial intelligence can speed up the development process, but the apps which are generated by AI are scalable, adaptable, and maintaining over time is challenging. The code that is generated frequently lacks clear documentation and structure, making it hard to modify and maintain [10]. Some of the solutions like automated documentation, artificial intelligence-based refactoring, and AI-human collaboration aim to improve its adaptability. Even with these improvements, human oversight is still needed to ensure long-term reliability and sustainability for AI-generated LCNC applications.

To address such problems, many best practices have been proposed, including automated security scanning, fairness frameworks for AI, and governance models for AI [3]. Researchers observe that LCNC applications that use artificial intelligence should be developed with stringent security controls, human-in-the-loop validation, and bias mitigation measures to prevent uninvited risk. However, a universally agreed-upon framework for incorporating Generative AI in LCNC platforms has yet to be developed. Future research must focus on the development of regulatory standards, improving AI explainability, and optimizing the AI-human collaboration to improve the security, fairness, and scalability of AI-based LCNC applications.

While LCNC platforms have been the subject of thorough studies, empirical research on the convergence of AI and LCNC security threats is limited. The majority of research compares AI as an enhancement without analyzing its shortcomings critically. This work fills the gap by methodically assessing security threats, AI bias, and maintainability concerns using experiments and case studies.

2.1 Research Gap:-

While Generative AI in Low-Code/No-Code (LCNC) platforms has significantly boosted software development and made it more powerful, security problems are the greatest headache. Apps generated by AI have built-in security flaws such as weak authentication, open APIs, and vulnerability to cyber-attacks. Such flaws leave AI-generated LCNC applications vulnerable to security breaches. Although some security products based on AI are out there, no standard process is available to test and confirm the security of AI-generated applications[9][5][3]. Current research is not yet presenting a comprehensive solution for automatically identifying risks and ensuring security through AI, and LCNC applications continue to be vulnerable to cyber-attacks[1]. Additional research would be needed to implement AI system-based security policies, automated security audits, and real-time vulnerability detection frameworks in enhancing the credibility and reliability of AI-generated LCNC applications[4][10].

Another big issue with AI-powered LCNC platforms is bias and fairness. AI is developed based on the data it is trained with, and if the data is biased, the AI-created application will probably also be unfair. This is a very important problem in finance, healthcare, and hiring, where fairness is very important[7][10]. There are some tools to measure AI fairness, but no standard method to guarantee that AI-generated applications are fair and transparent[6]. There is also limited literature on AI ethics rules and standards in LCNC platforms. Future research

should focus on reducing AI bias, developing fairness standards, and improving AI transparency to guarantee more trustworthy and fair AI-based applications[8].

Another major research gap in AI-generated LCNC applications is scalability and long-term maintainability. LCNC platforms enable applications to be developed quickly, but scaling them to large businesses and maintaining them long-term is a big issue. AI-generated code is generally hard to read, poorly documented, and hard to modify, making it harder for developers to debug issues, update features, or improve applications[3][10]. Most LCNC tools are sufficient for small projects but break down when businesses need scalable and complex solutions. Not enough work has been conducted on how to make AI-generated code maintainable and scalable[2]. Future research should be focused on developing AI-generated code quality, creating better AI-augmented documentation tools, and architecting AI systems that help maintain and evolve applications over time[1].

3 Research Methodology:-

3.1 Research Questions:-

RQ1:- What are the most critical security vulnerabilities in LCNC applications generated by AI, and how do they affect system integrity?
Research Methodology:- Case Study.
Justification:-Case studies enable tracking real-world security vulnerabilities, to analyze in depth how LCNC applications developed by AI introduce security flaws. Such a method provides patterns of vulnerability and their long-term impact on application integrity.
Other methods:- Survey: Surveys can be helpful in gathering widespread developer experience with security threats but do not provide technical information regarding real attacks.

RQ2:-In what ways is AI bias in LCNC applications impacting fairness and decision-making in different industries?
Research Methodology:-Survey.
Justification:-Surveys help allow feedback collection from end-users and developers regarding how AI-driven LCNC applications can create bias in highly sensitive fields like finance, health, and employment. Surveys detect perceived bias without quantitatively measuring technical bias in AI models.
Other methods:- Experiment: Through controlled experiments, it is possible to evaluate the AI bias directly, but with no big industry takeaways.

