Low/No Code Development and Generative AI

Thesis Report

Nourjan Sido
Eksan Ahmed Emon

Aalborg University, Copenhagen
Electronic Systems

Electronic Systems
Aalborg University, Copenhagen
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Low/No Code Development & Generative AI

Theme:
Thesis

Project Period:
Summer Semester 2024

Project Group:

Participant(s):
Nourjan Sido
Eksan Ahmed Emon

Supervisor(s):
Morten Falch

Page Numbers: 108

Date of Completion:
May 31, 2024

Abstract:

The landscape of software development is continuously evolving, with new technologies regularly emerging. This thesis aims to investigate the potential of low-code development, a rising technology, and its impact on the software development process. Additionally, it examines how the integration of generative AI, another trending technology, can further accelerate this impact. By exploring the synergy between generative AI and low-code development, we aim to understand its implications for the future of software development and address current limitations in low-code platforms. Furthermore, a business analysis is conducted to evaluate the market positioning of these technologies. Our investigation reveals these technologies' immense potential in transforming the software development landscape.

Contents

Chapter 1

Introduction

In the ever-changing technological world, a revolution is brewing, ready to transform the way software is developed. The growth of low/no-code development platforms represents a significant shift in how software is designed, distributed, and managed in today's dynamic digital environment [1] [2]. This disruptive development method democratizes app creation, taking it outside the traditional sphere of professional programmers. This shift is driven by two major forces: low/no-code Development Platforms (LCDPs) and Generative AI (Gen AI). As these technologies become more prevalent in the industry, they can democratize development, accelerate digitalization, and improve software production efficiency [1].

low/no-code development platforms (LCDPs) provide simple visual interfaces and prebuilt components, enabling anybody without coding knowledge to create usable applications [2]. This enables "citizen developers" (business users and non-technical users) to contribute into software development, potentially increasing agility and innovation [3].The growing need for solutions that simplify a company's operations has created opportunities for using platforms designed for low/no-code development. These platforms include pre-packaged features that may be utilized by both non-programmers inside a firm and software developers to construct software applications and automate operations without the need for human programming [4]. However, the promise of LCDPs extends beyond citizen development. By automating repetitive tasks, producing code snippets, and enabling speedier prototyping, they can enhance traditional software development processes. This has the potential to significantly enhance development efficiency while cutting costs for experienced professionals [5].

The low/no-code (LCNC) sector is quickly expanding, with excellent estimates and figures demonstrating its importance in the IT industry. Gartner believed that the LCNC industry would grow to $26.9 billion by 2023 [6], while Forrester expects a $45.5 billion market by 2025. Furthermore, by 2024, low-code approaches are predicted to account for 65% [6] of

app development activity, indicating that these platforms are becoming increasingly popular.

The rise of gen AI is fueling this shift even further. This cutting-edge technology includes several strategies that enable computers to produce innovative and creative outputs such as text, code, and graphics [7]. gen AI has enormous potential to transform software development by automating many processes formerly undertaken by humans, such as code creation, testing, and documentation [8]. This automation can not only improve development workflows but can also customize experiences and tailor solutions to individual project demands, resulting in increased productivity and creativity.

The possible collaboration between LCDPs and gen AI is a positive indicator for the future of software development. Integrating gen AI capabilities into LCDPs might simplify development processes, empower citizen developers, and enable even more efficiency and creativity [9]. This collaboration has huge promise for many industries, enabling faster time-to-market, improved software quality, and more business agility. However, managing in this changing environment has a variety of challenges. Ensuring the quality and security of AI-generated code, addressing ethical concerns around gen AI development and deployment, and managing potential disruptions to established development processes are all key issues that must be carefully handled.

Given these successes and difficulties, we intend to explore the current state of software engineering methodologies and identify the potential of low/no-code development to help in the discussion of those development methodologies and provide a way to digital innovation. Our research will concentrate on the low/no-code software development methodology and its implications on digitalization and business innovation. We intend to perform an analysis from both business and technical perspectives. Furthermore, we will explain how low code platforms work, identify the the life cycle of the development process and investigate how Gen AI can manifest itself over the development process.

1.1 Research Motivation

In the fast-changing technology world, software development is considered to be one of the biggest factors that foster innovation, productivity, and digital transformation in different sectors. Traditional methods for software development were based on a coding process that was complicated and required skilled programmers; for so long, it has served as the main pillar of software development industry. Although the concept of low/no-code development platforms (LCDPs) and generative AI technology (gen AI) is challenging the traditional paradigm and reshaping the way software is created and delivered, the new norm in the software industry will most probably be the adoption of these two trends [10].

The motivation behind our research is the acknowledgement of the transformative power of low/no-code development in addition to generative AI technologies. These innovations serve as disruptive forces for all software developers that aim to speed up the digitalization process and revolutionize the process for the creation of efficient and accessible apps. LCDPs allow individuals who do not have coding skills to be included in the development process and help boost agility, innovation, and technology democratization. On the one hand, the advancements of generative AI have made it possible to apply automation and augmentation, which on the other hand will accelerate the workflows, enhance the creativity and come up with solutions that fit the specific project requirements.

The core motivation of this research is the notion that low/no-code development, along with generative AI, can revolutionize software development and digitalization processes. Our goal is to examine how these technologies can be leveraged together and provide the necessary instructions, thus allowing us to contribute to a future where software development is increasingly affordable, faster, and inclusive.

1.2 Problem Statement

The integration of Generative AI and low/no-code development platforms promises to revolutionize the future of software development. By combining the advance capabilities of generative AI with the accessibility of low/no-code development platforms, these technologies can enhance productivity, reduce development time, and democratize software developments. Hence, it leads us to the following problem statement:

"How will the offering of low/no-code solutions, enhanced by generative AI technology, affect the future of software development?"

To answer the problem statement of our thesis, we identified the following research subquestions:

1.2.1 Research Sub-questions:

- How have the limitations of traditional and modern software development methodologies influenced the rise of low/no-code platforms and generative AI technologies?
- What are the core components and key features that define a development approach as "low/no-code" development?
- How can the integration of generative AI across the stages of the low code development process enhance its capabilities?
- How does integrating generative AI enhance the business value and competitive market positioning of low/no-code development platforms?

1.3 Delimitation

There are several delimitations faced during the research process in this thesis:

- Theoretical Approach: Some aspects of this thesis, such as the integration of generative AI in low-code development, are discussed only theoretically without actual implementation. The complexity of the topic requires more time and resources for technical realization than was available for this thesis.
- Available Resources: The technologies under investigation—low-code development platforms and generative AI—are relatively recent. Consequently, the research on their synergy is limited, which constrains our ability to draw comprehensive conclusions.
- Data Collection: Due to the limited research available, we intended to collect primary data by contacting relevant companies and conducting interviews. However, our efforts were constrained by the availability and willingness of companies to participate.
- Company Participation: Given that this technology is not widely adopted in our region, we found only a single company specializing in this field. Unfortunately, our request for collaboration with this company was declined, limiting our access to practical insights and firsthand information.

Chapter 2

Methodology

This chapter outlines the research approach and processes we followed to explore the current state and potential of low/no-code development platforms and generative AI technologies. We will provide a summarized explanation of what we have done in each individual chapter of the report to address the problem formulation of our thesis and the research subquestions identified in relation to our problem statement. By systematically addressing each chapter's key focus and intended outcomes, this chapter will offer a clear and structured framework for understanding how these emerging technologies can transform current software development practices and create business opportunities as well as digital innovation. The following sections will describe the methods we followed and the insights we gained from our exploration of low/no-code development and generative AI technologies.

2.1 Project Approach

In our research, we followed a document analysis approach so that we can gain a comprehensive understanding of the low/no-code development and generative AI technologies. This involved systematically reviewing and analysing existing literature, academic papers, industry reports, and relevant articles from databases like IEEE Xplorer, ACM Digital Library, and Google Scholar to gain historical context and understand significant advancement in these fields. We also looked at some industry reports and white papers from different big companies and firms such as Gartner, Forrester, and McKinsey for getting insights into the market scenario, trends and practical applications. We also reviewed and went through different articles and blog posts from different reputed technology websites and experts to stay updated on the latest developments and real-world case studies. By accumulating this information, we could identify the current market trends as well as emerging trends, and key insights that helped us to formulate our research questions and focus areas.

2.1.1 Background

For better understanding of the current software development methodologies we looked into the history of these development practices, how they evolve over time from manual coding to modern methodologies. We explored different traditional methodologies like waterfall, and spiral models highlighting their limitations and the subsequent emergence of iterative and incremental methods like rational unified process, agile, etc. This background study helped us to identify the core foundational shift that paved the way for low/no-code development and generative AI.

2.1.2 State of the Art

In this chapter, we explored the current state of low/no-code development and generative AI technologies. We looked at their functionalities, key features, benefits of usage and market adoption. We also looked at the differences between low and no code development methodologies. We looked at some of the existing platforms to understand their functionalities, features, and the challenges they currently have. We also discussed the advancements and applications of generative AI in software development. We explored the current implementation scenario of generative AI and low/no-code platforms as well as challenges of implementation.

2.1.3 Business Analysis

For the business analysis section of our report, we evaluated the market and strategic implications of low/no-code development platforms and generative AI technologies. We analyzed what are the market trends, adoption rates of these technologies, and key competitions available to understand the competitive landscape. We also utilized different business analysis frameworks like swot analysis, porter's five forces, value proposition canvas, and the business model canvas to demonstrate the business opportunities and challenges associated with the integration of generative AI technology into low/no-code development platforms.

Stakeholder Analysis

In the stakeholder analysis, we identified the stakeholders relevant to the successful implementation of generative AI into low/no-code development platforms. We made the power-interest grid for the identified stakeholders and placed them into specific grids based on their power, influence, interest, and contribution to the idea of integration.

SWOT Analysis

We performed the SWOT analysis so that we could identify the core strengths, weaknesses, opportunities, and threats associated with the integration of generative AI into the plat-

forms. This helped us to identify different factors both internal and external that can have an impact on the adoption and success of the integration.

