NOVA IMS Information Management School

MDDM

Master's Degree Program in Data-Driven Marketing

Factors Influencing User Acceptance and Adoption Among Different User Groups of Low-code Development Platforms

Elif Uslu

Master Thesis

presented as partial requirement for obtaining a Master's Degree in Data-Driven Marketing

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

Factors Influencing User Acceptance and Adoption Among Different User Groups of Low-code Development Platforms

by

Elif Uslu

Master Thesis presented as partial requirement for obtaining the Master's degree in Data-Driven Marketing, with a specialization in Marketing Intelligence

Supervised by

Vítor Santos, Assistant Professor

Jorge Carrola Rodrigues, Invited Assistant Professor

November, 2024

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism, any form of undue use of information or falsification of results during the process leading to its elaboration. I further declare that I have fully acknowledged the Rules of Conduct and Code of Honor from the NOVA Information Management School.

[Lisbon, 30.11.2024]

DEDICATION

To my ever-supportive family and friends, who have been my personal cheerleaders; to Bruno who endured my late-night rants and provided endless encouragement; to my dog, Gigi, who offered unconditional love and wagging-tail therapy during the toughest moments; and to Marisa, for being the ultimate provider of mental support and occasional reality checks. This work is as much yours as it is mine, though Gigi, you're not getting any royalties.

ACKNOWLEDGEMENTS

First and foremost, I extend my deepest gratitude to Prof. Jorge and Prof. Vitor, for guiding me through this journey with wisdom, patience, and just the right amount of constructive critique to keep me on my toes. Your support has been invaluable, and this work would not have been possible without your support and expertise.

To my managers at work, thank you for understanding the demands of this thesis and providing me with the flexibility to juggle it all. Your accommodation and encouragement have meant the world to me, allowing me to stay focused and motivated throughout this process.

To my survey participants, your contributions were the backbone of this research. Thank you for taking the time to share your insights and for helping to make this thesis both meaningful and possible.

ABSTRACT

This thesis investigates the factors influencing user acceptance and adoption of Low-Code Development Platforms (LCDPs) among diverse user groups with varying technical backgrounds. By utilizing a modified Technology Acceptance Model (TAM) as a conceptual framework, the study identifies key drivers and inhibitors affecting the adoption of LCDPs. Through a systematic literature review and a survey-based empirical analysis, the research explores constructs such as perceived usefulness (PU) and perceived ease of use (PEOU) and their relationships with behavioral intention (BI) and actual use (AU).

Findings reveal that PEOU and PU significantly predict behavioral intention to adopt LCDPs, with technical job roles exhibiting stronger adoption rates. The study also highlights organizational and technical barriers, including security concerns, limited customization, and vendor lock-in, which inhibit widespread adoption. Conversely, the empowerment of non-technical users, cost-efficiency, and rapid application development emerge as critical enablers.

The results contribute to both academic literature and practical applications by offering insights into how organizations can promote LCDP adoption to accelerate digital transformation and innovation. Future research directions are proposed to address limitations, such as expanding the sample size and exploring additional contextual variables. This study provides actionable strategies for fostering inclusive, efficient, and scalable adoption of LCDPs across technical and non-technical user groups.

Keywords: Low-code; Development platforms; User acceptance; User adoption; Business innovation; Marketing

Sustainable Development Goals (SDG):

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AU: Actual Use

BI: Behavioral Intention

CRUD: Create, Read, Update, Delete

DSL: Domain-Specific Language

FL: Federated Learning

HTMT: Heterotrait-Monotrait Ratio

IDE: Integrated Development Environment

IT: Information Technology

LCD: Low-Code Development

LCDP: Low-Code Development Platform

LCP: Low-Code Programming

MDD: Model-Driven Development

MDE: Model-Driven Engineering

MDSE: Model-Driven Software Engineering

MSA: Microservice Architecture Style

PEOU: Perceived Ease of Use

PU: Perceived Usefulness

RAD: Rapid Application Development

SRMR: Standardized Root Mean Square Residual

STS: Socio-Technical Systems

TAM: Technology Acceptance Model

TOE: Technology-Organization-Environment

UTAUT: Unified Theory of Acceptance and Use of Technology

VIF: Variance Inflation Factor

1. INTRODUCTION

Low-Code Development Platforms (LCDPs) are gaining traction as a solution to the increasing demand for rapid application development (RAD), enabling users without programming expertise to contribute effectively to software creation (Hoogsteen & Borgman, 2022). While their adoption continues to grow, driven by cost savings and efficiency (Mehta, 2022), there remain gaps in understanding the factors influencing acceptance across diverse user groups. This thesis explores these factors, focusing on user adoption through the lens of the Technology Acceptance Model (TAM).

1.1. BACKGROUND AND PROBLEM IDENTIFICATION

LCDPs are transforming application development by allowing developers to create applications with minimal hand coding (Outsystems, n.d.). These platforms address business innovation backlogs by empowering non-programmers to participate in software development (Hoogsteen & Borgman, 2022).

The global market for low-code technologies is forecasted to reach $26.9 billion in 2023, a 19.6% increase from the previous year. Gartner analysts emphasize that rising adoption is driven by the high cost of tech talent, hybrid workforces, and the increasing demand for fast application delivery and tailored automation workflows. Low-code tools enable collaboration between IT and business technologists, enhancing productivity, agility, and digital competency. These platforms support the delivery speed essential in today's agile business environment (Gartner, 2022).

1.2. RESEARCH GAPS

A review of literature on low-code platform adoption reveals a research gap in understanding the specific needs of diverse user groups with varying technical backgrounds.

While studies address developer challenges and adoption drivers, little attention is given to promoting adoption across user demographics. Future research should explore factors influencing user acceptance in different business contexts and develop strategies to enhance adoption among varied user groups.

1.3. OBJECTIVES

This research aims to identify factors influencing user acceptance of low-code platforms among diverse user groups and to develop strategies for enhancing adoption. In order to achieve the main goal, the following steps were defined: 1) Make a comprehensive study on low-code process development and its drivers and inhibitors; 2) Study the most relevant adoption models; 3) Build and run a survey on low-code adoption; 4) Select and run the appropriate adoption model; and 5) Analyze the results.

1.4. IMPORTANCE AND RELEVANCE

The adoption of low-code platforms has significant implications for the economy, society, and science. By identifying factors influencing user acceptance, research can drive faster application delivery, enhance productivity, and promote inclusivity, benefiting both technical and non-technical users. This fosters organizational efficiency, digital competency, and market growth while bridging the digital divide. Scientifically, understanding adoption challenges and strategies informs broader acceptance and supports advancements in digital transformation, innovation, and inclusivity, contributing to knowledge development and addressing research gaps in low-code adoption.

2. LITERATURE REVIEW

2.1. BACKGROUND ON LOW-CODE PROCESS DEVELOPMENT

2.1.1. CONCEPTS AND HISTORY

Low-code is a software development approach to coding that allows users to create and manage applications quickly with minimal manual coding required (Santis, n.d.). Rather than writing lines of intricate and complex code with traditional computing programming, low-code development uses drag-and-drop visual modelers and point-and-click interface creation to create completed apps rapidly. Whereas no-code platforms offer a range of essential features suitable for different application types that users can customize these functionalities through graphical or form-based interfaces to create business-specific end-user experiences. The graphical interfaces allow users to choose, organize, configure, and connect elements from pre-built libraries and third-party plugins. “The difference between low-code and no-code is often merely in how it is used and may be a matter of perception: For example, Excel transitions from no-code to low-code when macros are used, but one might also say that the use of conditions in formulas means that Excel should not be considered no-code.” (Steffen & Margaria, 2021).

Built on the foundation of coding modulization and visualization, low-code platforms appeal to demographics of all levels, regardless of skill or familiarity with business procedures and operations that anyone can build on low-code platforms (Low-code Development Guide, n.d.). Those individuals who operate outside the Information Technology (IT) department and lack formal programming expertise are called citizen developers who utilize low-code and no-code platforms to design applications tailored to the specific requirements of their work units or entire organizations (Lebens & Finnegan, 2021).

The key features of LCDPs center on the development of databases, business processes, and web-based user interfaces. Rooted in the ideology of fourth-generation programming (4GL) and RAD concepts. The inception of the Low-code concept in 2011 is regarded as a pioneering and innovative development in the programming domain. The methodologies that underpin Low-code Programming encompass model-driven software development (MDSE), RAD, automatic code generation, and visual programming (Waszkowski, 2019).

The historical roots of low-code trace back to Apple's Hypercard, an early platform fostering diverse applications. The World Wide Web also initially started as a low-code approach for static hypertext information systems before evolving into a high-code platform on both the front and back ends. In the 1970s, scripting languages used by Unix/Linux programmers and Apple Macintosh end-users reflected a motivation similar to the low-code vision, facilitating task automation. Over 40 years ago, spreadsheets became foundational, and the 1980s saw the marketing of 4GLs like Natural and Sperry's Mapper. These languages presented benefits to contemporary low-code platforms, envisioning business executives generating analytic reports from mainframe databases. The rise of MDE and Unified Modeling Language (UML) in the 1990s offers another perspective on the low-code concept. Describing an application's data in a class model and its behavior in state machine diagrams theoretically reduces the reliance on textual coding, aligning with the vision of MDE proponents (Steffen & Margaria, 2021).

Forwarding to recent developments in Low-code Platforms, a term initially coined by Rymer & Richardson (2015) offers potential solutions to challenges in software development by introducing higher-level abstractions, such as domain-specific models or languages. LCP draw inspiration from research domains like Domain-Specific Languages (DSL) and MDE,

with some scholars noting the absence of radical innovations while others view LCP as an opportunity to consolidate existing knowledge and techniques (Overeem, 2022, p.4).

To support citizen developers in creating increasingly large and complex software systems, LCDPs must incorporate approaches facilitating the development of such systems. Architectural patterns like the microservice architecture style (MSA) and event-driven architectures play a crucial role. MSA involves developing interconnected, smaller software systems instead of a monolithic one, while event-driven architectures enable asynchronous communication between different parts of a software system, fostering a loosely coupled environment. These architectural patterns are being integrated into LCP designs, allowing citizen developers to build scalable and manageable solutions (Overeem, 2022).

RAD is an approach to information systems development that can be summed up in one sentence: the commercial need to deliver working business applications in shorter timescales and for less investment. “A number of people see RAD as a complete approach to information systems development in that it covers the entire life cycle, from initiation through to delivery”(Beynon-Davies et al., 1999).

RAD prioritizes continuous development, swiftly creating prototypes for iterative user feedback whereas traditional plan-based models, rooted in physical engineering practices, proved challenging to adapt effectively to software production. Low-code and RAD share the common goal of speeding up software development but differ in their approaches. Low-code, a visual programming trend, allows domain experts to create applications, while RAD is a broad approach transforming traditional development cycles with rapid prototyping and user testing. RAD's use of low-code/no-code platforms depend on a team's chosen strategies.

Low-code is a technology, tied to specific platforms, while RAD is a conceptual process, a term encompassing modern adaptive software approaches. RAD specifies steps but

not a specific framework, language, or platform. RAD is typically embraced by professional programmer teams, collaborating with subject matter experts, whereas low-code often empowers domain experts to construct software, although low-code tools can expedite RAD processes (Bill Doerrfeld, 2022).

2.1.2. BENEFITS AND CHALLENGES

Low-code Development (LCD) concept appears to address the previously outlined issues by offering several advantages. Most LCD tools facilitate development through a graphical user interface, minimizing the necessity for extensive source code writing and, consequently, reducing the skill level required for software development. This democratizes software development, enabling individuals with limited technical expertise to create their applications. Additionally, LCD tools streamline the development process by allowing developers to focus on program logic rather than managing libraries and other technologies. This emphasis on program logic can lead to a reduction in time spent on handling dependencies, ultimately enhancing efficiency and lowering development time. These dual benefits of increased accessibility and efficiency also have the potential to alleviate the shortage of qualified developers (Pinho et al., 2023).

LCD aims to democratize the process, allowing domain experts to contribute and expedite development and deployment to mitigate the challenges of expensive training and hiring in the rapidly evolving software development landscape. LCD seeks to bridge the gap between system requirements and developer constraints, a common cause of prolonged development times in complex business applications. LCD employs a range of methodologies, including visual modeling, rapid app development, model-driven development (MDD), cloud computing, and automatic code generation. By leveraging low-code development tools, the approach enables the creation of production-ready applications

with reduced manual coding through automatic code generation. Additionally, LCD platforms also provide more flexibility, easy development, and maintainability (Di Sipio et al., 2020)Critical architectural decisions are undertaken to ensure minimal coding, rapidity, flexibility, reduced upfront investment, and out-of-the-box functionalities that facilitate the swift delivery of comprehensive applications. Sahay et al. (2020) assert that bug fixing, application scalability, and extensibility are streamlined and maintainable through the utilization of high-level abstractions and models in these platforms.