RQ3:- What are the AI-generated LCNC application scaling problems and how do they affect maintainability in the long term?
Research Methodology:-case study.
Justification:-Case studies provide true-life examples of scalability issues in LCNC solutions based on AI. The research approach allows one to study how organizations handle long-term system updates and code maintenance.
Other methods:-Survey:Surveys offer overall developer views of scalability issues but lack technical technicality of maintainability.

RQ4:-How effective are present AI-powered security solutions in detecting and blocking

vulnerabilities in LCNC applications?
Research Methodology:-Experimental Study
Justification:- Experimenting allows for direct assessment of AI-based security tools in detecting and preventing security threats in LCNC platforms. The method provides measurable information on the effectiveness of existing security mechanisms against real-world attacks.
Other methods:-Case Study: Case studies analyze past security attacks on LCNC applications without providing controlled experimentation of security solutions.

The formulated research questions address the key limitations identified in the literature review. These questions are designed to be specific and measurable, ensuring practical and actionable outcomes. For instance, RQ1 (security vulnerabilities) will be answered using penetration testing, while RQ2 (bias in AI) will involve controlled experiments with diverse datasets.

4 Study Design:-

In order to understand how Generative AI influences Low-Code/No-Code (LCNC) development, we begin by reviewing existing research on AI-enhanced LCNC platforms. The focus will be put on enhancing the speed of software development, security, and scalability using AI and discovering potential risks like security vulnerabilities, AI bias, and long-term maintainability challanges.

The existing literature shows that AI can effectively generate code, debug, and optimize processes. Security issues and ethics are also concerns with applications developed using AI. There is also a lack of standardized frameworks to ensure AI-generated LCNC applications are safe, fair, and scalable. This literature review will help identify these gaps and build the foundation for our research.

Additionally, recent comparisons between AI-based LCNC and traditional LCNC platforms are limited, and therefore it is not clear if AI actually improves efficiency and software quality. This review will also address how AI can be used to detect security vulnerabilities, reduce bias, and increase the scalability of LCNC applications.

By examining past research we aim to understand where AI-powered LCNC platforms succeed and where they still face significant challenges.This will guide our experimental approach, enabling us to validate AI's effectiveness in real LCNC application.

This study has undergone a number of revisions after receiving helpful feedback from supervisors and colleagues. The key enhancements include a more comprehensive literature review, better structured methodology, and additional validity checks to ensure strength. Further revisions will take on board feedback from stakeholders who are industry practitioners to enhance real-world applicability.

Data Collection Methods:
- Surveys: Online questionnaires will be distributed to LCNC developers and end-users to gather insights on security concerns, AI bias perception, and scalability challenges. Questions will use a combination of Likert scales and open-ended responses.

- Case Studies: Real-world examples of LCNC applications using Generative AI will be analyzed to identify security vulnerabilities and scalability issues. These will involve analyzing code, system architecture, and user feedback.

- Experimental Studies: AI-powered security tools will be tested against known vulnerabilities in LCNC applications to measure their effectiveness in detecting and preventing attacks. Penetration testing techniques will be used.

Ethical Considerations: - Informed Consent: Participants in surveys and interviews will be informed about the study's purpose and data usage. - Data Privacy: Anonymity and confidentiality will be ensured to protect participant data. - Bias Mitigation: Efforts will be made to minimize bias in survey design and data analysis.

4.1 Experiment:-

4.1.1 Variables in Experiment 1: Efficiency of AI-Enhanced LCNC Development:-

The first experiment points to measure how much AI boosts development speed, automation, and ease of use in LCNC platforms compared to traditional LCNC development. It will analyze how well AI-generated applications decrease development time and effort while maintaining software quality.

Setup:-

- Select multiple AI-powered LCNC platforms (e.g., Microsoft PowerApps, Mendix, OutSystems).

- Challenge test users with two tasks: one AI-based LCNC and another with regular LCNC.