Porter's Five Forces

We used the framework of porter's five forces to evaluate the market scenario and competitive landscape. This included analyzing the possible threat of new entrants, the bargaining power that both suppliers and buyers have, the threat of substitute products on the market, and the competitive rivalry intensity.

Value Proposition Canvas

The value proposition canvas was used to determine the kind of value that the users of low/no-code platforms would obtain after integrating AI. We mapped out customer requirements, pains they were facing, and this included generative AI integration that could help solve their problems and give them the expected value.

Business Model Canvas

We also developed a business model canvas around the idea of integrating generative AI into these platforms. We conceptually outlined the strategic positioning and potential revenue models for the successful implementation of the idea. We identified all the key components of the canvas, which helped us to understand how businesses can effectively leverage these technologies to create value, reach target customer segments, and generate sustainable revenue out of them.

2.1.4 Technical Analysis

The purpose of the technical analysis is to understand the architecture and functionality of low-code platforms, as well as their development life cycle, to investigate how emerging generative AI technologies can be integrated into this field. This part will be solely based on existing literature an research papers.

Chapter 3

Background

In this chapter, we provided a historical overview of software development methodologies from different traditional models like waterfall, spiral to modern methodologies like rational unified process and Agile. We discussed their development processes, principles, what kind of development is suitable for utilizing them, their advantages and disadvantages, and later how the evolution of low/no-code development happened to mitigate their limitations.

3.1 History of Software Development Practices

Software development approaches have transformed significantly in the last decades due to technological enhancements, changes in business needs, and software delivery models. The history of software development can be traced back as far back as the early 1950s and 1960s when programming was initially developed. During this period, the development process was often non-sophisticated and heuristic in nature with a minimum organized framework for a development process [11]. However, with the increased size of software projects, many people realized the need for more formalized methodologies.

Waterfall model is recognized as one of the first systematic approaches of software development that was introduced in the late 1960s and early 1970s [12]. The waterfall model developed by Winston W. Royce in 1970, offered a linear and strict methodology for software development where phases are well-defined [13]. While the Waterfall model offered a clear plan of action, it was criticized for its rigid structure and inability to accommodate changes in the requirements.

Later, in the 1980s and 1990s methods like incremental and iterative emerged to overcome the difficulties of waterfall model. In 1986, Barry Boehm developed the spiral model which combined the features of iterative and risk-based approaches that could be adjusted and improved during the development phases [14]. In addition, some other approaches

like rapid application development (RAD) and unified process (UP) emerged, focusing on the iterative approach and the participation of end-users [15].

It was only at the beginning of this century that this concept called Agile movement was popularised, claiming that the traditional methodologies had their shortcoming. The manifesto of agility, issued in 2001, detailed a set of stated ideals centered on humans and interactions, functional software, consumers, and rapid feedback to change [16]. This led to the rise of numerous Agile frameworks like scrum, kanban, XP, and lean software development, which encouraged iterative work, teamwork, and frequent delivery [17].

Alongside the agile movement, other complementary development approaches also gained attraction. Some of them are:

- **DevOps:** DevOps methodology emerged in the late 2000s and intended to close the communication gap between the operation and development teams to enhance collaboration, automation and continuous delivery [18].
- **Model-Driven Development:** MDD focuses on utilizing models as the key assets for software development, allowing automation and code generation [19].
- **Continuous Integration/Continuous Delivery (CI/CD):** Automating the build, testing, and deployment procedures was a key component of CI/CD methods, which allowed for frequent and dependable software releases [20].

Technologies such as cloud computing, containerization, and artificial intelligence (AI) have advanced significantly in recent years, impacting the software development landscape. Because of these improvements, new methods and techniques have evolved, such as:

- **Serverless Computing:** Serverless computing allows for users to create and deploy applications without managing the underlying infrastructure [21].
- **low/no-code Development:**This development method enables users (both technical and non-technical) to create software applications using visual interfaces and prebuiilt components [22].
- **AI-assisted Development:** Using machine learning and AI approaches to help with development, testing, and automatic code generation [23].

3.2 Tranditional Software Development Methodologies

Traditional software development has its roots in the early days of computer science, when software was developed sequentially and document-driven. This technique, known as the "Waterfall" model, was initially established in the 1970s and remained the predominant

paradigm for software development for several decades. [24].

The waterfall approach is linear in nature, with activities carried out indirectly related to the different stages of requirement gathering, design, implementation, testing, and deployment [24]. This approach completely specifies about the outcomes of each stage of the development process as well as the commitments of planning and documentation.

Over time, the traditional software development methodology has expanded to embrace a variety of approaches, including the spiral model, the V-model. These techniques emphasize prior planning, detailed documentation, and a disciplined phase-based approach to software development [24].

Some of the most common traditional development methods and their descriptions are described below:

- **Waterfall Model:** The waterfall model was first introduced first by Winston Royce, which is linear in nature [25]. This approach divides the development life cycle into segments that are convenient to manage. These phases are organized in a sequential manner, where a new phase begins only once another phase has been completed [26].

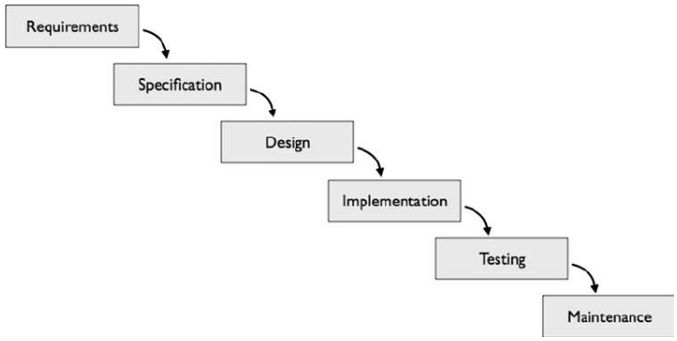

Figure 3.1: The Waterfall Model [27].

Phases of the Waterfall Model: The waterfall development model phases are described below [27]:

  - **Requirements:** In this phase the requirements of the users are collected and documented properly.
  - **Specification:** In this phase of the model, a formal document is created that contains all the requirements of the application.
  - **Design:** Based on the identified requirements and specifications, the architecture and design of the application are developed.

- **Implementation:** To translate the design into a functional system, the actual source code is written based on the system design documents in this section.
- **Unit Testing:** The components are tested individually in this phase of the development cycle.
- **Integration Testing:** In this stage of model all the components are combined and tested together here.
- **Maintenance:** In this last phase, the necessary enhancements and corrections of the final application is done based on the requirements.

Advantages and disadvantages of the waterfall model are described in the table:

Advantages	**Disadvantages**
– As the waterfall model is linear and sequential in nature so it provides a clear and structured approach to software development [26]. – Each phase produces detailed documentation, which aids in maintenance and knowledge transfer [26]. – The sequential flow of the model simplifies project management, especially for large teams and complex projects [25]. – The model assumes that all requirements will be collected at the beginning of the development, providing a clear roadmap for the development process [25].	– It is difficult to go back to early phases and make changes once a phase is completed, which makes the model less adaptable to evolving requirements [26]. – As testing is done after the implementation, it can result in late discovery of critical issues [26]. – The model requires full understanding and documenting of the requirements from the beginning which is difficult in real life scenarios [25]. – The model's rigidity can lead to inefficiencies and increased costs in long-term projects where changes are inevitable [25].

Table 3.1: Advantages and Disadvantages of the waterfall model [25]

The following types of projects are appropriate for the waterfall model [28]:

- Suitable for small-scale development.
- Stable and clear requirements are available/identified.
- Stable environment available.
- Stable development tools are available.

- Has well-trained resources available.

- **Spiral Model:** It is a risk-driven software development process that was developed by Barry Boehm in 1986 [29]. It is a higher-level model that integrates features of the other categories, namely, incremental, waterfall, and prototyping models, offering an orderly as well as cyclical method of developing application software.

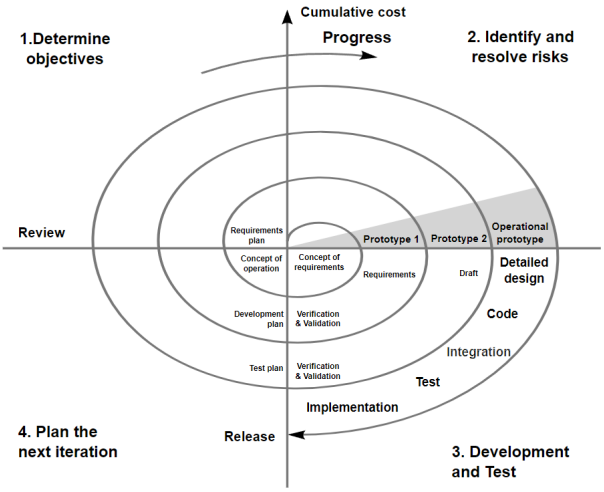

Figure 3.2: The spiral model [29].

Phases of the Spiral Model: Different phases of the spiral model are as follows [29]:

- **Planning:** It is the first phase of the spiral model where the main purpose of the project is identified. For the next phase of the model proper planning is also done in this phase.
- **Risk Analysis:** In this phase the risk related to the project is clearly identified and necessary assessment is also done for the identified risks.
- **Engineering:** After that the engineering phase comes where the software is built following the requirements that were collected in the previous phase of the model.
- **Evaluation:** Lastly in the evaluation phase, the software is evaluated to see whether the developed software fulfills requirements of the customer and also meets the quality standard.

  Applications of the Spiral Model: It is commonly used for these kinds of projects [30]:

  * Large Scale Projects

* Frequent updates and releases needed
* Prototyping Requirements
* Risk and cost evaluation
* Moderate to high risk
* Complex Requirements

Advantages and disadvantages of the spiral model are described in the table below [30]:

Advantages	**Disadvantages**
– Effective risk management through iterative cycles and continuous assessment. – Flexibility in accommodating changes based on stakeholder feedback and evolving requirements. – Allows for incremental releases, providing early prototypes and partial system implementations. – Enhances user involvement through regular reviews and evaluations, leading to better requirements validation.	– Complexity in managing multiple iterations and maintaining detailed documentation. – Higher cost and time investment due to iterative nature and continuous risk analysis. – Requires highly skilled project managers and developers with expertise in risk management. – Not suitable for smaller projects due to its extensive planning and resource requirements.