LCD is commonly used for developing high-performance, database-driven mobile, and online applications across various domains (Alamin et al., 2023).

Low-code development confers a multitude of advantages, as revealed by Martinez & Pfister (2023). Firstly, it enhances speed and agility in application development by minimizing hand coding, thereby expediting the development process, and ensuring the prompt availability of functional applications. Additionally, it leads to substantial cost savings by reducing the resources needed for development, including a decrease in working hours and the elimination of the necessity to hire professional developers. Moreover, low-code contributes to complexity reduction, as applications are built on pre-defined components and templates, streamlining development, and emphasizing the fulfillment of user requirements. The approach also facilitates easier maintenance due to the inherently lighter code, resulting in fewer lines of code and decreased effort in upkeep. Furthermore, low-code encourages higher involvement of business profiles in the development process, allowing individuals closer to operations to translate requirements into application features effectively. It also minimizes unstable or inconsistent requirements through the facilitation of early prototype building for requirement validation, preventing the allocation of resources to unnecessary features and avoiding development loops. Finally, increased privacy is achieved,

given that low-code applications are developed internally, obviating the need for external IT consultants, and ensuring data remains within the organization (Martinez & Pfister, 2023).

Given these advantages, it comes as no surprise that LCD is drawing significant attention from businesses, exerting a positive and transformative impact across various dimensions. However, despite its advantages, LCD comes with its own challenges.

Although LCD aims to minimize hand-coding, LCDPs still require some sort of programming skills which may come as a challenge for citizen developers, whereas lack of complete freedom in customization and limited scalability are some of the concerns that prevent developers from using LCDPs. According to Beranič et al., (2020), one of the most common reasons for the lack of adoption and usability of low-code/no-code development tools is the lack of knowledge about these platforms and tools. Other reasons include the concern for commitment to only one low-code/no-code tool provider and doubts about the feasibility of the mentioned approach when building their applications. Some of them were also concerned about the scalability and security of the application developed.

LCD practitioners face challenges in application maintenance and deployment. Vendors need to focus on addressing these issues. This affects how developers prioritize efforts during software design, development, and deployment using low-code software development (LCSD) platforms. Inadequate support for scalable usage and deployment can lead developers to seek alternatives with better support. LCSD platforms have limitations in customization and debugging due to their abstraction and feature constraints (Alamin et al., 2023).

Adopting LCD comes with commitments, as shifting vendors or platforms can be costly, particularly considering the extent of prior developments. Another challenge is the

limited extensibility, primarily stemming from the proprietary nature of these platforms (Sahay et al., 2020).

According to the comprehensive literature review conducted by Rokis & Kirikova (2022), low-code platforms present several notable challenges across software development phases. Requirements analysis faces issues in varying specifications, changing requirements, and vendor lock-in concerns, mitigated by minimum viable product development and low-code adoption. The Planning phase introduces challenges in platform selection, costs, and learning curves, with mitigations emphasizing criteria-based comparisons.

Moving to Application Design, challenges include extensibility, interoperability limits, scalability, and UI/data storage complexities. Proposed mitigations involve standards, considering extensibility in selection, and reducing knowledge gaps. The Development phase presents challenges in UI customization, business logic, integration, and debugging, mitigated by elaborate documentation, learning resources, and automated tools.

Testing phase challenges include limited support, third-party tool dependence, and neglect of non-functional requirements. Mitigations involve detailed documentation, third-party tool integration, and a low-code testing framework. Deployment phase challenges relate to configuration, accessibility, and performance, with strategies like improved documentation and repositories for version control.

Maintenance phase challenges include debugging and maintenance features, mitigated by comprehensive documentation and information availability. The literature emphasizes collaborative efforts for standards and advancements in low-code technologies (Rokis & Kirikova, 2022).

Other limitations of low-code development, as identified by the authors Martinez & Pfister (2023) include constraints on customization beyond the offerings of the low-code platform itself. Users are restricted to the functionalities provided by the platform, limiting their flexibility compared to traditional programming approaches. Additionally, scalability poses a challenge, with low-code primarily suitable for small applications, making it difficult to implement enterprise-level solutions. Fragmentation across various low-code systems from different vendors creates obstacles to seamless data integration. Vendor lock-in is a significant concern, as companies may find themselves compelled to continue using a specific vendor's environment after apps have been developed. High licensing costs emerge as an issue, particularly when scaling the app to a larger user base. Furthermore, the potential neglect of security considerations during app development by various individuals within the organization poses a threat to overall IT security (Martinez & Pfister, 2023).

2.1.3. TOOLS

A systematic literature review by Pinho et al. (2023) defines usability in LCDs as the "extent to which a system, product, or service can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction." The Nielsen-Norman Group (NNG) further categorizes usability into five quality components: efficiency, learnability, memorability, satisfaction, and error rates. LCDP vendors display a range of applications that users can build, from mobile apps to CRUD (Create, Read, Update, Delete) backend systems, applicable across sectors like education, retail, government, finance, and critical systems. Four primary LCDP market leaders identified are Mendix, Microsoft Power Apps, OutSystems, and Salesforce, with additional players such as Appian, Google App Maker, Kissflow, and ServiceNow noted in recent Gartner and Forrester reports (Sahay et al., 2020).

The following is an overview of eight key LCDP vendors and their usability features:

1. **Appian**: One of the oldest platforms, offering mobile and web app development with personalization tools, collaboration, task management, and a decision engine for complex logic modeling.
2. **Google App Maker:** Built on G Suite, it supports custom enterprise applications with drag-and-drop templates and integration using HTML, JavaScript, and CSS.
3. **Kissflow**: A workflow automation tool focused on small businesses, integrating with third-party APIs like Zapier and Office 365 for automated workflows.
4. **Mendix**: A no-code platform emphasizing collaborative, visual development with Docker and Kubernetes compatibility, pre-built connectors, and templates for rapid deployment.
5. **Microsoft Power Apps**: Offers a mix of model-driven and canvas approaches with strong Microsoft ecosystem integration, supporting legacy systems and third-party connectors.
6. **OutSystems**: Enables desktop and mobile app development for cloud and local deployment, with simplified publishing and extensive application support for billing systems, CRMs, and operational dashboards.
7. **Salesforce App Cloud**: Supports scalable cloud-based apps, with tools for automation, an AppExchange marketplace, and drag-and-drop builders.
8. **Zoho Creator**: Provides GUI-based development with drag-and-drop capabilities, suitable for web and workflow-centric applications with integrations to other Zoho and Salesforce services.

In addition to these vendors, more information on additional LCDP providers, including specific capabilities and use cases, is provided in the appendix table for a broader understanding of the LCDP landscape.

2.1.4. COST

The role of economics in driving the usage of LCP is critical. It includes acquisition expenses, license fees, cost savings gained from increased productivity, training costs, and maintenance and deployment costs. Furthermore, the economics of an LCP are dependent on how well relevant investments are protected. This is especially relevant to application portability to other environments (Frank et al., 2021).

Low-code applications empower business professionals to swiftly develop custom software applications tailored to their specific needs. This not only lowers communication expenses but also decreases research and development costs, playing a crucial role in expediting the digital transformation of enterprises (Wang et al., 2023).

Startups using the popular low-code tool Shopify take a different growth path compared to traditional ones. Despite starting with fewer resources, they achieve similar success, as measured by successful exits. Although their profiles at exit might seem average, a closer look reveals a unique "lean success" profile. These Shopify-based startups need fewer resources to launch and grow but still generate comparable value for investors and employees (Dushnitsky & Stroube, 2021).

The adoption and maintenance costs of low-code platforms are significant considerations for organizations. Low-code platforms typically employ a subscription-based pricing model, which can be application-based, per-user, or per-component, allowing organizations to pay for actual usage rather than committing to a bundle of services, making it more cost-effective and scalable as the business expands (The Real Cost of Low-code Applications | Medium, n.d.) Additionally, the reduced need for human resources in low-code development contributes to cost savings, making it a more affordable option compared to traditional high-code development. The cost of low-code development services can vary

depending on the project scope and the chosen low-code platform (Quick Comparison between Low-code vs High Code: Which Wins? n.d.). For small businesses, the cost of low-code development can be anywhere between $50 to $200 per month, depending on the scope of the project. For enterprises, the cost can be higher, with some projects costing up to $100,000. However, low-code development is still considered more affordable than traditional high-code development. The reduced need for human resources and the pre-built code blocks in low-code development contribute to cost savings. The cost of maintaining and managing low-code applications can vary depending on the platform used and the level of customization required. The cost of low-code development can be further reduced by choosing the right low-code platform, with some platforms offering subscription plans and pay-as-you-go plans (How Much Does Low-code Development Cost?, n.d.).

Two articles (Käss et al., 2023; Käss et al., 2022) also explore the role of cost considerations in LCDP adoption. The literature on this subject lacks consistency, with some authors highlighting the potential for cost savings. Käss et al. (2022) also cite a Forrester study that states that LCDPs can accelerate development by 5-10 times. According to a study undertaken by 451 Research, low-code platform maintenance efforts have exhibited efficiency savings ranging from 50% to 90% when compared to modifications performed with traditional coding languages (Chaudhary & Margaria, 2021, p. 5). This efficiency is attributed to the modular design with a high degree of automation, requiring less manual effort and a lower need for highly skilled IT personnel. However, others argue that LCDPs may not necessarily reduce costs, especially with pricing models based on the number of users, potentially leading to higher costs at scale. Both articles emphasize the need for more detailed research to delineate the conditions under which cost reduction can be considered as a driver or inhibitor of LCDP adoption (Käss et al., 2022).

2.1.5. NEED OF TRAINING

In terms of training required to learn an LCDP, Zhuang et al. (2022) propose the first low-code federated learning (FL) platform, EasyFL, to enable users with various levels of expertise to experiment and prototype FL applications with little coding. EasyFL allows beginners to start experiencing FL with only three lines of code, even without prior knowledge of FL. Compared with other platforms (LEAF, TensorFlow Federated (TFF), and PySyft, as well as industrial-level frameworks such as federated AI technology enabler (FATE) and PaddleFL), EasyFL requires just three lines of code (at least 10 times lesser) to build a vanilla FL application and incurs lower training overhead. The article highlights that existing platforms are complex to use and require a deep understanding of FL, imposing high barriers to entry for beginners, limiting the productivity of researchers, and compromising deployment efficiency. In contrast, EasyFL which is an LCDP aims to be user-friendly, support efficient experimentation, and provide seamless and scalable deployment. Shopify as an LCDP provides a comprehensive set of business functions, making it easily accessible even for individuals without extensive training (Dushnitsky & Stroube, 2021).

2.1.6. AVAILABILITY

The availability of LCDPs implies their presence in the market, ease of access, user-friendliness, support, and compatibility, making them ready and accessible for individuals and organizations to create applications with reduced manual coding efforts.

The past decade witnessed a surge in the availability of low-code tools, where software-based solutions can be developed with limited or no need for writing code (Dushnitsky & Stroube, 2021). There are many LCDP providers available in the market.

The LCDP market exhibits diversity, encompassing vendors of various sizes, many of which specialize in specific application domains. Certain LCDP providers offer pre-configured layouts designed for specific routine business processes, although their prevalence remains limited, even as they enhance user-friendliness (Bies et al., 2022).

Frank et al. (2021) discusses various aspects of availability that can be used to evaluate the platforms in their study. Some of these vendors mentioned include the following categories and vendors:

1. **Basic Data Management**: **Quickbase** and **TrackVia** offer quick development for simple, data-centric applications but have limited collaboration and customization options.
2. **Workflow Management: Bonita Studio** focuses on complex workflow modeling for experienced developers, while **Creatio Studio** provides AI support and collaboration tools, suitable for both lay and professional developers.
3. **Extended GUI- and Data-centric Integrated Development Environment (IDE)s**: **Mendix Studio (Pro)** supports diverse applications with AI support, while **WaveMaker Studio** emphasizes source code-based development for professional developers.
4. **Multi-Use Business Platforms**: **Microsoft Power Apps** and **Appian** offer varied configurations catering to both lay and professional developers, with limitations in collaborative features and dependencies on ecosystem integration.
5. **Pega Platform**: Provides distinct workspaces for lay and professional developers, with a focus on collaborative development, although balancing user-friendliness and advanced features remains challenging.

2.2. SYSTEMATIC LITERATURE REVIEW ON LCDP ADOPTION

This literature review synthesizes findings from multiple studies to identify and analyze the key drivers and inhibitors affecting LCDP adoption. Each factor is discussed through insights drawn from the relevant literature.