- Measure development time, automation capabilities, and error rates in AI-assisted vs. traditional LCNC applications.

- Conduct a user survey to assess ease of use and user satisfaction with applications developed by AI.

variables

Subject	**To examine how AI impacts development speed and automation in LCNC.**
Treatments	Comparing AI-enhanced LCNC tools with traditional LCNC platforms.
Dependent Variables	Development time, error rates, ease of use, and automation effectiveness.
Independent Variables	AI-driven automation tools, platform usability features.
Control Variables	Application complexity, user experience level, platform-specific limitations.

4.1.2 Variables in Experiment 2: Security Risks in AI-Generated LCNC:-

The second experiment is meant to determine security issues in AI-created applications, such as authentication flaws, open APIs, and cyberattack risks. The objective is to determine whether AI-created LCNC applications introduce hidden security risks compared to manually developed LCNC applications.

Setup:-

- Develop comparable applications using AI-aided LCNC tools and traditional LCNC methods.
- Automated security testing (penetration testing, API vulnerability scanning) on each set of applications.
- Compare security vulnerabilities that exist in AI-generated vs. human-generated applications.
- Evaluate the effectiveness rate of AI-based security software in detecting vulnerabilities.

Variables

Subject	**To analyze the security risks of AI-generated LCNC applications.**
Treatments	Comparing AI-enhanced LCNC applications with traditionally developed LCNC applications.
Dependent Variables	Number of security vulnerabilities, authentication flaws, API exposure risks.
Independent Variables	AI-generated code security measures, vulnerability detection tools.
Control Variables	Application type, testing tools used, industry security standards.

4.1.3 Variables in Experiment 3: Scalability & Long-Term Maintainability of AI-Generated LCNC Applications:-

The third test will determine if AI-created LCNC applications can scale well and how maintainable they are in the long term. It will examine code quality, documentation, and flexibility in AI-created apps versus those created using traditional means.

- Choose applications developed with AI-assisted LCNC and traditional LCNC approaches.
- Measure scalability factors, such as performance under increased load and ability to integrate with enterprise systems.
- Analyze AI-generated code for documentation clarity and maintainability.
- Conduct developer surveys to measure how easily AI-code is to modify and keep up with in the long term.

Variables

Subject	**To evaluate the scalability and maintainability of AI-generated LCNC applications.**
Treatments	Comparing AI-generated LCNC applications with traditional LCNC applications.
Dependent Variables	Scalability under high usage, documentation clarity, ease of maintenance.
Independent Variables	AI-generated code structure, integration capabilities.
Control Variables	Application type, industry use case, testing environment.

4.1.4 Validity Threats:-

Internal Validity:- This stuy depends upon the quality of AI-generated applications and how well testers evaluate them. In the event the AI produces inconsequent results or the testers are novices, the findings would not be accurate.
Mitigation:- We will ensure careful attention to the research process, provide proper training to testers, and follow a standard procedure to identify correct ratings.

External Validity:- If we only test a few AI-powered LCNC applications in specific industries, the results may not apply to all types of software development.We need to ensure that our outcomes are beneficial for other practical applications.
Mitigation:- We will test AI-powered LCNC tools across various industries like healthcare, finance, and e-commerce to get a broader perspective. We will also use different AI models to

avoid bias.

Construct validity:- Unless we use the right tool to measure AI's efficiency, security, and scalability, we might end up with false conclusions. For example, if we measure speed only and ignore security threats, we will not have the whole picture.
Mitigation:- We will select our key performance indicators like development time, number of security bugs found, and scalability metrics carefully to ensure we're measuring AI's effect properly. Experts will also review our chosen metrics before testing.

Conclusion Validity:- As we would be testing very few applications created by AI, the conclusions won't be valid. Additionally, unexpected external factors could affect the results, making them less trustworthy.
Mitigation:- We will increase the sample size by conducting more tests on AI-created LCNC applications. We will also attempt to regulate external variables to the highest degree to base our conclusions on strong data.