Table 3.2: Advantages and Disadvantages of the Spiral Model [30]

- **V-Model:** It is the evolution of the traditional waterfall model and also known as the verification and validation model. This model also incorporates a formal and structured approach to software development that follows a step-by-step process of software and puts importance on the verification and validation of the developed software. It is a linear model that has testing phases for each different stage of a project's development; therefore, it has a V shaped appearance [31]. This makes sure that each process is checked and verified before moving to the next step and this makes the work reliable and ensures the product quality.

  Phases of the V-Model: The phases of the V-Model are as follows [31]:

  – **Requirements Analysis:** This phase involves identifying the requirements for the system from stakeholders. These requirements are described in detail and

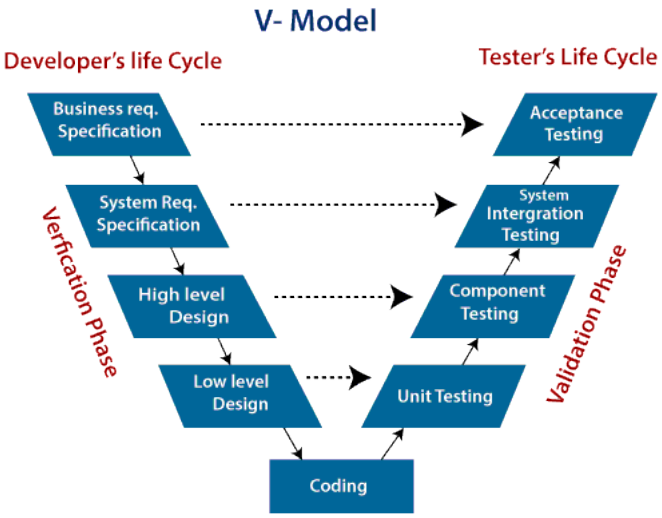

Figure 3.3: Diagram of V-model [31].

form the basis for all other stages.

- **System Design:** After that, both functional and non-functional requirements are described clearly, as well as the system elements and connections between them.
- **Architecture Design:** In this stage software architecture is designed in detail based on the system design.
- **Module Design:** In this stage the actual software components are planned and developed.
- **Coding:** The actual source code is then written based on the detailed design documents in this phase. Every module or component is developed and then tested separately.
- **Unit Testing:** Module tests are done to check whether each of the modules is functioning in the right manner. This phase deals with the testing phase whereby the correctness of a particular module is determined.
- **Integration Testing:** The modules are connected and the coupling between them is verified. This phase confirms that the integrated system components interact with each other in the desired manner.
- **System Testing:** To ensure that system has the required characteristics the verification of the integrated system is done in this phase.
- **Acceptance Testing:** This phase frequently includes end-users. In order to satisfy the needs and requirements of the end-users the system is implemented in the real world.

Applications of the V Model: The V-model is suitable and effective for projects or developments where the requirements are well-defined and have less chance of modifications [31].

- Suitable for small to medium-sized projects where requirements are clearly specified and finalized.
- Projects that have technical resources available and essential technical expertise also.
- Where accuracy and reliability are concerns and are often safety-critical as well as require thorough validation.

The table below illustrates the key advantages and disadvantages of V-Model [31]:

Advantages	**Disadvantages**
– Enhanced quality assurance is achieved through continuous verification and validation at each phase of the development cycle. – Clear structure and different phases make the project easy to manage and track progress. – Extensive documentation throughout each phase helps in understanding and maintaining the system. – Early detection of defects through systematic testing corresponding to each development phase.	– Inflexibility in accommodating changes once the project is underway, similar to the Waterfall model. – High documentation overhead can be time-consuming and costly. – Late discovery of some issues, as testing phases occur after development phases. – Not suitable for projects with unclear or frequently changing requirements.

Table 3.3: Advantages and Disadvantages of the V-Model

3.3 Modern Software Development Methodologies

The traditional approaches have been passed through a great deal of criticism due to their limits, sequential nature, and document-centricity. These methodologies often end up not being able to meet new requirements anymore, leading to problems such as delayed projects, high costs, and substandard software [32]. These challenges paved the way for the introduction of new methods of software development in an attempt to enhance the

efficiency of software development. They helped unblock the limitations of the old techniques through an emphasis on flexibility, cooperation, and continuous development [33].

Most of the software development processes that are followed today, like Agile, DevOps, and Rational Unified Process, are iterative and incremental processes through which software developers are able to respond to changes in requirements and deploy working software much more quickly [34]. The current software development practices involve multiple teams where developers, business people, and customers work together to ensure that the software being developed reflects the needs of the business and its customers.

Key characteristics of modern software development methodologies include:

- Iterative and incremental development
- Emphasis on collaboration and communication
- Continuous integration and deployment
- Agility and adaptability
- Focus on delivering value

Some of the most known modern development methodologies are described below:

- **Rational Unified Process (RUP):** The Rational Unified Process (RUP) is an iterative software development process that has been developed by the Rational Software Corporation, which is currently owned by IBM. RUP aims to offer a structured framework for project management and software development based on identified risks and collaboration [35]. The concept is not a prescriptive recipe but a framework for project teams and development organizations to follow and adapt to the needs as they see fit.

  RUP is built around six best practices in modern software engineering. They are [35]:

  - Developing Iteratively
  - Managing the requirements
  - Employing component-based architecture
  - Modeling software visually
  - Continuous verification of quality
  - Controlling changes

  Phases of RUP: The key phases of RUP model are described below:

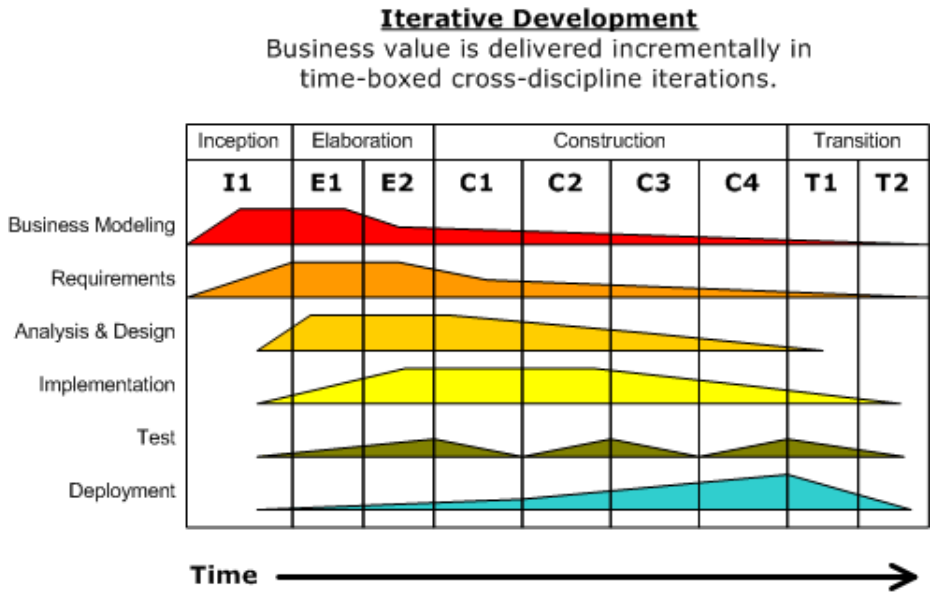

Figure 3.4: Rational Unified Process Model [35].

- **Inception:** In this phase, the project scope is outlined and its feasibility is considered. This involves the formation of project objectives and goals, the assessment of risks, and the development of an initial communication plan [35].
- **Elaboration:** The main emphasis is on defining the vision, risk analysis, and management of the project, as well as the subsequent creation of a detailed project plan. In this phase, an architecture is also created [35].
- **Construction:**This is actually the most critical phase in the development of the software system since the actual implementation of the system is done in this phase. These components are integrated among themselves and with other elements to guarantee the fulfillment of the requirements [35].
- **Transition:** It is moved from development to production, where the software products are released to the actual users. Information on feedback and, when necessary, modifications to the tool and parameters is received [35].

Applications of RUP Model: Some of the applications suitable for the Rational Unified Process (RUP) model are [36] [37]:

- Complex Software Development Projects
- Object-Oriented Systems Development
- Requirements Analysis and Documentation
- Iterative and Incremental Development
- Use Cases and Requirements Management

  - Component-Based Architecture
  - Risk Management
  - Collaboration and Communication

The table below illustrates the advantages and disadvantages related to the Rational Unified Process Model:

Advantages	**Disadvantages**
– Decreases the possibility of project failure by enabling ongoing input and development [37]. – The emphasis is on risk minimization and early detection, making sure that possible problems are dealt with as soon as possible [38]. – It can be customized so that it meets the unique requirement of a project due to its versatility and adaptive nature [37]. – Ensures that everyone involved in the project is aware of its current state and direction [36].	– It might take a lot of time and resources to install and maintain the framework because it can be complex [37]. – Experienced developers and project managers are necessary for the successful use of RUP [36]. – Development may be slowed down by the emphasis on documentation and procedure, which can create additional overhead [38].

Table 3.4: Advantages and Disadvantages of the Rational Unified Process (RUP)

- **Agile:** Agile methodologies emerged in the early 2000s due to the rigid and inflexible nature of traditional software development models like waterfall. With the appearance of the Agile Manifesto in 2001, Agile methodology and its foundations were established [39]. This manifesto was authored by seventeen software practitioners who wanted to promote more human-centric and adaptive methods of software development.

  The Agile manifesto has twelve principles and four core values, which primarily put a lot of emphasis on iterative approaches, customer interaction, and flexibility.