2.2.1. SYSTEMATIC LITERATURE REVIEW METHODOLOGY

In order to comprehensively understand the adoption drivers of low-code platforms, a Systematic Literature Review (SLR) following the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) framework was conducted to identify key drivers and inhibitors of LCDP adoption. PRISMA was selected because it ensures a transparent, replicable, and structured process for literature identification, screening, and synthesis.

This review aims to address several key research questions, which are outlined in Table 1:

Table 1 - *Systematic Literature Review's Research Questions*

SLRQ1	What are the main drivers of low-code adoption?
SLRQ2	What are the major issues in low-code adoption?
SLRQ3	What is the current status of research low-code adoption?
SLRQ4	What are the advantages and disadvantages of low-code adoption?

To systematically answer these questions, established methodologies and databases were utilized to gather relevant literature. The following keywords and phrases in Table 2 were fundamental in the search strategy:

Table 2 - *Keywords used in the Systematic Literature Review*

Ke	**Adoption Models**	**Low-code**

	User Acceptance	Low-code
	User Adoption	Lowcode
	User Satisfaction	Low-code
		Low-code development
		Low-code platform
		4GL

The databases used for this review included Scopus, Web of Science, and ScienceDirect as in Table 3.

Table 3 - *Databases searched for the Systematic Literature Review*

Resource Database	**Resource URL**
Scopus	https://www.scopus.com/
Web of Science	https://www.webofknowledge.com/
Science Direct	https://www.sciencedirect.com/

Followingly, the search strings were carefully constructed to ensure comprehensive and relevant studies, as presented in Table 4:

Table 4 - *Search strings for each of the databases*

Scopus and Web of Science
("Low code" OR "low-code" OR "lowcode" OR "4GL") **AND** **("user adoption" OR "adoption" OR "User Acceptance" OR "User Satisfaction")** **AND** **(“factor*" OR "driver*" OR "motivation*")**
For ScienceDirect:

("Low code" OR "low-code" OR "lowcode" OR "4GL")

AND

("user adoption" OR "adoption" OR "User Acceptance" OR "User Satisfaction")

AND

("factor?" OR "driver?" OR "motivation?")

The inclusion and exclusion criteria ensured the relevance and quality of the selected articles. These criteria are shown as in Table 5.

Table 5 - *Inclusion & Exclusion criteria for the Systematic Literature Review*

Inclusion Criteria	**Exclusion Criteria**
Any scientific article showing evidence of adoption of low-code platforms	Papers focusing on Low-code but without focusing on AI techniques utilization
Paper must be a peered reviewed conference or journal paper written in English	Articles not in English and duplicate papers
Paper is published in a scholarly journal with at least quartile two classification	Articles from quartile three or quartile four journals
Paper is published between 2019 and the first two quartiles of 2024	Articles published before 2019
	Non-academic or non-scientific papers (e.g., websites, magazines reports, newspapers, consulting articles, books, citations)
	Papers with titles outside the scope of this work

2.2.1.1. SEARCH RESULTS

The initial search produced a total of 155 articles, with 141 articles from ScienceDirect, 6 articles from Web of Science, and 8 articles from Scopus. After removing duplicates, 110 unique articles remained. These articles underwent a screening process based on their titles and abstracts. From the title screening, 51 articles were retained for further evaluation. Following abstract screening narrowed this down to 19 articles deemed relevant

for further reading. Ultimately, 14 articles were read in full to assess their relevance to the research questions by their abstracts.

In addition, the search was updated to include articles published in 2024. This updated search involved using additional keywords related to low-code platforms, specifically "factor" 43 articles, "driver" 12 articles, and "motivation" 22 articles, resulting in a total of 77 articles from ScienceDirect. Combined with 2 articles from Scopus, this resulted in a total of 79 new articles. After compiling these results and removing duplicates, 45 unique articles remained. These 45 articles were then reviewed based on their titles, and 13 articles were identified as relevant to the research context. Following a detailed abstract review of these 13 articles, 5 articles were selected for a full-text review due to their potential resourcefulness for the literature review.

By integrating these articles from 2024, the review ensures a comprehensive and up-to-date analysis of low-code platform adoption which enriches the review with the latest insights and developments, providing a broader and more current understanding of the field. Below we can see all the articles with their respective details can be found in Table 6.

Table 6 - *Articles in full-text reading*

#	**Authors**	**Article Name**	**Contribution**	**Publication Type**
1	David, I., Aslam, K., Malavolta, I., & Lago, P.	Collaborative Model-Driven Software Engineering — A systematic survey of practices and needs in industry	Presents a systematic survey of practices and needs in collaborative model-driven software engineering in industry	Journal of Systems and Software
2	Dushnitsky, G., & Stroube, B. K.	Low-code entrepreneurship: Shopify and the alternative path to growth	The study focuses on the impact of low-code tool Shopify on e-commerce startups, highlighting differences in resource allocation, financial and human resources, and success	Journal of Business Venturing Insights

#	Authors	Article Name	Contribution	Publication Type
			metrics between Shopify-based and non-Shopify startups.	
3	Hoogsteen, D., & Borgman, H.	Empower the Workforce, Empower the Company? Citizen Development Adoption	This article explores the factors influencing organizational decisions to adopt citizen development and provides insights into successful adoption practices using an extended TOE framework.	Conference/ Industry Report
4	Kaess, S.	Association for Information Systems Low-code Development Platform Adoption: A Research Model	Proposes a research model combining social, technical, and environmental factors to explain LCDP adoption.	Research Article/ Conference Paper
5	Käss, S., Strahringer, S., & Westner, M.	Drivers and Inhibitors of Low-code Development Platform Adoption	The article provides a comprehensive literature review to identify the drivers and inhibitors of LCDP adoption. The authors structured their findings along the Diffusion of Innovation (DOI) framework, focusing on factors such as compatibility, complexity, relative advantage, trialability, and observability. They identified seven drivers and thirteen inhibitors for LCDP adoption through their literature review.	IEEE Conference Paper
6	Käss, S., Strahringer, S., & Westner, M.	Practitioners' Perceptions on the Adoption of Low-code Development Platforms	It empirically validates drivers and inhibitors for LCDP adoption, adds six new drivers and six new inhibitors to the body of knowledge, and analyzes the importance of these factors.	IEEE Access
7	Martins, J., Branco, F., & Mamede, H.	Combining low-code development with ChatGPT to novel no-code approaches: A focus-group study	Proposes a development model that integrates ChatGPT with low-code platforms to streamline software development	Intelligent Systems with Applications

#	Authors	Article Name	Contribution	Publication Type
			processes and address the shortage of skilled developers. The model involves five stages: data modeling, business logic implementation, integrating external services, testing and deployment, and user training and feedback collection.	
8	Mosch, P., Majocco, P., & Obermaier, R.	Contrasting value creation strategies of industrial-IoT-platforms – a multiple case study	Provides insights into the strategic measures and challenges faced by different types of IIoTPs, emphasizing the importance of data security, open platform structures, and community management in the context of industrial IoT platforms.	International Journal of Production Economics
9	Pinho, D., Aguiar, A., & Amaral, V.	What about the usability in low-code platforms? A systematic literature review	Identifies common characteristics of low-code platforms, highlighting the growing awareness of usability among users, listing factors influencing user experience, referencing feasibility studies, and calling for increased research on usability to advance the field.	Journal of Computer Languages
10	Shamsujjoha, M. et al.	Developing Mobile Applications Via Model Driven Development: A Systematic Literature Review	Contributes a systematic review of Model-Driven Development techniques for mobile app development, identifying research gaps, strengths, and limitations in the field.	Journal of Information and Software Technology
11	Solaimani, S., & Swaak, L.	Critical Success Factors in a multi-stage adoption of Artificial Intelligence: A Necessary Condition Analysis	Identifies critical success factors and assesses their criticality across various stages of AI adoption using Necessary Condition Analysis (NCA).	Journal of Engineering and Technology Management

#	Authors	Article Name	Contribution	Publication Type
12	Sundberg, L., & Holmstro ¨m, J. (2024)	Democratizing artificial intelligence: How no-code AI can leverage machine learning operations	Discusses challenges and benefits of MLOps, emphasizes the importance of collaboration between technical and domain experts, and outlines managerial recommendations for MLOps.	Journal Article
13	Rafiq, U., Cenacchi, F., & Wang, X	Understanding Low-code or No-code Adoption in Software Startups: Preliminary Results from a Comparative Case Study	Addressing the gap in empirical studies on low-code/no-code adoption, particularly in startups, providing preliminary insights, methodological approaches, practical implications, and future research directions.	ResearchGate
14	Wald, R., Piotrowski, J. T., Araujo, T., & van Oosten, J. M. F.	Virtual assistants in the family home. Understanding parents' motivations to use virtual assistants with their Child(dren)	Investigates the motivations and intentions of parents using virtual assistants (VAs) with their children, identifies family typologies, and key motivational factors like enjoyment.	Computers in Human Behavior Peer-Reviewed Journal Article
15	Monteiro T., Abrantes S., Ratinho, M.	A Solution for Submitting Expenses	Development of a mobile application for submitting expense reports using the Power Apps platform, highlighting low-code development and process automation.	Procedia Computer Science
16	Mosquera D., Pastor O., Spielberger J.	LEMON: A Tool for Enhancing Software Requirements Communication through Requirements Pattern-based Modelling Assistance	Introduction of LEMON, a tool to enhance software requirements communication using requirements patterns, integrated with low-code/no-code platforms.	Joint Proceedings of REFSQ-2024 Workshops
17	Holmström J., Carroll N.	How Organizations Can Innovate with Generative AI	Exploration of generative AI integration with low-code platforms to enhance innovation, providing a	Journal of Business Innovation

#	Authors	Article Name	Contribution	Publication Type
			framework for automation and augmentation.	
18	Abdul Razak S. F., Yow P. E., Yussoff F. I., Bukar U. A., Yogarayan S.	Enhancing Business Efficiency through Low-Code/No-Code Technology Adoption	Examination of user behavioral intentions towards adopting low-code/no-code technology platforms using the Unified Theory of Acceptance and Use of Technology (UTAUT) model, identifying key factors influencing adoption.	Journal of Enterprise Information Management
19	Mosquera D., Ruiz M., Pastor O., Spielberger J.	Understanding the landscape of software modelling assistants for MDSE tools A systematic mapping	Systematic mapping of software modelling assistants for MDSE and low-code/no-code tools, identifying strategies, goals, and limitations.	Systematic mapping of software modelling assistants for Model MDSE and low-code/no-code tools, identifying strategies, goals, and limitations.

2.2.2. DRIVERS AFFECTING THE ADOPTION OF LCDPS

Improved Efficiency and Cost-Effectiveness: LCDPs are praised for their potential to increase software development efficiency by enabling faster application development, reducing time to market, and lowering costs, as stated by Käss et al. (2022, 2023) and highlight that LCDPs enhance productivity through component reuse and shorter development cycles, leading to significant cost savings. Fatimah et al. (2024), using the UTAUT model, further validate that perceived performance improvements (Performance Expectation) strongly influence adoption, with users recognizing LCDPs as a way to streamline processes and reduce reliance on traditional IT development. Monteiro et al.

(2024) and Martins et al. (2023) confirm these findings, noting the economic benefits of reduced development and maintenance costs as well as reduced demand for specialized developers.

Empowerment of Citizen Developers: One of the defining strengths of LCDPs is their ability to empower non-technical employees to create and manage applications. Käss et al. (2022, 2023) and Hoogsteen & Borgman (2022) discuss how LCDPs enable domain experts without extensive coding knowledge to build applications, which enhances collaboration and reduces the dependency on IT departments. This democratization of development capabilities addresses the shortage of qualified developers by allowing non-technical staff to actively contribute to software initiatives.

Enhanced Collaboration and Business Process Efficiency: LCDPs improve cross-functional collaboration by offering user-friendly interfaces that allow business units and IT departments to work more closely. According to Käss et al. (2022, 2023) and Mosquera et al. (2024), these platforms align business and IT teams, streamlining development and improving process efficiency. The integration of pre-built templates and standardized practices further supports alignment and effective communication between technical and non-technical users (Mosquera et al., 2024).

Seamless Integration and Flexibility: LCDPs are valued for their integration capabilities, ensuring compatibility with existing IT infrastructures and enabling organizations to leverage their current systems. Käss et al. (2022, 2023) emphasize that LCDPs support integration with existing systems, which is crucial for smooth adoption. Similarly, Monteiro et al. (2024) note that LCDPs offer flexibility, adaptability, and

extensibility, allowing organizations to rapidly adjust to evolving business requirements and market demands.