4.1.5 Data Analysis:-

Thestudy collects information in three major ways. For case studies, information is gathered via interviews, company records, and security assessments of LCNC implementations. The data is analyzed to look for commonproblems and possible solutions. For the surveys, the professionals are questioned on their experience with AI-enabled LCNC applications through structured questionnaires. Their answers are evaluated through statistical analysis and trend analysis to decide whether patterns are connected to security, bias in AI, and scalability. In tests, AI-created LCNC programs are processed through security tools to identify how efficiently they handle threats. The effectiveness of different security protocols is compared to identify the strengths and weaknesses of AI-created software. This combination of techniques helps to gain a proper and complete understanding of the topic

Beyond technical analysis, this study considers ethical implications such as transparency, accountability, and AI governance. AI-generated LCNC applications may introduce biases, and this study aims to propose a bias-mitigation framework informed by industry standards and AI fairness guidelines.

5 Expected Outcomes:-

This study will assist in the understanding of how Generative AI improves and challenges Low Code/No-Code (LCNC) development. It will also consider security issues, AI bias, scalability, and maintenance in the long run while providing working solutions for organizations and developers. Some of the significant outputs are the determination of security threats from AI developed LCNC applications, which cover poor security, open APIs, and cyberattacks. AI security tools will be analyzed through this research with appropriate suggestions to utilize better protective strategies to make AI-generated apps secure and reliable. Another important outcome is reducing AI bias in LCNC applications. Since AI can deliver unfair outcomes based on biased data, the study will be involved on ways of improving AI training and fairness detection to deliver more ethical and unbiased AI-based solutions. The study will also take into

account scalability and maintenance issues. AI-based applications are not well-documented and, hence hard to maintain and update. The study will suggest AI-based tools and best prac tices that will ease the maintenance of LCNC applications in the long run. Finally, the study will examine how effective security solutions are based on AI are in secure LCNC applications from attacks. It will measure their performance and suggest enhanced security frameworks for stronger protection against cyber attacks. In total, this study will allow businesses and develop ers to adopt AI in LCNC development securely and efficiently by providing practical solutions, security recommendations, and best practices for developing secure, equitable, and scalable AI driven applications

Time Activity Plan

S.No	Activity	Start Date	End Date	Days Allotted
1	Selecting a topic	20/01/2025	25/01/2025	6
2	Collecting related research papers	26/01/2025	10/02/2025	16
3	Gathering information and identifying research gaps	11/02/2025	20/02/2025	10
4	Formulating research questions and planning study design	21/02/2025	28/02/2025	8
5	Obtaining feedback from the supervisor	01/03/2025	04/03/2025	4
6	Submitting research proposal	05/03/2025	06/03/2025	2
7	Performing a systematic literature review on related works	07/03/2025	21/03/2025	14
8	Conducting Experiment 1: Evaluating AI-driven LCNC efficiency	22/03/2025	11/04/2025	20
9	Conducting Experiment 2: Security assessment of AI-generated LCNC apps	12/04/2025	26/04/2025	15
10	Conducting Experiment 3: Scalability and maintainability of AI-generated LCNC apps	27/04/2025	12/05/2025	16
11	Analyzing results and summarizing findings	13/05/2025	22/05/2025	10
12	Writing and reviewing thesis	23/05/2025	06/06/2025	15
13	Obtaining feedback from the supervisor	07/06/2025	12/06/2025	6
14	Making changes based on supervisor feedback	13/06/2025	18/06/2025	6
15	Presentation of Thesis	19/06/2025	24/06/2025	6
16	Submitting the final draft of thesis	25/06/2025	30/06/2025	6

6 Risk Management

Risk 1: Expanding Research Scope

Impact: If the research covers too many areas, it may become difficult to manage, leading to scattered findings and difficulty in drawing meaningful conclusions.

Mitigation: Keep the study focused on key aspects such as *AI security, efficiency, and scalability* in Low-Code/No-Code (LCNC) platforms. Clearly define research objectives and set boundaries to ensure the study remains manageable.

Risk 2: Security Vulnerabilities in AI-Generated LCNC Applications

Impact: AI-generated LCNC applications may contain security flaws like weak authentication, exposed APIs, and potential data breaches, increasing cybersecurity risks.