  The four core values of the Agile Manifesto are [39]:

  - People and Their Interactions over Procedures and Instruments
  - Functional Software instead of Detailed Documentation

- Consumer Involvement in Contract Negotiations
- Adapting to Change Rather Than Sticking to a Plan

The Agile Manifesto has twelve principles [40]:

- To meet the needs of customers by delivering valuable software on time and consistently.
- Accepting changes in requirements, even if they arrive late in the development cycle.
- Delivering working software frequently in shorter timescale.
- Close collaboration with business stakeholders and developers is required.
- Project should be built around motivated individuals and trust them to get the job done.
- The most efficient and effective way to deliver information is through face-to-face conversation.
- Working software is the fundamental metric of progress.
- The sustainable rate of development should be maintained.
- Continuous focus on technical excellence and good design is required.
- The art of simplicity, or optimizing the amount of effort not done, is crucial.
- The best designs and architectures are produced by self-organizing teams.
- keeping a regular eye on ways to improve effectiveness and making the necessary modifications.

There are many phases in the Agile software development technique, each with a distinct emphasis and set of tasks. The main stages of Agile are described below:

- **Project Initiation:** In this phase the project vision and ROI justification is discussed. The role of DevOps is considered, and the project charter is prepared during project initiation to discuss the project vision and ROI justification [41].
- **Planning:** It is the foundational pillar of the entire project. Project team collaborates with the stakeholders to identify the features and functionalities of the project. This phase involves release planning, backlog creation, and prioritization based on business value and dependencies [42].
- **Development:** During this phase, the development is incremental and takes place in sprints or iterations, which can be enhanced through AI and/or machine learning. The development stage is crucial to the agile approach; it permits the construction of a product with only the bare necessities while implementing the rest of the functionality afterward [42].

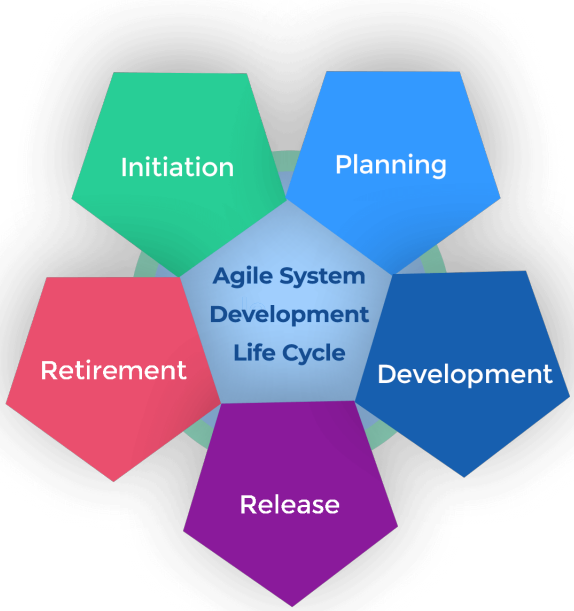

Figure 3.5: Agile System Development Life Cycle [41].

- **Testing:** The primary activities in this phase include quality assurance activities, the generation of technical documentation, and the verification that the software is operating optimally. Unit, integration, and system testing are among the several tests which are included in this phase. [41].
- **Deployment:** During this stage, the program is made available to end users. In addition to making sure the software is installed and set properly, the team needs to teach end users so they can utilize it efficiently [42].
- **Retirement:** When it comes to this stage, it typically involves discontinuing the product, which is usually because newer versions are available or there is a lack of support [41].

Popular Agile Frameworks: Some of the popular agile frameworks are:

- **Scrum:** This is a well-known Agile model that divides development work into sprints, which are two to four-week-long fixed-length iterations. Along with roles like Scrum Master, Product Owner, and Development Team, it includes sprint planning, daily stand-ups, sprint reviews, and retrospectives [43].
- **Kanban:** It is an agile approach that emphasizes reducing work-in-progress and visualizing the process to increase productivity [43]. To monitor work and encourage continuous delivery, it makes use of a Kanban board.

– **Extreme programming (XP):** Through practices like test-driven development (TDD), pair programming, and frequent releases, XP focuses a high emphasis on technical excellence and continuous improvement [43].

The table below illustrates the advantages and disadvantages associated with the Agile software development methodology:

Advantages	Disadvantages
– As it is an iterative approach, teams can react rapidly to shifting demands and market situations, which increases customer satisfaction. [44]. – Agile encourages constant communication and collaboration between stakeholders and team members [45]. – Agile approaches guarantee that functional software is accessible early and throughout a project by providing it in short, incremental cycles [46]. – Software quality is improved overall when problems are found and fixed early on, thanks to continuous testing and integration [47]. – In order to lower the risk of producing a product that does not fulfill user demands, regular feedback loops make sure the changing product satisfies consumer expectations [44].	– It might be difficult to implement agile in conventional, hierarchical businesses since it demands a culture shift toward cooperation and adaptability [48]. – Consistency and dedication to iterative procedures and feedback loops are necessary for implementing agile principles [49]. – If changes are not properly handled, the adaptability of Agile might result in scope creep [47]. – It can be difficult to scale agile for big projects or organizations, and additional frameworks like SAFe may be needed [50]. – There is a chance that as Agile grows, its guiding principles will be compromised, resulting in a "waterfall" method disguised as Agile [49].

Table 3.5: Advantages and Disadvantages of Agile Methodology

3.4 Evolution of low/no-code Development

The origin of low/no-code development (LCND) goes back to the complexity of traditional software development and the simpler, more accessible platforms that exist now. The journey began with traditional methods that, despite their systematic approach to software production, usually resulted in lengthy development cycles. The 1980s and 1990s witnessed a shift towards Rapid Application Development (RAD), which emphasized shorter,

more iterative creation cycles and user interaction, establishing the foundation for accelerating the development process [51].

Parallel to RAD, the invention of Fourth-generation Languages (4GLs) around the same time was a revolutionary move from the more complex progeny languages, processing raw coding to a higher level of abstractions and a more natural language similarity. These language types were proposed as a way to reduce the technological limits of software development from the initial stages [52].

In the late 1990s and early 2000s, contemporary LCND systems emerged, including drag-and-drop interactivity and visual processes. This was a period of principle for these low-code enthusiasts to open up the development of the application to a wider community as expert developers. This was illustrated by tools and platforms including Microsoft Access and Salesforce.com, leading to the organic acceleration of the application development of the LCNDc [53].

The previous decade has seen rapid expansion and adoption of LCND systems by ordinary users, due to the availability of cloud computing in recent years. This advancement has laid the framework for complicated software operating on LCND platforms to expand, enhance its appeal, and become widely used in all sectors [54].

At present, LCND platforms are the leading actors of digital transformation operations, allowing organizations to react swiftly to the market's demands, and innovate without going through the old barriers that used to impede software development. New technologies, including AI and machine learning, incorporated into LCND systems will have a profound impact on software creation making it more accessible and effective [29].

In the future, we anticipate seeing a lot more of LCND combined with new technologies like generative AI, as the area of LCND has a bright future. These expected developments will impact automation, efficiency and software development and function, demonstrating the leadership and implementation of LCND platforms [55].

Chapter 4

State of the Art

In this chapter, we explore the current advancements and trends in low/no-code development platforms and generative AI technologies. We discussed the core features, benefits, and market adoption of these technologies, emphasizing their role in democratizing software development and enhancing efficiency. We also explored the challenges associated with these technologies and how the handling of these limitations can bring numerous opportunities.

4.1 Low/No Code Development

Low-code/no-code development platforms have evolved rapidly since their introduction in the early 2000s. LNCD began in the early 2000s with the emergence of RAD (Rapid Application Development) platforms, which aimed to provide a more visual approach to application development. In 2016, a market research institute created the phrase "low code" to refer to a variety of low code players [56].

Low code development is a visual method to software development that enables faster application delivery with less hand coding. A low code development platform's graphical user interface and drag-and-drop capability automate aspects of the development process, reducing the requirement for traditional computer programming approaches [57]. These low code development platforms make software development more accessible, especially to "citizen" developers, enterprises that use minimal professional coding skills, such as business analysts or project managers [57]. On the other hand, A no-code development platform (NCDP), similar to low-code also lets the users to develop software with drag-and-drop interfaces rather than traditional manual programming. Both low-code and no-code platforms are developed to accelerate development process automation while also providing scalability in these operations [57].

The rise of these platforms is a dramatic shift in the software development landscape

that breaks the barrier of programming complexities, allowing anybody to use already existing reusable building blocks of programs to create an application [58]. These platforms empower both programmers already skilled in techniques and non-technical users via graphical interfaces and pre-written templates, thus lowering the need for extensive hand-coding of code [59].

LCND platforms implement abstraction and visual programming strategies. They provide a visual development environment in which users may drag and drop components to create apps, graphically design processes, and set functions without difficult coding [60]. This template-based approach not only speeds up the development process, but also makes it available to an increasing public, which is a symbol of the generalization of development [61].

Among the core merits of the LCND tools are high productivity, low cost, and making software development accessible to a large number of users, even those with no or little technical skills [57]. Such platforms accelerate the prototyping and deployment of solutions and, therefore, the company is able to respond to market demands and innovation opportunities as quickly as possible [62]. Nevertheless, LCND platforms offer their advantages, yet they are not free from weaknesses. However, critics are of the view that while they offer unique features of accessibility and efficiency, they might lack the high level of flexibility and customization that regular software development options offer. This could cause performance and scalability problems when implemented in complex business scenarios [63].

The integration of AI and machine learning (ML) models into these low/no-code development platforms can create new possibilities for automation capabilities, analytical prediction, and decision-making. While these platforms are still in their early stages, if they continue to advance, they will have a significant influence on a variety of businesses, particularly those that require rapid transformation yet have limited resources for software development [64]. This phenomenon has not only democratized the software development environment, allowing a larger part of the user community to create, but it has also resulted in a more collaborative approach to software development, in which everyone contributes their unique expertise to a given problem. Surpassing phone applications, such as issues related to secure data security, regulatory compliance, and application performance, remains a key priority. However, the basic notion symbolizes the continued role of LCND platforms in future technological growth, as it functions as a force that allows for simplicity of the process, efficiency, and responsiveness to the changes experienced by companies and society [62].