Innovation and Speed to Market: Many organizations view LCDPs as a means to quickly respond to market changes, as they allow for rapid prototyping and deployment. Holmström & Carroll (2024) describe the synergy between generative AI and LCDPs in accelerating development cycles and enhancing innovation by automating repetitive tasks. This speed, coupled with the reduced complexity of development, provides organizations with a competitive edge by enabling faster delivery of digital solutions.

Reduction of Shadow IT and Improved Control: LCDPs also reduce the prevalence of unauthorized or "shadow IT" development by providing a sanctioned platform for business users to create applications. Käss et al. (2023) and Monteiro et al. (2024) discuss how providing an official, managed development environment minimizes security risks associated with unsanctioned IT solutions, enhancing organizational control over software deployment.

2.2.3. INHIBITORS AFFECTING THE ADOPTION OF LCDPS

Limited Functionality and Customization Challenges: Although LCDPs offer rapid development, they may not support the customization and functionality needed for complex applications. Käss et al. (2022, 2023) and Dushnitsky & Stroube (2021b) point out that LCDPs often lack the depth required for highly specialized applications, making them less suitable for projects with advanced technical requirements. Mosquera et al. (2024) emphasize that limited customization options can be a barrier for organizations needing tailored solutions.

Security and Compliance Concerns: Security remains a prominent concern with LCDPs, as applications built by non-professional developers may contain vulnerabilities. Käss et al. (2022) and Fatimah et al. (2024) highlight that perceived risk, particularly in terms of security and compliance, deters organizations from adopting LCDPs. Mosquera et al. (2024) and Martins et al. (2023) also emphasize the need for robust security and compliance protocols to mitigate risks, especially for applications that handle sensitive data.

Integration Challenges and Technical Debt: Integrating LCDPs with legacy systems can present significant technical challenges as noted by Käss et al. (2022, 2023) and Monteiro et al. (2024) also note that organizations often struggle to achieve seamless integration with older systems, which may require additional resources and incur technical debt over time. In some cases, the added complexity may offset the advantages of adopting LCDPs, especially for large enterprises with extensive legacy infrastructures.

Vendor Lock-In and Licensing Complexity: Vendor dependency is a significant inhibitor, as organizations may feel "locked-in" to a particular LCDP provider according to Käss et al. (2022) and Dushnitsky & Stroube (2021) discuss the challenges of vendor lock-in, where organizations face limited flexibility if they become reliant on a single provider. Furthermore, complex licensing models can create financial uncertainties and discourage smaller organizations from adopting LCDPs (Monteiro et al., 2024; Martins et al., 2023).

Resistance from IT Departments and Traditional Developers: LCDP adoption often encounters resistance from traditional developers and IT departments as highlighted by Käss et al. (2022, 2023) and Martins et al. (2023) note that concerns over job security, loss of control, and skepticism regarding the capabilities of LCDPs can hinder adoption. Traditional

developers may also view the low-code approach as a threat to their specialized skills, leading to reluctance in supporting LCDP initiatives.

High Learning Curve and Lack of Awareness or Training: Despite being marketed as easy-to-use platforms, LCDPs can still pose a learning curve, particularly for organizations that are new to the concept. Mosquera et al. (2024) and Käss et al. (2023) highlight that inadequate training and lack of awareness may hinder the adoption, as employees might find it challenging to adapt to the visual development interfaces and functionalities unique to LCDPs. Without sufficient training, organizations may struggle to maximize the potential benefits of these platforms.

Performance and Scalability Issues: Some LCDPs face scalability and performance limitations, especially when used for mission-critical or large-scale applications as pointed out by Käss et al. (2022, 2023) and Mosquera et al. (2024) discuss concerns around the performance of applications built on LCDPs, as they may not meet the standards required for enterprise-level or high-demand applications. These limitations can reduce confidence in LCDPs for projects that require robust performance.

To conclude, the adoption of LCDPs is influenced by a large and complex set of drivers and inhibitors. However, these findings above are the most commonly mentioned drivers and inhibitors during the systematic literature review scanning.

Drivers like increased efficiency, empowerment of non-technical users, enhanced collaboration, and faster time to market make LCDPs an attractive option for organizations aiming to reduce costs and accelerate innovation. Nonetheless, significant inhibitors, including security concerns, integration challenges, limited customization, and potential vendor lock-in, temper the motivation for LCDP adoption. Empirical validations from studies

such as Fatimah et al. (2024) using the UTAUT model underscore the influence of factors like performance expectations and perceived risks on adoption.

Understanding these drivers and inhibitors is critical for organizations considering LCDP implementation. Addressing inhibitors through training, clear governance frameworks, and robust integration strategies can help organizations harness the full potential of LCDPs, while also managing risks associated with security, customization, and scalability. The systematic literature review, thus, provides a comprehensive framework to guide organizations in making informed decisions about adopting LCDPs.

The findings align with prior literature, reaffirming that user-friendly interfaces drive adoption, while concerns about performance and scalability deter it (Käss et al., 2022). Notably, this study highlights the significant role of job roles in influencing adoption, adding nuance to the existing body of work (Fatimah et al., 2024). Future research should explore how organizational strategies can mitigate these inhibitors to maximize LCDP potential.

2.2.4. Current Status of Research on Low-code Adoption

Research on LCDP adoption reflects a rapidly evolving field, driven by LCDPs' potential to address key software development challenges such as skill shortages, time-to-market pressures, and the need for agile, cost-effective solutions. However, significant gaps and limitations in the current research landscape call for a more robust theoretical foundation and comprehensive empirical studies to guide adoption strategies effectively. This section synthesizes the state of research on LCDP adoption, summarizing key themes, advantages, and disadvantages identified across recent studies.

The current state of LCDP research indicates strong interest in the platforms' potential to address software development challenges such as cost, speed, accessibility, and workforce shortages. Key theoretical contributions have emerged, notably the integration of the Technology-Organization-Environment Framework (TOE) and Socio-Technical Systems (STS) frameworks to explore socio-technical factors affecting adoption. Yet, the literature reveals significant gaps, particularly around the complex dynamics of organizational, technical, and social aspects of LCDP use, underscoring the need for more robust empirical research.

Despite substantial advantages, including speed, efficiency, and democratization of software development, the adoption of LCDPs also comes with challenges such as vendor lock-in, limited customization, and governance and security concerns. Addressing these trade-offs requires a strategic approach, with effective governance frameworks, training, and technical support to mitigate potential risks. As AI integration with LCDPs continues to evolve, understanding these platforms' complexities and opportunities will be critical for organizations looking to leverage them for digital transformation and innovation.

2.2.4.1. KEY TRENDS AND THEORETICAL FOUNDATIONS IN THE LITERATURE

Lack of Strong Theoretical Models: Many studies highlight a need for a robust theoretical basis for researching LCDP adoption. Dushnitsky and Stroube (2021) note that existing research often lacks theoretical rigor, proposing a model that integrates the TOE framework with STS theory to capture the multi-dimensional factors (technical, social, and environmental) that influence adoption. This model provides a foundation for systematically studying the impact of both technological and organizational aspects on LCDP uptake, highlighting a holistic approach that can deepen our understanding of adoption dynamics.

Empirical Gaps and Emerging Nature of the Field: The body of research on LCDP adoption remains emerging, with a significant increase in publications noted only after 2019, as pointed out by Pinho et al. (2023). Most of this research consists of conference papers, underscoring its emerging nature and novelty. Käss et al.,2022 emphasize that existing studies predominantly focus on technological aspects, with limited exploration of organizational and social factors, suggesting a need for a more comprehensive approach to studying the complex dynamics of LCDP adoption.

Influence on Entrepreneurship and Scaling: LCDPs' influence on entrepreneurship and business scaling is highlighted in work by Shamsujjoha et al. (2021), who discuss the impact of platforms like Shopify on resource requirements for startups. However, they identify a significant gap in understanding how LCDPs affect scalability and resource allocation, calling for research that examines how low-code tools facilitate or hinder business growth.

Citizen Development and Workforce Augmentation: The rise of "citizen development" as a trend in LCDP adoption is noted by Hoogsteen & Borgman (2022), who study how LCDPs empower end-users without programming backgrounds to participate in application development. Through an extended TOE framework, they discuss factors influencing organizational decisions, such as risk perceptions, management support, and IT governance, underscoring LCDPs' role in democratizing software development amid a global shortage of skilled developers (Martins et al., 2023).

Integrating Low-Code with AI for Enhanced Functionality: LCDPs increasingly intersect with AI, as highlighted by Solaimani & Swaak (2023), who emphasize the significance of performance expectancy and organizational compatibility for AI adoption,

relevant factors for LCDPs as well. Research by Shadrack & Muiruri (2023) and Drenik (2023) also points to the potential of AI-driven LCDPs in democratizing development and accelerating innovation, although they acknowledge challenges in maintaining quality, security, and user training in AI-powered environments.

Insights from Industry and Market Trends: The role of industry insights, like the Gartner Magic Quadrant, in shaping LCDP selection and adoption decisions is underscored by Monteiro et al. (2024). Such assessments provide organizations with benchmarks on vendor capabilities, vision, and strategy, reflecting the increasing demand for guidance in navigating the complex LCDP landscape.

2.2.5. Advantages and Disadvantages of Low-Code Adoption

2.2.5.1. Advantages

Speed and Efficiency: LCDPs offer significant time savings, enabling rapid prototyping and accelerating time-to-market (Pinho et al., 2023; Shamsujjoha et al., 2021). Studies by Käss et al. (2022) and Martins et al. (2023) further emphasize efficiency gains, as LCDPs reduce the need for extensive coding and support agile development methodologies.

Cost Savings: Reduced development and maintenance costs are a primary advantage of LCDPs, attributed to lower demand for specialized resources and streamlined workflows (Käss et al., 2022). By minimizing repetitive tasks, LCDPs enable cost efficiencies during development, testing, and maintenance phases, making them financially attractive for organizations (Martins et al., 2023).

Accessibility and Empowerment of Citizen Developers: LCDPs democratize application development by empowering non-technical users—often referred to as "citizen

developers"—to build applications, thereby expanding the pool of contributors to software projects (Käss et al., 2022; Martins et al., 2023). This accessibility allows organizations to leverage domain expertise without requiring extensive coding knowledge, addressing the global shortage of skilled developers.

Flexibility and Scalability: LCDPs allow for easy updates and modifications, ensuring applications can scale in response to evolving business needs (Käss et al., 2022). Flexibility in adapting to new requirements enhances LCDPs' appeal for dynamic and fast-paced business environments (Monteiro et al., 2024).

Enhanced Collaboration: LCDPs promote collaboration between IT and business units by simplifying development processes, allowing iterative development and alignment with agile methodologies (Käss et al., 2022; Martins et al., 2023). Simplified workflows facilitate communication and shared ownership of projects across teams.

Integration with AI for Improved Performance: LCDPs increasingly integrate AI capabilities, such as automation and predictive analytics, which can enhance productivity and support digital transformation (Solaimani et al., 2023; Microsoft, 2023). The synergy between AI and LCDPs provides organizations with innovative tools to meet performance expectations and improve usability (Lee & Shin, 2020; Jöhnk et al., 2021).

2.2.5.2. DISADVANTAGES

Vendor Lock-In and Licensing Complexity: A reliance on proprietary LCDPs can create vendor lock-in, limiting flexibility and complicating interoperability (Käss et al., 2022). This dependency, coupled with complex licensing models, can restrict organizational agility and hinder switching between platforms, a concern echoed in studies by Sundberg & Holmström (2023).

Complexity and Customization Limitations: LCDPs may struggle with developing complex applications requiring high customization, which limits their effectiveness in projects needing unique, tailored solutions (Käss et al., 2022; Martins et al., 2023). The drag-and-drop functionality, though user-friendly, often restricts customization, potentially leading to compromised user experience and functionality.

Security and Compliance Challenges: The integration of AI in LCDPs introduces data security and privacy concerns, particularly when applications are built by non-professional developers without security training (Solaimani et al., 2023). Security risks are a prevalent barrier, as inexperienced users may overlook best practices, exposing organizations to vulnerabilities (Käss et al., 2022).

Interoperability and Scalability Issues: Integrating LCDPs with existing systems and achieving scalability for complex applications remain significant challenges. Studies highlight difficulties in seamlessly connecting low-code solutions with legacy systems, which can affect the platform's scalability and long-term viability (Sundberg & Holmström, 2023; Käss et al., 2022).

Lack of Formal Definition and Governance Concerns: The lack of a standardized definition of low-code development hinders the field's growth, with many researchers unconsciously incorporating usability considerations (Pinho et al., 2023). Furthermore, maintaining governance over application development on LCDPs can present issues, particularly in areas of security, compliance, and oversight (Hoogsteen & Borgman, 2022; Käss et al., 2023).