Mitigation: Implement regular security testing using *automated vulnerability scans and manual penetration testing*. Ensure that AI-generated applications follow established security guidelines and best practices.

Risk 3: Bias in AI-Generated Applications

Impact: Since AI learns from existing datasets, it might introduce biases into LCNC applications, leading to unfair or incorrect decisions in critical areas such as *finance, healthcare, or hiring systems*.

Mitigation: Use *bias detection algorithms, diverse datasets, and fairness assessments* to reduce bias. Ensure AI-generated applications are tested for fairness and transparency before deployment.

Risk 4: Scalability and Maintainability Issues

Impact: AI-generated applications may work well initially but struggle with long-term maintenance and scalability, making it difficult to expand or modify them later.

Mitigation: Evaluate AI-generated code for *readability, documentation quality, and ease of modification*. Ensure that applications are tested for *scalability under high workloads* and can integrate seamlessly with existing enterprise systems.

Risk 5: Data Loss or Technical Issues

Impact: Losing research files or encountering technical failures in AI-powered LCNC platforms could delay progress and affect the accuracy of results.

Mitigation: Maintain *regular backups* of all research data on *cloud storage and external drives*. Use version control for documentation and set up alternative solutions in case of unexpected system failures.

References

[1] A. Bahi, J. Gharib, and Y. Gahi. "Integrating Generative AI for Advancing Agile Software Development and Mitigating Project Management Challenges". In: *International Journal of Advanced Computer Science and Applications* 15.3 (2024). URL: https://thesai.org/Publications/ViewPaper?Volume=15&Issue=3&Code=IJACSA&SerialNo=2.

[2] C. Ebert and P. Louridas. "Generative AI for Software Practitioners". In: *IEEE Software* 40.4 (2023), pp. 30–38. URL: https://ieeexplore.ieee.org/document/10123456.

[3] K. Kalluri. "Artificial Intelligence in BPM: Enhancing Process Optimization Through Low-Code Development". In: *International Journal for Multidisciplinary Research* 5 (2023). DOI: 10.36948/ijfmr.2023.v05i06.23396.

[4] L. Korada. "Low Code/No Code Application Development - Opportunity and Challenges for Enterprises". In: *ResearchGate* (2023). URL: https://www.researchgate.net/publication/370799345_Low_CodeNo_Code_Application_Development_-_Opportunity_and_Challenges_for_Enterprises.

[5] A. Mehra. "Integrating Generative AI into the Software Development Lifecycle: Impacts on Code Quality and Maintenance". In: *International Journal of Advanced Computer Science and Applications* 15.3 (2024). URL: https://thesai.org/Publications/ViewPaper?Volume=15&Issue=3&Code=IJACSA&SerialNo=1.

[6] O. Ogundare, G. Q. Araya, and Y. Qamsane. "No Code AI: Automatic Generation of Function Block Diagrams from Documentation and Associated Heuristic for Context-Aware ML Algorithm Training". In: *arXiv preprint arXiv:2304.04117* (2023). URL: https://arxiv.org/abs/2304.04117.

[7] G. Paliwal et al. "Low-Code/No-Code Meets GenAI: A New Era in Product Development". In: *IEEE 8th Ecuador Technical Chapters Meeting (ETCM)*. 2024. URL: https://ieeexplore.ieee.org/document/10746160.

[8] Usman Rafiq, Cenacchi Filippo, and Xiaofeng Wang. "Understanding Low-Code or No-Code Adoption in Software Startups: Preliminary Results from a Comparative Case Study". In: *International Conference on Product-Focused Software Process Improvement*. Springer, 2022, pp. 390–405. DOI: 10.1007/978-3-031-21388-5_27.

[9] N. Rao et al. "AI for Low-Code for AI". In: *arXiv preprint arXiv:2305.20015* (2023). URL: https://arxiv.org/abs/2305.20015.

[10] L. H. M. Wong and R. M. Davison. "Combining Low-Code/No-Code with Noncompliant Workarounds to Overcome a Corporate System's Limitations". In: *MIS Quarterly* (2023). URL: https://misq.org/combining-low-code-no-code-with-noncompliant-workarounds-to-overcome-a-corporate-system-s-limitations.html.