4.2 Low Code Vs No Code

low/no-code development has been leading the current software development and technology area with its functions and capabilities. These are two different approaches of creating software applications and solutions. Low-code is a method of software development that employs tools and technologies to dramatically reduce the number of lines of code required for a similar software solution. No code Development is defined as a software development approach that does not require coding to create a functional software application [65]. Both speed up application development while appealing to a diverse audience and use cases. Let's look at each individual's qualities and differences.

4.2.1 Low Code Development:

Low-code is a technique for developing applications that converts textual coding to visual [58].

Low-code uses a model-driven, drag-and-drop interface approach instead of a technical coding environment. This development method allows developers of all skill levels like beginner, professional developers, subject matter experts, business stakeholders, and decision-makers — to create business applications that can create value [58].

Some of the characteristics of low code development and their benefits are described below:

- **Visual modeling:** Through drag-and-drop functionality and an integrated visualization UI, professional developers can boost their own performance; citizen developers too can build all sorts of applications [58]. By using model-driven technology, the user can graphically present how an application works, simultaneously launching the one-click deployment. The benefits of visual modeling are :
  - Make use of current personnel to develop and deliver apps more quickly.
  - Encourage participation in development from departments other than IT
  - Create a range of alternatives without spending a lot of money .
  - Improved cross-functional team cooperation and decision-making
  - Allow expert developers to focus on more ambitious and intricate projects.
- **Reusable components:** Users can create cross-platform apps using pre-built logic, connections, templates, and modules. Skilled developers can alter and expand the functionality of low-code application components. The benefits of reusable components are as follows [58]:
  - Accelerate development with increased effectiveness.

  - Depend on reusable components with performance and security testing completed beforehand.
  - Create apps that are more scalable and consistent.
  - Extend program features as soon as necessary

- **Collaboration tools:** Capable of collaborative development with integrated features for user stories, revision tracking, feedback loops, chat, and more. Low code keeps everyone using the same programming language since it is visual. Benefits are [58]:
  - Divide departments to promote a more robust business-IT collaboration.
  - Encourage enhanced inter-organizational communication.
  - Simplify the development process and minimize rework.

- **Scalable environments:** As consumer wants and company requirements change, developers can swiftly implement new apps and enhance current ones [58]. Scalability for both run-time and development-time on demand, flexibility, and support for continuous delivery are features of a cloud-based low-code platform. Benefits are:
  - Creating scalable, easily maintainable systems with cloud-native architecture.
  - When required, assisting in making quick adjustments.
  - Continuously improving end user experiences by means of engaging mobile, web, conversational, or immersive experiences.
  - Maintaining an adaptable organization in the face of change.

- **Data integration:** Tntegration of data and logic can be done securely from any source, system, or service. It can be done by either using APIs and connectors that are pre-configured or experienced developers can create custom integrations [58].
  - More than 30% of the developers' time can be reclaimed.
  - Creating and managing larger-scale systems.
  - Finding and sharing data between projects and teams with ease.
  - Easily accessing company data to create reusable components and microservices more quickly.

- **Application lifecycle management:** Integrative low-code platforms combine Agile development principles and DevOps technologies to support all stages of the application development lifecycle with tools that accelerate project management, requirements management, version control, testing, deployment, and more. Benefits [58]:
  - Moving apps through every stage of their lifetime with ease: Conceptualization, creation, evaluation, implementation, and management.
  - Automating and abstracting to accelerate every stage for faster delivery.
  - Giving teams the freedom to operate independently and iteratively.

4.2.2 Use Cases:

For companies looking to adapt digitally, cut costs, and quickly create applications, low-code development has become an excellent option. Low-code development facilitates the rapid and effective creation of applications by both developers and non-developers due to its visual development tools, drag-and-drop capability, and low coding requirements [66]. The most common business use cases for low-code development include the following:

- **Legacy application modernization:** Modernizing and updating outdated systems is a common use case for low-code platforms. This entails, without beginning from scratch, converting outdated programs into more contemporary, effective, and scalable solutions [66].
- **Process automation:** Companies automate and simplify complicated business operations with low-code platforms. Increasing overall efficiency and decreasing the need for manual intervention, might involve anything from straightforward procedures to complicated operational systems [66].

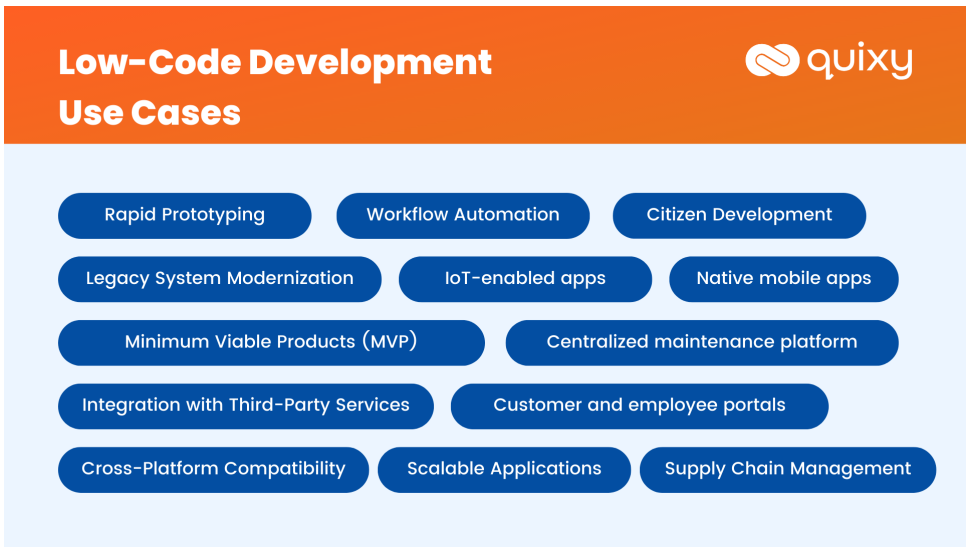

Figure 4.1: Low code development use cases [67]

- **Digital experience solutions:** In order to improve the customer experience, this includes developing online portals, mobile applications, and other digital interfaces. Low-code platforms, which frequently have responsive designs for cross-platform compatibility, let companies create and implement these solutions fast [66].
- **Rapid prototyping and development:** Rapid development and prototyping are made possible by low-code platforms. Companies can quickly develop and refine new concepts and apps by utilizing low-code platforms. This flexibility is essential in a fast-moving corporate environment, where requirements can change abruptly [66].

- **Data integration and management:** Effective data management and integration of several data sources are possible with low-code platforms. Businesses that need to significantly combine vast volumes of data from many sources may find this very helpful [66].
- **Custom business applications:** These platforms enable businesses to develop unique apps that are suited to their own business requirements and workflows. This can include consumer applications, as well as internal tools for staff use [66].
- **Customer relationship management (CRM) and sales automation:** Low code platforms can be used by businesses to improve customer engagement and sales productivity by automating sales processes and developing or upgrading CRM systems [66].
- **Supply chain management:** Supply chain processes, including distribution and transportation, as well as inventory management, can be made more efficient with low-code [66].
- **Reporting and analytics:** Applications for improved reporting and analytics may be developed by businesses, allowing them to make data-driven choices more quickly and effectively [66].
- **Compliance and risk management:** Solutions to manage risk assessment and management, as well as compliance with various requirements, can be developed using low-code platforms [66].

4.2.3 Challenges and Limitations:

Low-code development is becoming more and more popular, as it can accelerate software development, save costs and promote cooperation. Businesses should be aware of the challenges and limitations associated with this technique before using it. These are some of the primary challenges and limitations that come with low-code development.

- **Limited Flexibility and Customization:** The limitation of low-code platforms lies in the predefined templates and components that limit the level of applications to which they can be customized to meet specific business requirements. Finding a suitable balance between speed and individualization is one of the most important things.
- **Integration Complexity:** Integration of applications with existing systems and databases is demanding, as these systems are becoming more and more complex. In order to achieve effective integration with older systems as well as the third-party apps, low-code platforms should provide powerful integration capabilities.

- **Security Risks:** The security of low-code applications should be a top priority for the platforms, as they ensure that the applications are safe from any vulnerability and fall within the regulations of the industry. Inability to handle security correctly can lead to data leaks and legal implications.
- **Scalability Challenges:** Low-code platform applications need to be adaptable enough to meet evolving business needs and handle increasing loads. To achieve scalability, extensive planning and architectural work are required.
- **Dependency on Skilled Developers:** Low-code platforms are meant to be user-friendly, but in order to solve complicated issues and complete advanced activities, experienced developers are required. Hiring a competent team that can fully utilize low-code development's strength and potential is something that organizations should focus on.
- **Vendor Lock-in:** Most low-code platforms do not allow access to the source code, making it difficult to migrate programs to different platforms. This limitation will force the business to continue working on the same platform or rebuild it from scratch, which will result in a substantial increase in time and cost.
- **Team Collaboration:** The shortcomings of low-code platforms often are that they do not provide the appropriate platform for team cooperation. The lack of standard mechanisms for code reviewing, as well as pull requests, can cause conflicts or overwrites with multiple people working on the project, and this may be the cause of the issue of maintaining coherence.
- **Developer Experience and Limitations:** At this early stage, the user experience of most existing low-code and no-code products is typically not up to the expected level of quality. This can severely reduce development productivity, since developers may have to choose between depth of feature and convenience of usage.

To deal with these problems, companies should consider adopting a hybrid development model, establishing a dedicated team responsible for integration solutions, assigning an unambiguous security position, establishing a scaling plan, and conducting continuous staff training. Moreover, organizations would be responsible for a good appraisal before adoption and ensuring the platform is scalable and meets the long-term needs of the entity.