Functional Limitations and Quality Concerns: LCDPs may not meet the quality and functional standards of applications developed by experienced developers. For instance,

Monteiro et al. (2024) and Käss et al. (2022) note that complex applications may be compromised by LCDP limitations in functionality, scalability, and customization.

3. METHODOLOGY

3.1. INTRODUCTION TO METHODOLOGY

This section outlines the methodology employed in this research to investigate the factors influencing user adoption among different user groups LCDPs. It includes research design, data collection methods, data analysis procedures, and considerations for ensuring validity and reliability.

3.2. RESEARCH DESIGN

This research adopts a quantitative approach to examine the factors affecting the acceptance and adoption of LCDPs among different user groups. A survey-based methodology was chosen for its ability to gather comprehensive data from a large sample.

This study employs Partial Least Squares Structural Equation Modeling (PLS-SEM) and bootstrapping techniques using SmartPLS to analyze the survey data. PLS-SEM was chosen due to its strength in handling complex models with multiple constructs and indicators, particularly when the sample size is relatively small, and the data distribution is non-normal. This approach allowed for the simultaneous evaluation of measurement models (validating the reliability and validity of constructs) and structural models (testing hypothesized relationships between constructs).

Bootstrapping, a non-parametric resampling technique, was applied to estimate the precision of the PLS-SEM results. It provided confidence intervals for path coefficients and enabled significance testing of the relationships in the model. These methods were critical for this study as they ensured the reliability and validity of findings while addressing potential limitations associated with sample size and data distribution.

4. CONCEPTUAL MODEL AND HYPOTHESES

The Technology Acceptance Model (TAM) was chosen as the primary framework because of its proven effectiveness in studying technology adoption and user behavior. The TAM was first introduced by (Davis, 1989) as a framework to understand and predict technology adoption based on two key constructs: Perceived Usefulness (PU) and Perceived Ease of Use (PEOU). According to TAM, these constructs influence a user's Attitude Toward Use (ATU), which, in turn, impacts Behavioral Intention to Use (BI) and, ultimately, Actual Use (AU). TAM has been widely adopted due to its simplicity and strong predictive power in understanding technology acceptance.

Subsequent studies identified that Attitude Toward Use (ATU), while valuable in some contexts, contributed minimally to predicting BI in organizational settings. (Venkatesh & Davis, 2000) simplified TAM by excluding ATU and introduced TAM2, which focuses on the direct effects of PU and PEOU on BI. TAM2 also incorporates external variables such as subjective norms and voluntariness to enhance its applicability in organizational contexts.

A customized, simplified version of TAM was chosen for this study, based on the streamlined structure proposed in TAM2 by (Venkatesh & Davis, 2000), TAM2 has proven effective in predicting technology adoption and usage across various domains and in assessing technology acceptance at an individual level. Unlike the UTAUT model, which is more suited for mandatory, organizational contexts, TAM2 offers a framework for examining individual adoption behaviors, making it particularly appropriate for studying LCDP adoption. The relationships among these constructs can be summarized as in the diagram in Figure 1:

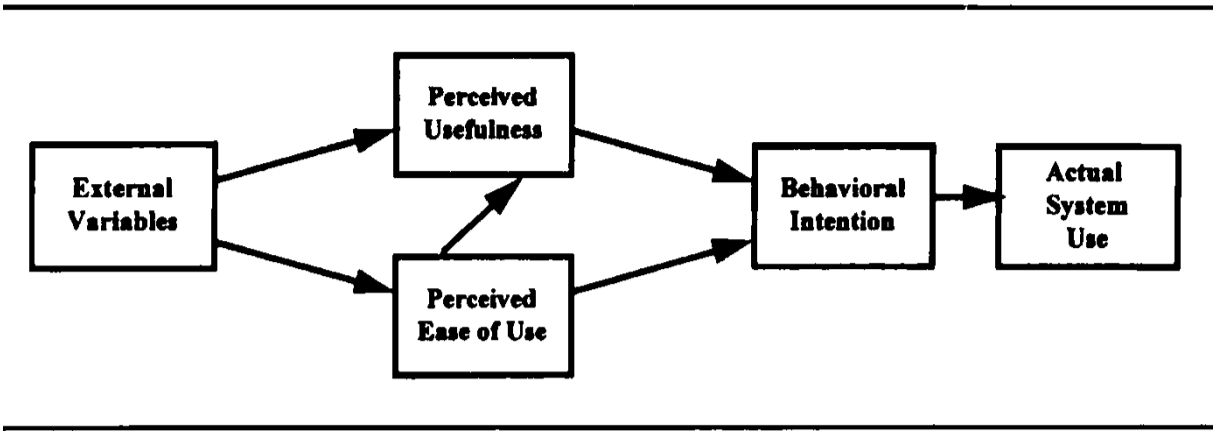

Figure 1 - *Technology Acceptance Model (Davis, 1989)*

This model is advantageous due to its simplicity, empirical support, and adaptability to various technological contexts, making it particularly suitable for studying the adoption of LCDPs.

4.1. Model Structure

Additionally, tailoring TAM to the specific context of LCDP adoption, age, job role (technical, non-technical, executive), and company size were included as control variables. By integrating the simplified TAM with the specific drivers and inhibitors identified in the literature review, this conceptual model aims to provide a comprehensive yet focused framework for understanding the factors influencing the adoption of LCDPs. The conceptual model shapes around the constructs of TAM in this study as below:

Independent Variables (IV): PU & PEOU
Mediating Variables: BI
Dependent Variable (DV): AU
Control Variables: External Variables (e.g., Demographic Factors, Job Role, Company Size)

As the next step, the drivers and inhibitors found in the SLR is mapped into TAM constructs shown as the Table 7 below:

Table 7 - *SLR items map into TAM constructs*

Construct	Items	Measures	Source	Survey Question
PU				
PU1	Improved Performance and Efficiency	Enhances software development processes, leading to cost savings, increased productivity, component reuse, and reduced development time.	Käss et al. (2022), Käss et al. (2023), Monteiro et al. (2024), Mosquera, Ruiz, et al. (2024), Sundberg & Holmström (2023), Fatimah et al. (2024)	Using LCDPs enhances my job performance.
PU2	Cost Reduction & Resource Saving	Lowers the cost of development and maintenance compared to traditional methods.	Käss et al. (2022), Käss et al. (2023), Monteiro et al. (2024), Mosquera, Ruiz, et al. (2024), Fatimah et al. (2024), Dushnitsky & Stroube (2021b)	LCDPs provide significant cost savings.
PU3	Flexibility and Customizability	Supports quick adaptations and modifications, making it easier to respond to changing business requirements and market conditions.	Monteiro et al. (2024), Mosquera, Pastor, et al. (2024), Mosquera, Ruiz, et al. (2024), Holmström & Carroll (2024)	I find LCDPs flexible to interact with.
PU4	Compatibility and Integration with Existing Systems	Ensures compatibility and seamless integration with the organization's current IT infrastructure and software.	Käss et al. (2022), Käss et al. (2023), Monteiro et al. (2024), Mosquera, Pastor, et al. (2024), Mosquera, Ruiz, et al. (2024)	LCDPs integrate easily with existing systems.

PU5	Improved Business Agility and Responsiveness	Enhances the ability to swiftly develop and deploy applications in response to changing market conditions.	Käss et al. (2022), Käss et al. (2023), Monteiro et al. (2024)	I am confident that I can customize applications and use advanced features (e.g., workflow automation, custom scripting) of the LCDP effectively.
PEOU				
PEOU1	User-Friendly Interfaces and Reduced Complexity	User-friendly interface allows for easy adoption and application development by non-technical users.	Käss et al. (2022), Fatimah et al. (2024), Monteiro et al. (2024), Mosquera, Pastor, et al. (2024), Sundberg & Holmström (2023)	The user-friendly interface of LCDPs makes it easy to use.
PEOU2	Empowerment and Reduced Technical Barriers of Citizen Developers	Enables non-technical users to develop applications, integrating domain knowledge directly, and reducing dependency on IT departments.	Käss et al. (2022), Käss et al. (2023), Pinho et al. (2023b), Martins et al. (2023), Sundberg & Holmström (2023)	I am confident that I can independently find and use online resources or documentation to improve my skills with the LCDP.
PEOU3	Faster Development and Iteration	Increases the speed of application development, resulting in faster delivery and reduced time to market.	Käss et al. (2022), Käss et al. (2023), Fatimah et al. (2024), Monteiro et al. (2024), Sundberg & Holmström (2023)	I am confident that I can use the LCDP efficiently to complete tasks and improve my productivity.
PEOU4	Scalability and Reduced Infrastructure Concerns	Concerns about the performance and scalability of applications developed using LCDPs may deter adoption, especially for	Käss et al. (2022), Monteiro et al. (2024), Sundberg & Holmström (2023)	Using LCDPs requires less effort compared to traditional development methods.

		mission-critical systems.		
PEOU5	Guidance and Support, Learning Curve	Provides guidance and support, with a manageable learning curve.	Käss et al. (2022), Fatimah et al. (2024), Monteiro et al. (2024), Mosquera, Pastor, et al. (2024)	Learning to operate LCDPs is easy for me.
Behavioral Intention to Use				
BI	The degree to which a person has formulated conscious plans to use or not use the technology.	Influenced by PU and PEOU	(Davis, 1989; Venkatesh & Davis, 1996, 2000)	• I intend to use LCDPs frequently in the future. • I will recommend LCDPs to my colleagues. • I am likely to increase my use of LCDPs in my work.
Actual System Use				
AU1	Frequency of Use		(Davis, 1989)	How frequently do you use LCDPs?
AU2	Duration of Use		(Davis, 1989)	How many hours per week do you use LCDPs?
AU3	Intensity of Use		(Venkatesh & Davis, 2000)	How essential are LCDPs to your work tasks?

4.2. BUILDING THE CONCEPTUAL MODEL

To build the conceptual model on the factors affecting the adoption of LCDPs among different user profiles, firstly, the core hypotheses below are generated. This approach will help us understand how PU and PEOU influence users' attitudes ATU towards LCDPs, which in turn lead to users' BI and AU.

The proposed hypotheses will guide empirical testing to validate the relationships between these factors and their impact on user acceptance and usage behavior and fill the gap of empirical studies in this area in literature.

Table 8 - *Hypothesis Generated*

Hypothesis	**Description**
H1	PEOU positively influences the PU of LCDPs.
H2	PEOU positively influences the BI to use LCDPs.
H3	PU positively influences the BI to use LCDPs.
H4	BI positively influences the AU of LCDPs.

In order to better explain the entire conceptual model based on the hypothesis in Table 8, the diagram below is created to present the factors in question and their relations with each construct and with each other in Figure 2.

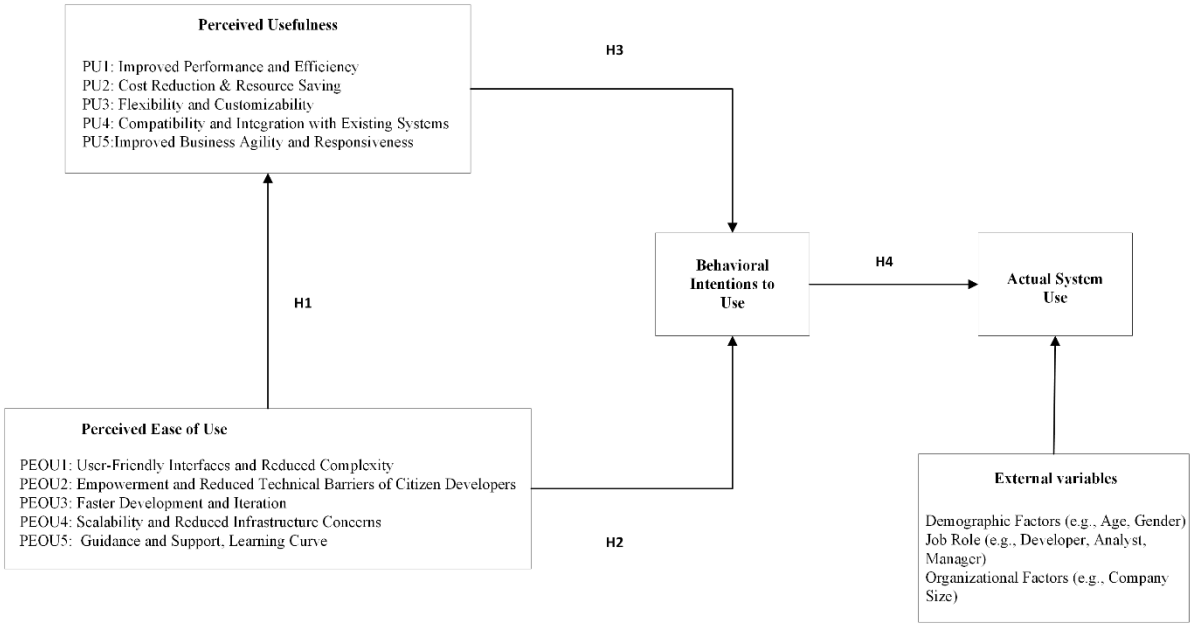

Figure 2 - *Conceptual Framework for LCDP Adoption*

Note. This diagram illustrates the hypothesized relationships between PU, behavioral BI, and AU in the adoption of LCDPs. The PU and PEOU constructs are adapted from the TAM (Davis, 1989).