Some of the most popular low code development platforms are [68]:

- OutSystems
- Appian
- Microsoft PowerApps
- Salesforce Lightning

- Mendix
- Zoho Creator
- Quickbase
- Creatio

4.2.4 No Code Development

With the use of the no-code software development process, anyone without any prior programming experience or knowledge may develop and distribute apps without writing any code. By utilizing a robust visual programming environment with drag-and-drop functionality and pre-existing components, no-code development gives users the opportunity to construct web and mobile applications without ever writing a line of code [69].

Some of the common functionality that no code tools provide is as follows:

- A good, user-friendly interface with a drag-and-drop editor, visual workflow, and other capabilities that allow users to develop apps without writing any code [69].
- A collection of already assembled components, such as forms, tables, buttons, and other user interface elements commonly used in software systems [69].
- Users can use website, mobile, and workflow templates to start using their no-code apps [69].
- Integration with other tools and services, such as payment gateways, email marketing platforms, and customer relationship management (CRM) systems, enables users' apps to connect to other systems and data sources [69].
- A variety of deployment choices, enabling customers to roll out their apps on other platforms, the web, or mobile devices [69].

4.2.5 Use Cases:

No-code development platforms have been more popular in the last several years, allowing companies and people to develop software applications without having to write traditional code. These platforms meet a variety of demands and circumstances by providing a range of use cases. The following are a few typical applications for no-code development platforms [70]:

- **Rapid Prototyping:** Without requiring extensive coding skills, users may construct and validate application prototypes using no-code development platforms. This case study has several benefits, particularly when it comes time to design and you require

prototypes based on user feedback [70]. Programming visual tools and pre-built modules speed up the approval and prototype process, cutting down on time-to-market and length accordingly.

- **Internal Tools and Automations:** Organizations frequently need specialized internal tools and automated processes to optimize operations and improve operational efficiency. No-code platforms enable non-technical people to create solutions without relying on IT resources [70]. From staff onboarding systems to data gathering forms and workflow automation, no-code platforms allow the building of customized apps that answer particular business objectives.
- **Cost-Effective Solutions:** It may be expensive and resource-intensive to develop apps using standard coding techniques, particularly for startups and small enterprises. By eliminating the requirement for specialist developers and cutting down on development timeframes, no-code platforms provide a more affordable option [70]. Thanks to this use case, businesses may construct useful apps without having to pay hefty costs for developing proprietary software.
- **Reduced Maintenance Costs:** Conventional software programs may be difficult and expensive to maintain and update, sometimes needing specialist knowledge and resources. Due to its visual design and pre-built components, no-code systems often have cheaper maintenance expenses. The difficulty of updating and improving custom-coded apps is decreased, as is the overhead involved in doing so.
- **Agility and Iteration:** No-code platforms provide for more agility and iteration in application development. Because modifications may be made fast and without considerable coding, firms can swiftly adjust to changing requirements, user input, or market situations [70]. This use case provides a more iterative development strategy, allowing applications to be improved and refined on a constant basis.
- **Easier Maintenance and Over-the-Air Updates:** Maintaining and upgrading programs becomes easier with no-code platforms. Many systems support over-the-air (OTA) updates, which allow upgrades to be sent to deployed apps without the need for manual intervention [70]. This use case streamlines the update process, ensuring that apps are up to date with the newest features and advancements.
- **Order management:** No-code platforms may be used to create order management systems that help organizations optimize their order processing and fulfillment procedures. These apps can interface with a variety of systems, including inventory management, shipping, and payment gateways, to provide a consolidated order management solution [70].
- **Employee directories:** Employee directories are a common use case for no-code platforms in business. Companies may use these platforms to create customized directories that include employee information, organizational hierarchies, and search

functions [70]. Employees may instantly modify and access these directories, which improves internal cooperation and communication.

4.2.6 Challenges and Limitations:

Much like low-code development platforms, no-code development platforms (NCDPs) have revolutionized software development by enabling non-technical individuals to write programs using graphical user interfaces and configuration instead of traditional coding [71]. Though these platforms save development times and improve accessibility, they have a number of drawbacks that may restrict their applicability in settings that are more complicated or scaled. Some of the challenges and limitations associated with no code development tools are described below:

- **Limited Customization:** Most no-code platforms come with ready to use templates along with limited customization options making it hard to build robust applications which cater the unique business requirements [70].
- **Scalability Concerns:** No-code platforms may be not designed for managing large amount of data or number of users so it would be difficult to develop applications that the broader audience uses them. Scale to handle more traffic from users or complex functionalities are still a limitation that many no-code tools have [70].
- **Vendor Lock-In:** The no-code platforms are proprietary, thus, if a project was build with a certain platform, it can be hard to migrate or have another platform. It may reduce the agility and portability of programming and hurt the chances to apply new gadgets or methods [71].
- **Limited Integration and Collaboration:** No-code platform may have a drawback of integration of other systems and collaboration between technical and non-technical personnel too. This can obstruct the smooth data flows and automation of workflow among various tools [71].
- **Data Security and Compliance:** It may be difficult to ensure data security and regulatory compliance while using no-code applications. Organizations should exercise extreme caution when it comes to access restrictions, encryption, and data protection to prevent unauthorized access or data breaches[71].

Some of the popular no-code development platforms are:

- Bubble
- Airtable
- Webflow
- Zapier

- Quickbase
- FlutterFlow

4.2.7 Difference between low code and no code development

The following table highlights the key distinctions between low-code and no-code development tool:

Criteria	Low-Code Development	No-Code Development
Customization	Allows for some custom coding	Does not require any coding
Target Audience	IT professionals with coding skills	Non-technical users without coding knowledge
Flexibility	More flexible than no-code, but less than traditional coding	Limited flexibility due to rigid templates
Security	Requires some attention to security, but can be customized	Security can be a concern due to lack of control
Cost	Can be more expensive due to the need for coding talent	Generally more cost-effective due to ease of use
Integration	Can be integrated with other systems and customized	Limited integration abilities due to rigid templates
Scalability	Better positioned for scalable solutions	Limited ability to scale complex apps
Examples	Appian, Mendix, Zoho Creator	Webflow, Bubble, Glide, Zapier, Adalo, Thunkable

Figure 4.2: Low Code Development vs No Code Development [72]

4.3 Core Principles of Low/No-Code Development Platforms

Traditional application development always requires skilled coders and is typically delayed until the department of those coders in IT approves any creation or modification of apps [73].

Low-Code Development Platforms (LCDPs) and No-Code Development Platforms (NCDPs) utilize visual programming, model-driven development, and automatic code generation

concepts. These platforms are generally intended for users who are already familiar with their company's department's technological stack, protocols, and workflows, independent of their programming expertise level. Individuals with and without prior experience will be required to participate under the concept [73].

The core principles are described below:

- **Visual Programming (Drag-and-Drop Interfaces):** Drag-and-drop interfaces reflect a paradigm shift in software development, with a focus on visual connections rather than textual code. This interface allows users to create programs by dragging and dropping objects such as texts, buttons, and connections onto a grid and organizing them to construct a working user interface and workflow [73] [74]. This visual mode converts fundamental coding logic into graphic blocks that are much easier to grasp and work with, lowering the technical barrier to app creation. Not only accelerates workflow and core tasks, but also democratizes software development, making it available to a wide range of stakeholders, including business analysts, designers, and subject matter experts [73].

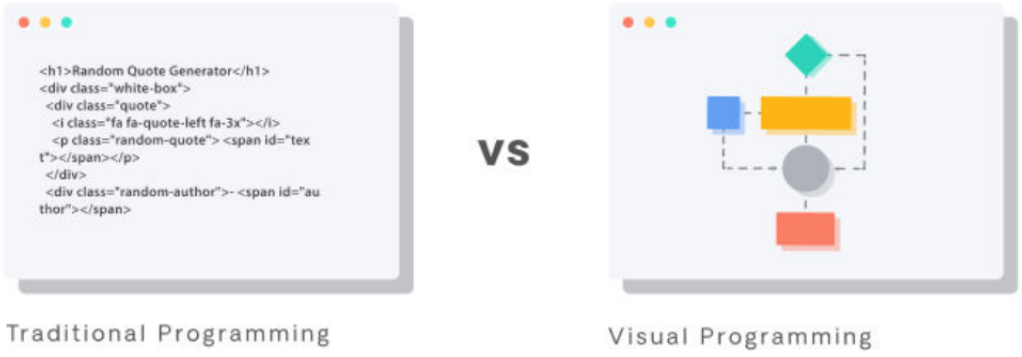

Figure 4.3: Traditional Programming vs Visual Programming [74]

- **Model-driven Design:** Model-driven design is the basic idea behind LCND platforms, which uses high-level concept models to define an application's design by categorizing its structure, behavior, and interaction of application components. This solution automates a substantial portion of the app production process; the system converts technical models into operational apps using a set of predefined rules and a library of templates. The model-based architecture naturally encourages quick and repeated simulations and versions of the application, as any changes to the model are immediately reflected in the code without the need for manual intervention [58].

  This flexibility feature ensures that app development is done quickly and easily and

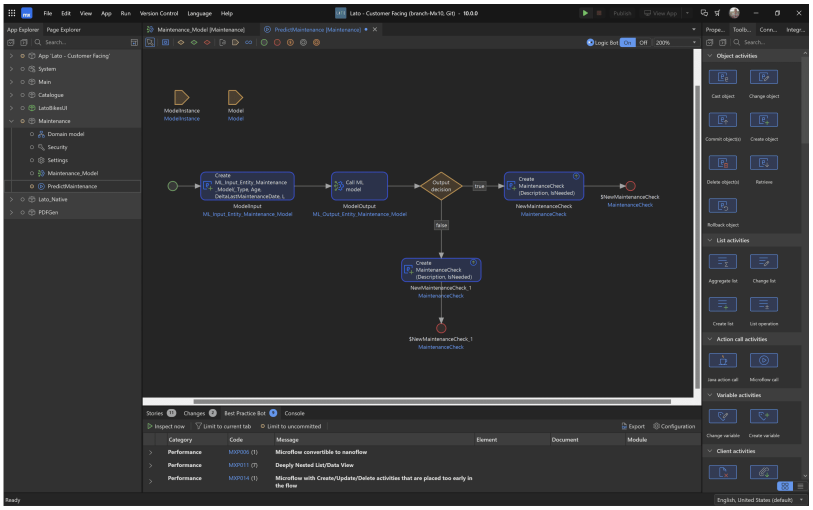

Figure 4.4: Model Driven Development [58]

that any changes based on feedback and growing needs are properly addressed. On the contrary, model-oriented design integrates technological and syntactical aspects into the design solution in a smooth manner by focusing on the domain layer, which is the logical part of software development. In addition to making the development cycle more beautiful and deliberate, the idea also lends the application design a more focused and strategic approach [58].

- **Automatic Code Generation:** Automatic code creation in low/no-code Development platforms simplifies software development by eliminating the tediousness of manual coding. These systems, such as OutSystems and Mendix, start code creation via a visual interface, turning user actions into executable code using model-driven concepts [75]. Using pre-defined templates and libraries, users, including citizen developers, can focus on functions while the platform addresses technical needs such as reusability and usability. Automation empowers developers by making app development more accessible while also improving code quality, maintainability, and processes, allowing for the effective and easy creation of software in a dynamic digital arena [73].

Low-Code/No-Code (LCND) development is a more efficient way to build software than traditional coding. Here is an overview of the key steps involved.

- **Defining the needs and desired outcome:** To begin low/no-code software development, the first step is to determine the project's business requirements and end goals. Recognizing the problem to be solved, the possible users and the data needed for the required functionalities is critical at this stage [73].

- **Drawing a Business Process or Workflow:** low/no-code Business Process Management (BPM) and development technologies are used to establish and capture the desired procedures and workflow. Dividing the application into modules based on its intended use, and then generating these modules as independent components. These modules can collect data, initiate actions or events, and combine to deliver all desired results [73].
- **Testing and Deploying the Project:** After developing the application using low/no coding platforms, the next step is to test and deploy the product. low/no-code systems that eliminate complexity from the back end, facilitating testing by IT specialists and beta testers. Once the feedback-based adjustments have been implemented, the application may be made available to the general public [73].

4.4 Current State & Trends of low/no-code Development

With the rise of low-code and no-code development platforms in recent years, the landscape of software development has undergone significant shift [76]. The process of developing software has been transformed by these newly developed tools, which enable both technical and non-technical people to construct completely working software applications without writing a significant amount of code.

Low-code and no-code development has evolved to meet the demand for quick, efficient, and accessible software. Customers from many business sectors perceive these platforms as a strategic asset that enhances prototyping, streamlines procedures, and empowers citizen developers [76]. Understanding the current and future of low-code and no-code development requires understanding the key trends, developments, and challenges that shape this dynamic ecosystem. Some of these patterns and present conditions are mentioned below:

- **Increasing Adoption and Mainstream Acceptance:** A major factor influencing low-code and no-code development is the growing recognition of the platforms as more people start using them. According to the 2022 survey by Mendix, 94% [76] of enterprises from different sectors used low-code solutions in comparison to 77% in 2021 [76], as stated earlier. Furthermore, 69% of the respondents acknowledge that their companies need to adapt low-code options rather than utilize them only for temporary measures [76]. This transformation of view is an acknowledgment that low-code and no-code development approaches become a strategic competitive advantage to the business.
- **Expanding Use Cases and Industry Adoption:** Low-code and no-code development is no longer restricted to specific use cases or only for small-scale companies [76]. These platforms are being deployed across numerous fields, such as health care, e-commerce, education, and public institutions. Employers take advantage of these

tools to resolve issues related to logistics, the supply chain, and customer support [77].

- **Increasing Integration and Collaboration:** With the ongoing development of low-code and no-code tools, collaboration and exchange become increasingly crucial [77]. Platforms are no longer confined to very simple functionality but also provide more advanced alternatives such as real-time or seamless interaction with current systems, real-time collaboration, and version control tools. The company's continuous success can be due to larger businesses' demand for a platform that allows for increasing levels of cross-departmental connectivity [78].

- **Emergence of Citizen Developers:** The introduction of low-code and no-code development has contributed to the concept of "citizen developers," which are individuals who can build apps themselves utilizing these platforms without prior coding experience [77]. Such movement will continue since technology increases user base diversity and allows individuals to participate in software development while also meeting their commercial objectives [78].

- **Advancements in AI and Machine Learning:** The current trend is that low-code and no-code development platforms integrate AI and ML to enable their users to create more complex apps without having specialized technical knowledge. This trend is anticipated to be faster as advanced technologies become more affordable and user-friendly [77].

  The evolution of AI-integrated platforms has greatly aided the expansion of the worldwide low-code and no-code sectors that were valued at $13.2 billion in 2019 and growing at a rate of 21% per year [79]. AI-based support in these techniques is expected to contribute to market growth, potentially increasing the number of users from $50 billion to $167 billion over the next four years [79]. This integration aims to boost the influence of citizen developers on typical coding activities while also improving performance. However, the language barrier between natural language and generative AI can impede the democratization of software creation, necessitating instructional support [79].

- **Evolving Pricing and Deployment Models:** As the low-code and no-code development markets grow, pricing and deployment strategies will likely alter [78]. Platforms may provide more flexible and scalable pricing choices to meet the expanding number of users and complexity of applications. Furthermore, cloud-based deployment and subscription-based models are expected to become increasingly common, giving enterprises greater flexibility and scalability [77].

To summarize, the current state of low-code and no-code development is characterized by greater popularity, wider use cases, improved integration and collaboration capabilities,

the rise of citizen developers, breakthroughs in AI and machine learning, and flexible pricing and deployment options. These advancements highlight the platforms' transformative impact on the software development landscape, allowing both technical and non-technical users to create new solutions with more efficiency and effectiveness.

4.5 Existing Low/No-Code Development Platforms

In this section, we are going to explore some of the major players in the low-code and no-code development arena that are currently shaping the market. We'll look at platforms like OutSystems, Mendix, and Joget DX. Our discussion will dive into what makes each platform tick—their key features, how they operate, and what sets them apart in terms of strengths and weaknesses. This exploration will help us understand how these platforms respond to different needs and operational requirements.

4.5.1 OutSystems

OutSystems is a leading low-code application development platform with a strong presence in the enterprise sector. OutSystems is designed to facilitate quick application development and delivery, allowing enterprises to build and launch sophisticated applications with little hand-coding [59]. The platform's novel approach has received praise from industry experts and research groups, establishing it as a leader in the low-code development scene.

Visual Development Environment: OutSystems' major component is its sophisticated visual development environment, which abstracts away the complexities of traditional coding. A drag-and-drop interface, pre-built components, and visual modeling tools can help developers design and create apps quickly. This visual method accelerates the development process, reduces human mistakes, and allows developers to focus on business logic and application functionality rather than low-level code [59].

Model-Driven Architecture: It implements model-driven design, which views application infrastructure through the lenses of logical, visual, and data layers. These designs offer increased flexibility, easily accessible scalability, and the chance to reuse components [80]. Developers may describe data models, business processes, and integration points, and the platform leverages visual tools to generate the underlying code. This eliminates the need for developers to start from scratch, ensuring that the output is consistent and follows best practices [66].

Cloud-Native and On-Premises Deployment: In order to satisfy the diverse demands of organizations, as well as location-independent requirements, OutSystems offers both cloud-native and on-premises deployment options. The foundation of cloud-native de-

ployment is containers, which combine with orchestration technologies to enable resilient environments, automated upgrades, and enhanced load management [81]. Alternatively, businesses can opt to integrate OutSystems into their infrastructure, which would enable them to maintain control over any other service provider. All the while, they would capitalize on OutSystems' extensive feature set for solution development.

Enterprise-Grade Capabilities: It is designed to address the performance, security, and scalability problems with corporate applications, which need a significant amount of computing power. It offers extensive security features that others would not, such as data encryption, role-based access control, and compliance with industry standards like HIPAA and GDPR [82]. Additionally, pre-built interfaces for the APIs and integrated data sources are there, along with functionality from other systems. The same technology is also used to smoothly connect the bespoke integrations.

Ecosystem and Community: A vibrant ecosystem and community of developers, partners, and clients have been established by OutSystems. Developers may exploit pre-existing resources and shorten development cycles by taking advantage of the extensive selection of reusable components, templates, and connectors available in the OutSystems Forge [83]. In addition, the OutSystems Community offers a cooperative setting for information exchange, best practices, and assistance, encouraging ongoing education and creativity.

Continuous Innovation and Roadmap: As part of its ongoing commitment to innovation, OutSystems often updates its platform with new features and improvements. In order to maintain the platform's leadership in low-code development, the business bases its product roadmap on input from customers, market trends, and upcoming technologies [84]. Improved DevOps integration for faster application delivery, improved mobile app development tools, and more artificial intelligence (AI) and machine learning (ML) capabilities are some of the most recent advances.

Platform Architecture: The OutSystems platform's architecture is built for high-performance low-code development, allowing programmers to easily create a variety of applications. The platform provides several deployment options for various corporate infrastructures and tech stacks [85]. OutSystems provides a layered environment that facilitates the development of applications that modify internal business processes, expand systems of record, restructure important business systems, and construct web and mobile applications. Key

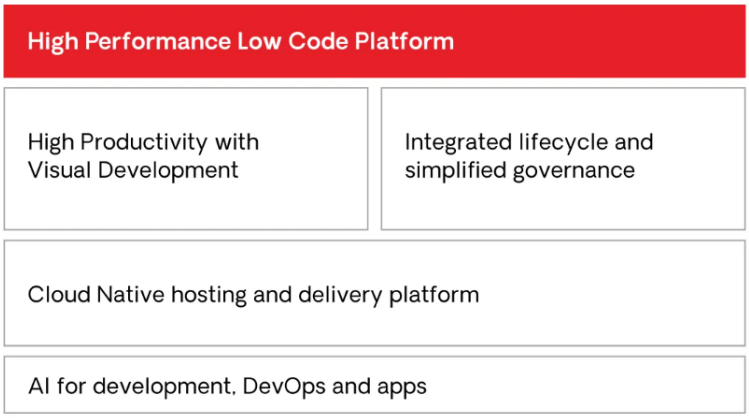

Figure 4.5: OutSystems Platform Overview [85]

Components of OutSystems Architecture are:

- **Cloud-native Architecture (ODC):** Developers may create and launch apps in a cloud environment using OutSystems Developer Cloud (ODC), a cloud-native deployment solution [85].