The diagram effectively explains to us the relationships among the constructs and how the items selected from the SLR were assigned to the construct together with hypotheses generated based on the relationships.

4.3. Data Collection

This section outlines the process used to collect data for analyzing the factors influencing the adoption of LCDPs within the framework of TAM. It describes the methodology employed, the design and the structure of the survey with the encoding of its data for analysis. By ensuring the inclusion of only relevant participants with prior experience using LCDPs, the study increases the reliability and relevance of the collected data. The subsequent subsections provide a detailed breakdown of the survey methodology, structure, and the procedures for preparing the data for statistical modeling and analysis.

4.3.1. DATA COLLECTION METHODOLOGY

This study utilized a quantitative research design to investigate factors influencing user acceptance of LCDPs within the TAM framework. Data were collected using a structured online survey, which comprised multiple sections designed to measure key TAM constructs as well as demographic and organizational control variables, such as **Age**, **Job Role**, and **Company Size** utilizing a survey-based methodology to collect data on the factors influencing LCDP adoption.

A total of 217 participants were initially recruited, with 216 consenting to participate in the study. Among them, 134 respondents indicated prior engagement with LCDPs, while 11 participants did not fully completed the survey, yielding a drop out rate of approximately 5%. The survey included demographic questions to capture participants' age, job roles, and company sizes, enabling a detailed analysis of adoption factors across different user profiles. Incomplete responses were excluded from the final analysis to ensure data reliability.

4.3.2. SURVEY DESIGN AND STRUCTURE

The survey included screening questions to ensure respondents had prior interaction with LCDPs, followed by sections targeting the main TAM constructs and control variables. Questions were primarily close-ended and based on a Likert scale to facilitate quantitative analysis. The survey sections were structured as follows:

1. **Screening and Usage of LCDPs**: Respondents were first asked if they had experience with LCDPs. If they answered "No," the survey concluded, ensuring that only participants with relevant experience continued.
2. **Demographic Information**: Questions regarding **Age**, **Gender**, and **Job Role** helped categorize respondents. Age was categorized into ranges (e.g., Under 25, 25-34, etc.),

while job roles included options for technical, analytical, and managerial positions to capture a diverse professional background.

3. **PEOU**: This section included items such as “I find it easy to use LCDPs” and “LCDPs integrate easily with existing systems,” capturing aspects of usability, complexity reduction, and flexibility.

4. **PU**: Questions like “Using LCDPs enhances my job performance” and “LCDPs provide significant cost savings” measured the perceived benefits of using LCDPs in the workplace.

5. **BI**: This construct was assessed with items like “I intend to use LCDPs frequently in the future” and “I am likely to increase my use of LCDPs in my work.”

6. **AU**: The frequency of LCDP usage was captured through questions about usage frequency, hours spent per week, and the essentiality of LCDPs to their work tasks.

7. **Control Variables**: Additional questions assessed company size, job roles and age demographics to capture organizational context and potential influences on usage patterns.

4.3.3. DATA ENCODING PROCESS

To facilitate analysis, survey responses were encoded according to structured guidelines (see Appendix for the full encoding schema). Each variable and response option were assigned a numerical code to ensure consistency and ease of import into SmartPLS, which was used for PLS-SEM analysis due to several advantages:

- **Exploratory Nature**: PLS-SEM is ideal for studies exploring new research models or under-researched areas, such as LCDP adoption across different user groups.

- **Small-to-Medium Sample Suitability**: Unlike covariance-based SEM, PLS-SEM is well-suited for analyzing datasets with moderate sample sizes.
- **Ability to Handle Complex Models**: The research model includes multiple constructs and relationships, making PLS-SEM a suitable choice for hypothesis testing.
- **Predictive Power**: Since the study aims to identify practical strategies for increasing LCDP adoption, PLS-SEM provides valuable predictive insights.

Key encodings in the data preparation process included:

- **Age**: Encoded into categories (e.g., Under 25 = 1, 25-34 = 2, etc.).
- **Job Role**: Technical roles were coded as 1, non-technical as 0.
- **Agreement Levels on Likert Scale**: Responses were encoded from 1 (Strongly Disagree) to 5 (Strongly Agree) for TAM constructs.
- **Frequency of Use and Relevance**: Frequency was categorized from 1 (Never) to 5 (Always), while relevance ranged from 1 (Not relevant) to 5 (Very relevant).

This encoding process streamlined data preparation, enabling efficient statistical analysis in SmartPLS to validate the TAM constructs and explore the hypothesized relationships among variables.

5. ANALYSIS RESULTS AND DISCUSSION

While other external variables were included in this study, the survey data indicated that they did not significantly influence LCDP adoption in the sample studied. This highlights that Job Role effectively captures meaningful variations in user behavior, emphasizing the importance of job-specific requirements over general demographic or organizational factors for our sample.

The TAM framework, which reinforces this analysis, primarily focuses on two key predictors: PEOU and PU, which influence BI and ultimately AU of technology. While TAM effectively captures the core determinants of user acceptance, incorporating Job Role allows us to contextualize these relationships based on users' roles. This distinction is particularly relevant in the context of LCDPs, where technical users may have different usability and functionality expectations compared to non-technical users, whose needs might focus more on accessibility and simplicity.

Future studies could explore a more diverse sample or integrate additional qualitative methods to better understand why some external variables, such as demographics and organizational factors, might lack significance in influencing adoption.

Job Role was selected as the sole control variable to account for variations in technology acceptance across different user groups within the organization, specifically distinguishing between technical (developers, engineers) and non-technical roles (business analysts, managers) because this distinction is critical for understanding user acceptance and adoption of LCDPs, as these two groups typically have different levels of technical expertise, experience, and expectations regarding technology. By including Job Role as a sole control variable, we can isolate the impact of technical roles on acceptance and adoption without

introducing complexity from additional demographic, personal, or organizational factors. The diagram below demonstrates a representation of the relationships between constructs in this study.

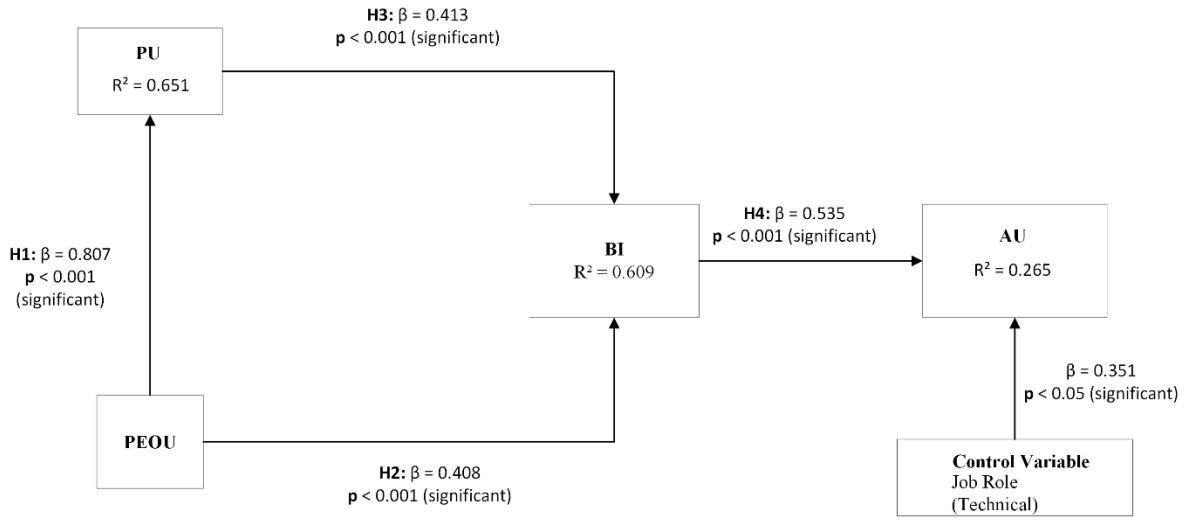

Figure 3 - *Estimated Model*

In the Figure 3, it highlights the significant positive effects of PEOU on both PU and BI, as well as the strong influence of PU on BI and BI on AU. The inclusion of R^2 values demonstrates the explanatory power of the model, with PU ($R^2 = 0.651$), BI ($R^2 = 0.609$), and AU ($R^2 = 0.265$). The control variable, Job Role (Technical), shows a moderate but significant impact on AU, adding depth to the analysis. Overall, the diagram effectively illustrates significant relationships, and the variance explained within the framework.

5.1. MODELS EVALUATION

The results were summed and organized in Table 9 and Table 10 for better readability for both Measurement Model and Structural Model, for both with and without control variable.

Table 9 - *Results of the Measurement Model and Structural without any Control Variable*

Measurement Model without Control Variable				
Metric	**Threshold**	**Construct**	**Analysis Result**	**Interpretation**
Cronbach's Alpha	> 0.70	AU	0.891	Good internal consistency.
		BI	0.914	Excellent internal consistency.
		PEOU	0.753	Acceptable internal consistency.
		PU	0.795	Acceptable internal consistency.
Composite Reliability (CR)	> 0.70	AU	0.932	Strong reliability.
		BI	0.946	Strong reliability.
		PEOU	0.843	Strong reliability.
		PU	0.867	Strong reliability.
Average Variance Extracted (AVE)	> 0.50	AU	0.821	Adequate convergent validity.
		BI	0.854	Adequate convergent validity.
		PEOU	0.574	Adequate convergent validity.
		PU	0.62	Adequate convergent validity.

Variance Inflation Factor (VIF)	1 ≤ VIF < 2	AU1, AU2, AU3	3.711, 2.638, 2.436	Acceptable level of collinearity for AU indicators.
		BI_1, BI_2, BI_3	3.580, 3.528, 2.737	Acceptable level of collinearity for BI indicators.
		PEOU1, PEOU3, PEOU4, PEOU5	1.507, 1.321, 1.556, 1.470	Minimal collinearity for PEOU indicators.
		PU1, PU2, PU3, PU4	1.882, 1.732, 1.438, 1.557	Minimal collinearity for PU indicators.
Heterotrait-Monotrait Ratio (HTMT)	< 0.90	AU - BI	0.565	Discriminant validity established.
		AU - PEOU	0.577	Discriminant validity established.
		AU - PU	0.49	Discriminant validity established.
		BI - PEOU	0.884	Discriminant validity established.
		BI - PU	0.863	Discriminant validity established.
		PU - PEOU	1.025 (exceeds threshold)	Discriminant validity concern between PU and PEOU.

Structural Model without Control Variable

Metric	**Threshold**	**Model Aspect**	**Analysis Result**	**Interpretation**

Path Coefficient (β)	Significant ($p < 0.05$)	BI → AU	$\beta = 0.515$, $p < 0.05$	BI has a significant positive effect on AU.
		PEOU → BI	$\beta = 0.408$, $p < 0.05$	PEOU has a significant positive effect on BI.
		PEOU → PU	$\beta = 0.807$, $p < 0.05$	PEOU has a strong positive effect on PU.
		PU → BI	$\beta = 0.413$, $p < 0.05$	PU has a significant positive effect on BI.
R^2 (Explained Variance)	Higher is better	BI	$R^2 = 0.54$	BI is moderately explained by its predictors (PEOU and PU).
		AU	$R^2 = 0.52$	AU is moderately explained by BI.
Standardized Root Mean Residual (SRMR)	< 0.08	Model Fit	Estimated Model: 0.085	Slightly above threshold, minor model fit concerns.

The measurement model in Table 9 demonstrates strong reliability and validity across all constructs (AU, BI, PEOU, PU). Cronbach's Alpha and Composite Reliability (CR) values exceed 0.70, ensuring internal consistency, while AVE values meet the 0.50 threshold, confirming convergent validity. Minimal collinearity is indicated by VIF values below 5. Discriminant validity is established for most construct pairs (HTMT < 0.90), except for PU-PEOU (HTMT = 1.025), indicating some overlap. The removal of PU5 and PEOU2 due to low factor loadings (< 0.70) improved construct reliability and representation, addressing potential issues in the measurement model.

The structural model in Table 9 confirms all hypothesized paths as statistically significant ($p < 0.05$). BI positively affects AU ($\beta = 0.515$), while PEOU significantly impacts BI ($\beta = 0.408$) and PU ($\beta = 0.807$). PU also positively influences BI ($\beta = 0.413$). The model explains a moderate variance ($R^2 = 0.54$ for BI, $R^2 = 0.52$ for AU) and aligns with the TAM, validating the proposed relationships. Although the SRMR (0.085) slightly exceeds the recommended threshold, the model remains robust and highlights the importance of PEOU, PU, and BI in predicting AU in the adoption of LCDPs.

As explained before, this study tested the results including and excluding the control variable. Table 10 will represent the results including Job Role as control variable.

Table 10 - *Results of the Measurement Model and Structural with Job Role being Control Variable*

Measurement Model with Technical Job Role as control variable				
Metric	**Threshold**	**Construct**	**Analysis Result**	**Interpretation**
Cronbach's Alpha	> 0.70	AU	0.891	Good internal consistency.
		BI	0.914	Excellent internal consistency.
		PEOU	0.753	Acceptable internal consistency.
		PU	0.795	Acceptable internal consistency.
Composite Reliability (CR)	> 0.70	AU	0.932	Strong reliability.
		BI	0.946	Strong reliability.
		PEOU	0.843	Strong reliability.
		PU	0.867	Strong reliability.

Average Variance Extracted (AVE)	> 0.50	AU	0.821	Adequate convergent validity.
		BI	0.854	Adequate convergent validity.
		PEOU	0.574	Adequate convergent validity.
		PU	0.62	Adequate convergent validity.
Variance Inflation Factor (VIF)	$1 \leq VIF < 5$	AU1, AU2, AU3	3.711, 2.638, 2.436	Acceptable level of collinearity for AU.
		BI_1, BI_2, BI_3	3.580, 3.528, 2.737	Acceptable level of collinearity for BI.
		PEOU1, PEOU3, PEOU4, PEOU5	1.507, 1.321, 1.556, 1.470	Minimal collinearity for PEOU.
		PU1, PU2, PU3, PU4	1.882, 1.732, 1.438, 1.557	Minimal collinearity for PU.
HTMT	< 0.90	AU- BI	0.565	Discriminant validity established.
		AU- PEOU	0.577	Discriminant validity established.
		AU- PU	0.49	Discriminant validity established.
		BI - PEOU	0.884	Discriminant validity established.

		BI - PU	0.863	Discriminant validity established.
		PU - PEOU	1.025	Discriminant validity concern between PU & PEOU.
Structural Model with Control Variable (Job_Role_Tech)				
Metric	**Threshold**	**Model Aspect**	**Analysis Result**	**Interpretation**
Path Coefficient (β)	Significant ($p < 0.05$)	BI → AU	$\beta = 0.535$, $p < 0.05$	BI has a significant positive effect on AU.
		PEOU → BI	$\beta = 0.408$, $p < 0.05$	PEOU has a significant positive effect on BI.
		PEOU → PU	$\beta = 0.807$, $p < 0.05$	PEOU has a strong positive effect on PU.
		PU → BI	$\beta = 0.413$, $p < 0.05$	PU has a significant positive effect on BI.
		Job_Role_Tech → AU	$\beta = 0.351$, $p = 0.035$	Job_Role_Tech (Technical roles) has a significant positive effect on AU.
R^2 (Explained Variance)	Higher is better	BI	$R^2 = 0.54$	BI is moderately explained by PEOU and PU.
		AU	$R^2 = 0.52$	AU is moderately explained by BI and Job_Role_Tech.
Standardized Root Mean Residual (SRMR)	< 0.10	Model Fit	Estimated Model: 0.095	Slightly above threshold, minor model fit concerns.

The measurement model in Table 10 demonstrates strong reliability and validity for all constructs (AU, BI, PEOU, PU). Cronbach's Alpha and Composite Reliability (CR)

values exceed the threshold of 0.70, ensuring good internal consistency and strong reliability. Convergent validity is confirmed as all constructs meet the Average Variance Extracted (AVE) threshold of 0.50. Collinearity is minimal, with all VIF values below 5, indicating no multicollinearity concerns. Discriminant validity is established for most construct pairs based on the HTMT criterion (< 0.90), except for PU-PEOU (HTMT = 1.025), which suggests a lack of discriminant validity between these two constructs. Overall, the measurement model remains robust despite the observed overlap between PU and PEOU.

The structural model in Table 10 confirms that all hypothesized paths are statistically significant ($p < 0.05$). BI significantly influences AU ($\beta = 0.535$), while PEOU positively affects both BI ($\beta = 0.408$) and PU ($\beta = 0.807$). Additionally, PU significantly influences BI ($\beta = 0.413$). The inclusion of Job Role (Technical) as a control variable shows a significant positive effect on AU ($\beta = 0.351$, $p = 0.035$), indicating that technical roles are more inclined to adopt the platform. The model explains a moderate proportion of variance, with R^2 values of 0.54 for BI and 0.52 for AU. While the SRMR value (0.095) slightly exceeds the recommended threshold, the model fit remains acceptable. These results validate the proposed relationships and emphasize the role of technical users in platform adoption.

5.2. Discussion of the Results

Discriminant Validity (PU-PEOU): The results of the Heterotrait-Monotrait Ratio (HTMT) analysis revealed a value exceeding the threshold of 0.90 (HTMT = 1.025) for the constructs PU and PEOU. This high correlation suggests a potential overlap between these constructs, which could indicate conceptual redundancy. This overlap may be caused by the inherent interdependence of these constructs within the TAM, where ease of use often directly contributes to the perceived utility of a system. While the results remain statistically valid, the overlap requires caution when interpreting the unique contributions of PU and

PEOU to BI. Future studies should consider refining these constructs to enhance discriminant validity.

SRMR and Model Complexity: Including Job_Role_Tech as a control variable increased SRMR to 0.095, slightly exceeding the ideal threshold. This reflects added complexity from the binary nature of Job_Role_Tech but does not undermine the model's robustness.

Impact of Job_Role_Tech: Job_Role_Tech significantly affects AU ($\beta = 0.351$, $p = 0.035$), highlighting the importance of technical expertise in adoption. Tailored training for non-technical users could address adoption gaps.

Explanatory Power (R^2 Values): Moderate R^2 values for BI (0.54) and AU (0.52) indicate that PEOU, PU, and Job_Role_Tech explain much of the variance, though future research could explore factors like organizational support or user experience for better predictive power.

Practical Implications: Improving platform usability (PEOU → PU, $\beta = 0.807$) and leveraging technical users as champions can enhance adoption. Strategies should address diverse user needs, ensuring inclusivity and maximizing platform benefits.

5.3. HYPOTHESES EVALUATION

Based on the findings on the PLS-SEM and Bootstrapping results, all the hypotheses are supported as described in Table 11.

Table 11 - *Hypotheses Testing Results*

Hypothesis	Description	Result	Interpretation

H1	PEOU positively influences the PU of LCDPs.	**Supported** ($\beta = 0.807$, $p < 0.05$)	PEOU has a strong positive and significant effect on PU.
H2	PEOU positively influences the Behavioral Intention BI to use LCDPs.	**Supported** ($\beta = 0.408$, $p < 0.05$)	PEOU has a significant positive effect on BI.
H3	PU positively influences the Behavioral Intention BI to use LCDPs.	**Supported** ($\beta = 0.413$, $p < 0.05$)	PU has a significant positive effect on BI.
H4	BI positively influences the AU of LCDPs.	**Supported** ($\beta = 0.535$, $p < 0.05$)	BI has a significant positive effect on AU.

The hypotheses testing results confirm the theoretical relationships within the model, emphasizing the importance of PEOU and PU in shaping behavioral intentions and AU. The strong influence of PEOU on Perceived Usefulness PU highlights the need for intuitive and user-friendly designs in low-code platforms. Similarly, the significant role of Behavioral Intention BI in predicting AU underscores the importance of fostering positive attitudes toward adoption. These findings validate the conceptual framework and provide actionable insights for improving low-code platform adoption strategies.

6. DISCUSSION

The findings validate the proposed conceptual model and provide significant insights into the adoption of LCDPs. The positive relationship between PEOU and PU highlights the critical importance of user-friendly designs in fostering platform adoption. This underscores the need for intuitive interfaces that reduce the cognitive burden on users, particularly for non-technical roles. Similarly, the significant effect of BI on AU reflects the importance of cultivating positive user perceptions and readiness for adoption. Job_Role_Tech, as a control variable, reveals the stronger inclination of technical users toward adoption, suggesting that organizational strategies should focus on bridging the gap for non-technical users through tailored training and support.

However, the overlap between PU and PEOU, as indicated by the HTMT value exceeding the threshold, highlights a potential need for construct refinement. This suggests that future research should re-evaluate the item design to ensure clear distinctions between constructs. Furthermore, the slightly elevated SRMR value when Job_Role_Tech is introduced indicates the complexity added by binary control variables, which may require more nuanced modeling to capture their variance accurately.

These results have both theoretical and practical implications. From a theoretical standpoint, the study extends the TAM by demonstrating its applicability to LCDPs and emphasizing the role of technical expertise in adoption. Practically, the findings provide actionable insights for organizations and LCDP providers, such as the importance of usability enhancements and targeted onboarding strategies to drive adoption across diverse user groups.

6.1. Limitations and Future Work

A notable limitation of this study reveals an overlap between the constructs PU and PEOU, as indicated by HTMT values exceeding the recommended threshold of 0.90. This overlap challenges the discriminant validity of the measurement model and suggests potential conceptual redundancy between these constructs. Future research should explore alternative operationalizations of PU and PEOU to ensure clearer differentiation. For instance, adopting additional constructs, such as system satisfaction or task-related effectiveness, could help mitigate overlap. Moreover, testing the model across different contexts and user groups may reveal more nuanced distinctions between PU and PEOU. Additionally, longitudinal studies are recommended to explore how PU and PEOU evolve over time and influence long-term adoption behaviors.

Second, this study focused on individual-level factors, but organizational dynamics such as culture, leadership support, and change management practices could also play a significant role in adoption. Future research should investigate these contextual factors to provide a more comprehensive understanding of LCDP adoption.

Third, the inclusion of Job_Role_Tech as a control variable highlights its significance, but future studies could explore its interaction with other variables, such as organizational size or industry type, to capture nuanced adoption patterns.

Finally, the emerging integration of artificial intelligence (AI) in LCDPs presents an exciting area for exploration. Future research could examine how AI features, such as automated decision-making and predictive analytics, impact user perceptions and adoption. Cross-platform comparisons, focusing on scalability and security concerns, would also offer valuable insights for improving platform design and implementation.

By addressing these areas, future research can build on this study's findings, contributing to both academic knowledge and practical strategies for advancing the adoption and effectiveness of LCDPs.

7. CONCLUSION

This paper has discovered the factors influencing user acceptance and adoption LCDPs among diverse user groups, utilizing a simplified TAM as the conceptual framework. During a combination of systematic literature review, survey-based empirical analysis, and statistical modeling as PLS-SEM, the study identifies both drivers and inhibitors that affect adoption between technical and non-technical user profiles.

The findings highlight that PEOU and PU are critical predictors of behavioral intention to adopt LCDPs, with technical users exhibiting higher adoption rates than non-technical ones. Key drivers of adoption include improved development efficiency, cost savings, and empowerment of non-technical users, while significant inhibitors such as security concerns, limited customization, and vendor lock-in cause hesitation for broader acceptance.

From a practical perspective, the research points out the potential of LCDPs to democratize application development and bridge the gap between technical and non-technical users. By reducing reliance on highly skilled developers, these platforms can drive digital transformation, foster innovation, and enhance organizational agility.

While the results are promising, this study has limitations, such as the relatively small sample size and its focus on specific adoption constructs. Future research could expand on these findings by exploring additional contextual factors, such as organizational culture and cross-industry comparisons, to develop a more comprehensive understanding of LCDP adoption dynamics.

In conclusion, this paper underscores the transformative potential of LCDPs in shaping the future of software development. By addressing both drivers and inhibitors, organizations

and platform providers can foster inclusive and efficient adoption, ensuring that LCDPs fulfill their promise as a driver of innovation in the digital economy.

BIBLIOGRAPHICAL REFERENCES

Alamin, M. A. Al, Uddin, G., Malakar, S., Afroz, S., Haider, T., & Iqbal, A. (2023). Developer discussion topics on the adoption and barriers of low code software development platforms. *Empirical Software Engineering*, *28*(1). https://doi.org/10.1007/s10664-022-10244-0

Beranič, T., Rek, P., & Heričko, M. (2020). *Adoption and Usability of Low-Code/No-Code Development Tools*.

Beynon-Davies, P., Came, C., Mackay, H., & Tudhope, D. (1999). Rapid application development (Rad): An empirical review. *European Journal of Information Systems*, *8*(3), 211–232. https://doi.org/10.1057/palgrave.ejis.3000325

Bies, L., Weber, M., Greff, T., & Werth, D. (2022). A Mixed-Methods Study of Low-Code Development Platforms: Drivers of Digital Innovation in SMEs. *International Conference on Electrical, Computer, Communications and Mechatronics Engineering, ICECCME 2022*. https://doi.org/10.1109/ICECCME55909.2022.9987920

Bill Doerrfeld. (2022, March 2). *The Pros and Cons of Low Code/No Code and Rapid App Development*. Acceleration Economy. https://accelerationeconomy.com/low-code-no-code/the-pros-and-cons-of-low-code-no-code-tools-and-rapid-application-development-platforms-that-everyone-needs-to-know/

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly: Management Information Systems*, *13*(3), 319–339. https://doi.org/10.2307/249008

Di Sipio, C., Di Ruscio, D., & Nguyen, P. T. (2020). Democratizing the development of recommender systems by means of low-code platforms. *Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020 - Companion Proceedings*, 471–479. https://doi.org/10.1145/3417990.3420202

Dushnitsky, G., & Stroube, B. K. (2021). Low-code entrepreneurship: Shopify and the alternative path to growth. *Journal of Business Venturing Insights*, *16*. https://doi.org/10.1016/j.jbvi.2021.e00251

Frank, U. ;, Maier, P. ;, & Bock, A. (2021). *Low code platforms: Promises, concepts and prospects. A comparative study of ten systems*. https://doi.org/10.17185/duepublico/75244

How Much Does Low-code Development Cost? (n.d.). Retrieved January 6, 2024, from https://synodus.com/blog/low-code/low-code-development-cost/

Lebens, M., & Finnegan, R. (2021). *Rise of the Citizen Developer* (Vol. 5).

Martinez, E., & Pfister, L. (2023). Benefits and limitations of using low-code development to support digitalization in the construction industry. In *Automation in Construction* (Vol. 152). Elsevier B.V. https://doi.org/10.1016/j.autcon.2023.104909

Overeem, M. (2022). *Evolution of Low-Code Platforms*. www.ridderprint.nl

Pinho, D., Aguiar, A., & Amaral, V. (2023). What about the usability in low-code platforms? A systematic literature review. *Journal of Computer Languages*, *74*. https://doi.org/10.1016/j.cola.2022.101185

Quick Comparison between Low Code vs High Code: Which wins? (n.d.). Retrieved January 6, 2024, from https://synodus.com/blog/low-code/low-code-vs-high-code/

Rokis, K., & Kirikova, M. (2022). Challenges of Low-Code/No-Code Software Development: A Literature Review. *Lecture Notes in Business Information Processing*, *462 LNBIP*, 3–17. https://doi.org/10.1007/978-3-031-16947-2_1

Rymer, J. R., & Richardson, C. (2015). *Low-Code Platforms Deliver Customer-Facing Apps Fast, But Will They Scale Up?*

Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. *Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020*, 171–178. https://doi.org/10.1109/SEAA51224.2020.00036

Santis, C. (n.d.). *How Low-Code Development Helps Companies Increase Productivity*. Retrieved November 7, 2023, from https://www.pillir.io/edgeucation-center/blog/how-low-code-helps-companies-increase-productivity

Steffen, B., & Margaria, T. (2021). Leveraging Applications of Formal Methods, Verification and Validation. In *Proceedings of the 10th International Symposium on Leveraging Applications of Formal Methods, Verification, and Validation (Isola 2021): Vol. LNCS 13036* (p. Proceedings). http://www.springer.com/series/7407

The Real Cost of Low Code Applications | Medium. (n.d.). Retrieved January 6, 2024, from https://medium.com/@ar.arun/the-real-cost-of-low-code-applications-3be9dece1b59

Venkatesh, V., & Davis, F. D. (1996). A model of the antecedents of perceived ease of use: Development and test. *Decision Sciences*, *27*(3), 451–481. https://doi.org/10.1111/j.1540-5915.1996.tb00860.x

Venkatesh, V., & Davis, F. D. (2000). Theoretical extension of the Technology Acceptance Model: Four longitudinal field studies. *Management Science*, *46*(2), 186–204. https://doi.org/10.1287/mnsc.46.2.186.11926

Wang, Z., Jing, Z., Li, S., Qi, G., & Wang, Z. (2023). Research and Application of Multi-Source Data Collection Method of Power System Based on Microservice Idea. *Proceedings - 2023 International Conference on Power System Technology: Technological Advancements for the Construction of New Power System, PowerCon 2023*. https://doi.org/10.1109/PowerCon58120.2023.10331234

Waszkowski, R. (2019). Low-code platform for automating business processes in manufacturing. *IFAC-PapersOnLine*, *52*(10), 376–381. https://doi.org/10.1016/j.ifacol.2019.10.060

Zhuang, W., Gan, X., Wen, Y., & Zhang, S. (2022). EasyFL: A Low-Code Federated Learning Platform for Dummies. *IEEE Internet of Things Journal*, *9*(15), 13740–13754. https://doi.org/10.1109/JIOT.2022.3143842

APPENDIX A

Summary of LCDP vendors examined in Frank et al. (2021)

Vendor	Accessibility and Convenience of Use	Modes of Use	Focus on Deployment and Scalability	Focus on AI
Quickbase	no specific methodical support for the development fairly convenient and easy to use. GUI pages can be adjusted according to different user roles. use of largely inaccessible, proprietary modeling languages	**Platform (i.e., Application Development)** CRUD-based right specification (application users as application developers) no further support for collaborative application development **Application** CRUD-based right specification scarce integration capabilities between applications no further support for the collaborative use of the platform	cloud-based platform, accessible via a regular web browser data persisted on the platform. no further accessible support mechanisms.	no AI services included
TrackVia	no methodical development support fairly convenient and easy to use. no adaption to specific user groups use of largely inaccessible, proprietary modeling languages	**Platform (i.e., Application Development)** access to applications granted by "super administrator" (application users as application developers) no direct access to external database schemata	cloud-based platform, accessible via a regular browser outside accessibility through RESTful API	no AI services included in TrackVia

		no further support for collaborative application development **Application** no distinction of CRUD rights no further support for the collaborative use of the platform		
Bonita Studio	no methodical development support implementation and use split into three environments. implementation in Bonita Studio demands some familiarity with programming concepts. use of BPMN for workflows	**Platform (i.e., Application Development)** need for two supplementing environments: Form Designer and Bonita Studio no specific support for collaborative development **Application** access to developed applications is granted through Bonita Portal definition of hierarchical role structures with a set of predefined interrelations (e.g., “is the manager of”) roles are defined as a set of workflow elements within one lane	application hosting needs to be managed by a developer. no further accessible support mechanisms	no AI services identified for Bonitasoft
Creatio Studio	no methodical development support fairly convenient and easy to use. access to the underlying database and source code to most user-developed artifacts	**Platform (i.e., Application Development)** internal chat functionalities shall support collaborative application development.	cloud-based platform, accessible via a regular web browser no further accessible	Separate ML module enables specification and configuration of inductive, supervised ML models

	use of well-known workflow modeling language	development and use of applications not distinguished. **Application** the distinction between functional and organizational roles organizational roles can be arranged hierarchically to aggregate specified CRUD rights on higher levels. platform chat shall support collaborative application use. separate Creatio Portal shall enable restricted access to organization-external users	support mechanisms	
Mendix Studio	two IDEs (Mendix Studio and Mendix Studio Pro) address different kinds of developers - Mendix Studio is fairly convenient and intuitive -Mendix Studio Pro demands some training and knowledge of development-related concepts methodical support shall be provided through the inclusion of Scrum-related concepts like user stories, product backlog, and others. use of multiple models for data and	**Platform (i.e., Application Development)** locally deployed IDE that runs a version control system. access rights are managed through "module roles." **Application** access rights are managed through "module roles." no inherent support for the collaborative use applications	Mendix public cloud server per default, an application can be accessed via the web browser. alternative deployment options (e.g., private cloud) are also supported. use of Kubernetes for container orchestration	Mx Assist Logic Bot to support modeling of workflows not convincing

	different workflows that are based on standard modeling notations			
WaveMaker Studio	no methodical application development support Except for superficial data model adjustment and GUI development, the platform largely requires some. sort of source code specification proprietary data modeling language	**Platform (i.e., Application Development)** version control to support collaborative development. elaborate definition of developer roles not assessable in the considered version **Application** no direct support for collaborative application use by WaveMaker	cloud-based platform, accessible via a regular browser The domain name can be customized for applications. support to build docker images	no specific support for any AI service
Zoho Creator	no methodical application development support fairly convenient and easy to use. access to source code in proprietary programming language for more advanced users use of largely inaccessible, proprietary modeling languages	**Platform (i.e., Application Development)** no further support for collaborative application development **Application** distinction of role and permission types role types shall resemble organizational roles and are. used in some workflow types. permission types specify CRUD rights	cloud-based platform, accessible via regular web browser or Zoho app for mobile devices no further accessible support mechanisms	pre-implemented, inaccessible AI “fields” in GUI “forms” that produce ML classifications not convincing
Microsoft Power Apps	Guidance -no specific development method -modest guidance through different modes of use	**Platform (i.e., Application Development)** definition and management of access rights	development environment runs in the web browser. runtime systems available for	various pre-trained ML solutions, such as text recognition, are available

	LCP user interface -based on proprietary widgets and general GUI framework. offered modeling languages -proprietary process modeling language	conjoint use with other Microsoft IDEs **Application** definition and management of access rights use in distributed environments. possible in heterogeneous environments by running Power Apps on different platforms for which it is available.	Windows, Android, and iOS	with the platform
Appian	no methodical development support the vast number of available design objects and confusing terminology present a burden to the initial use of the platform -the "Quick Apps Designer" module is insufficient to overcome this limitation since its features are too restricted -Appian is thus hardly adaptable to a diverse set of user groups and demands some experience in application development Data modeling language lacks expressiveness	**Platform (i.e., Application Development)** users of Appian must be assigned to some user group. Each design object must specify the required CRUD rights for all user groups. user notifications can be embedded in workflows. methodical application development support is not provided. **Application** different users can be provided with different GUI screens. no inherent support for collaborative use of an Appian application	cloud-based platform, accessible via a regular web browser support to deploy Appian applications in Dockers containers	"Next-best action" in workflow models not convincing support for integration of AI services as offered by Google Cloud Platform
Pega Platform	user-friendly App Studio is intuitive and convenient to use. Dev Studio is more demanding and	**Platform (i.e., Application Development)** version control system, "agile workbench",	cloud-based platform, accessible via a regular web browser	provision of some generic pre-trained ML models

	requires some training first. use of proprietary modeling languages	“developer workbench” shall support collaborative development. allows shared access to user stories, comments, identified bugs, and feedback. detailed specification of different developer roles and rights in Dev Studio **Application** different application users can be assigned different GUI screens. end users are assigned a “persona” that specifies CRUD-based rights	Focus on AI possibility to define custom branches. support for Docker deployment	support to integrate Google AI and SageMaker services

APPENDIX B - ETHICS COMMITTEE REPORT

Thank you for filling in the Research Ethics Checklist. After reviewing your request, you can proceed with the study we do not foresee any major ethical concerns with the project.

Project No.: **DDMKT2024-7-106749**

Project Title: **Factors Influencing User Acceptance and Adoption Among Different User Groups of Low-code Development Platforms**

Principal Researcher: **Elif Uslu**

according to the regulations of the Ethics Committee of NOVA IMS and MagIC Research Center this project was considered to meet the requirements of the NOVA IMS Internal Review Board, being considered **APPROVED** on 11/07/2024.

It is the Principal Researcher's responsibility to ensure that all researchers and stakeholders associated with this project are aware of the conditions of approval and which documents have been approved.

The Principal Researcher is required to notify the Ethics Committee, via amendment or progress report, of

- Any significant change to the project and the reason for that change;

- Any unforeseen events or unexpected developments that merit notification;

- The inability of the Principal Researcher to continue in that role or any other change in research personnel involved in the project.

Lisbon, 11/07/2024

NOVA IMS Ethics Committee

ethicscommittee@novaims.unl.pt


NOVA
IMS
Information
Management
School