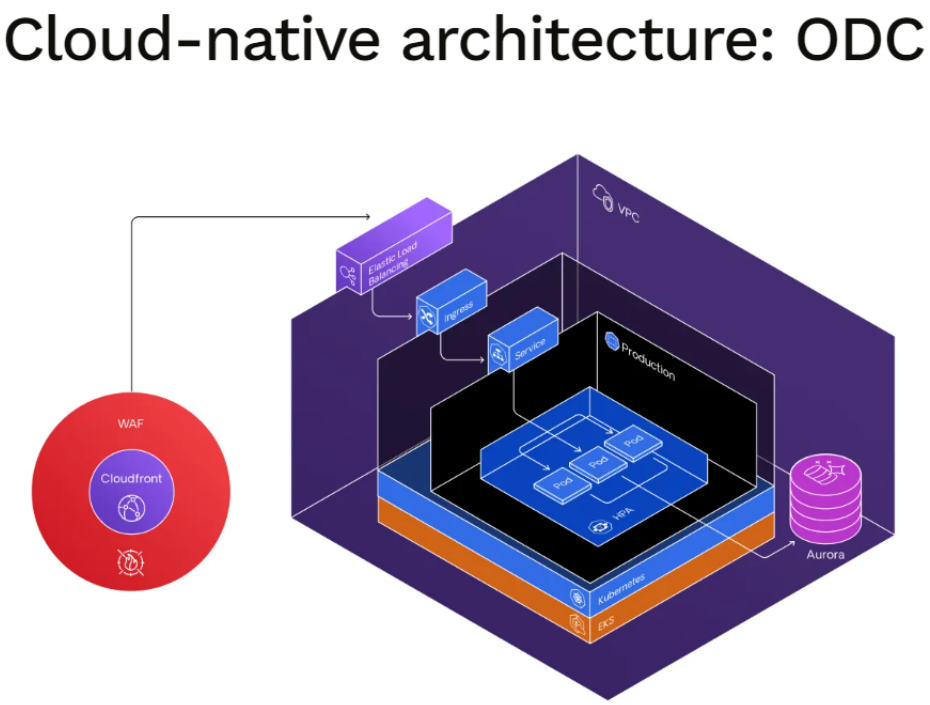

Figure 4.6: Cloud-native architecture: ODC [85]

- **Cloud-ready Architecture (OutSystems 11 Cloud):** Building extensible apps that can operate on a typical.NET application server is made easier with the common architecture offered by OutSystems 11 Cloud, a cloud-ready platform [85].
- **Key Tools and Environments [85]:**

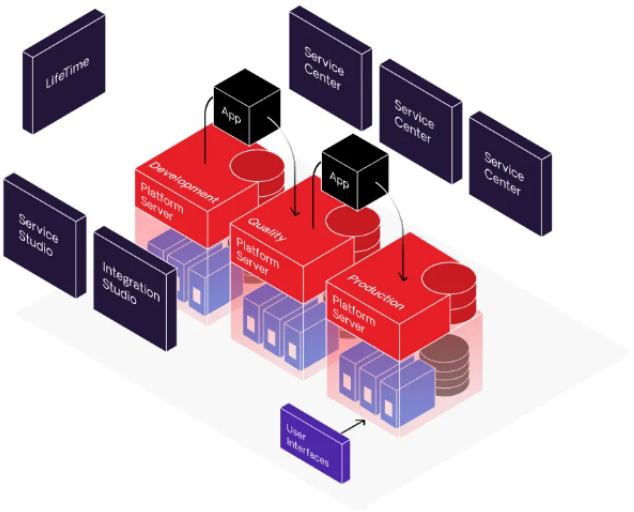

Figure 4.7: Key Tools and Environments [85]

  - **Service Studio:** It's the environment for creating various part of the application stack.
  - **Integration Studio:** It is for creating components to extend the platform and integrate with third party applications.
  - **Platform Server:** This is the core server component that generates, optimized, complies and deploys applications.
  - **Builders:** SaaS tools to simplify the complex development activities like creating workflow apps and integrating with external systems.

- **Management Consoles [85]:**

  - **Service Center:** It manages the operational aspects of the environments.
  - **LifeTime:** This enables centralized management of development environments

- **Infrastructure and Security:** Outsystems cloud uses AWS data centers with dedicated virtual machines and database instances. For security measurements, it includes AWS Security Groups, Web Application Firewall(WAF), and high availability options [85]. Customers has the option to upgrade to a high-compliance Outsystems Cloud For enhance security and compliance measures.

Although the OutSystems platform has many advantages, users and organizations should be aware that it also has limitations and obstacles. These elements may have an effect on the user experience as a whole, integration capabilities, and the development process. Now let's examine the drawbacks and restrictions of the OutSystems platform.

- **Licensing Model and Cost Constraints:** The OutSystems licensing and application object fees could probably be very challenging, for example, for enterprises that need their business to expand. It is often difficult to justify the expenses of future developments due to high user license costs and user number restrictions [86].
- **Migration Challenges:** There might be some challenges involved such as the migration of data or applications from other platforms to OutSystems as there aren't many language importers available in it. This makes the process of migrating the existing applications or the data to the OutSystems platform smoothly challenging [86].
- **Documentation Quality:** There are a couple of issues with the OutSystems documentation, such as empty pages and duplicate or missing content, which are particularly in the best practices' standards. Such bugs make it difficult for developers to take full advantage of the platform, resulting in chaos and ineffectiveness [86].
- **Limited Database Integration:** OutSystems' incompatibility with numerous databases, including PostgreSQL and Google Sheets, may limit the flexibility and compatibility of programs built on the platform. Organizations that rely on certain database systems may have issues as a result of the absence of native integration, potentially reducing the platform's usefulness for their needs [86].
- **Community Support and Isolation:** Because the OutSystems platform has a unique community forum for member engagement, users might not be able to get help outside of this official channel. When attempting to troubleshoot issues or engage with a wider community, some users may find it challenging to access forums and find solutions. This might have an impact on the project and their experience [86].
- **Development Process Overhead:** Small changes, e.g. adding a feature or modifying the user interface, result in more complexity because the whole program has to be published again. This can lead to disruptions in the smooth working of development processes and make them inflexible [86].

Although the OutSystems platform has several benefits, such as low-code development, scalability, and integration possibilities, users should be aware of its drawbacks and limits.

Comprehending these concerns may aid businesses in making well-informed selections and adeptly navigating the complex nature of software development through the utilization of the OutSystems platform.

4.5.2 Joget DX

Joget DX is a cutting-edge open source application platform built to speed and simplify digital transformation. Joget DX is a simple, adaptable, and open platform that combines the best of business process automation, workflow management, and low-code app development. Business and technical teams may collaborate to rapidly build full-fledged business applications visually, from anywhere, at any time [87].

Some of the features of Joget DX platform are as follows [87]:

- Non-programmers may create and manage apps from anywhere at any time using a web-based visual approach..
- Shortens the time to market from several months to a few weeks or days [87].
- Joget DX-built applications are ready for the cloud and mobile devices.
- It offers plugin architecture for extension and APIs for integration.
- It also has "App Store" for enterprise apps – Joget Marketplace

It emphasizes the following key focus areas [87]:

- **Progressive Web Apps (PWA) and User Experience (UX):** Joget DX encourages Progressive Web Apps (PWA) development in order to provide apps that function seamlessly and provide an app-like experience regardless of the device used. PWAs are essential in current web applications since they enable offline capabilities, load quicker, and allow for platform-independent interfaces. This underlines the significance of user experience, which ultimately contributes to user approval [88].
- **DevOps and Application Performance Management (APM):** Joget DX makes use of DevOps technics such as continuous integration and delivery (CI/CD) and enables enterprises to do rolling updates quickly and manage the applications better. Moreover, with Application Performance Management (APM), errors are detected and fixed to guarantee the speed of service levels to meet clients' expectations [88].
- **Artificial Intelligence (AI) and Smart Decisions:** Joget DX integrates artificial intelligence (AI) into applications to enhance decision-making processes. AI in Joget gives smart systems the capacity to make automated judgments on complex processes employing data-driven insights, resulting in more effective and efficient operations [88].

Figure 4.8: Joget DX focus areas [87]

- **Extensibility via Add-On Builders and Enhanced Workflow:** Extensibility is one of the key features of Joget DX, as it gives users the ability to add builders in order to tailor and extend the features of their applications. The enhanced capabilities of workflows imply that, no matter how complicated a business may be, it can be automated and controlled smoothly [88].

Platform Architecture: The Joget DX platform architecture is intended to offer a flexible and scalable environment for developing and delivering corporate applications. It is an open-source platform that employs a no-code/low-code approach, allowing users to develop apps without significant programming skills.

Key Components of Joget DX Architecture:

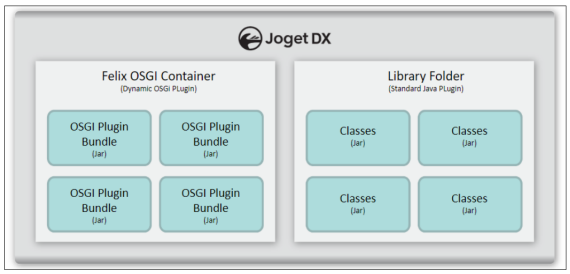

Figure 4.9: Joget DX Pluggin Architecture [87]

- **Plugin Architecture:** Standard Java Plugin and Dynamic OSGI Plugin are the two forms of plugin packaging that Joget DX supports. Because of this architecture, developers may add new plugins to increase the functionality of the platform [87].
- **Embedded Git Integration:** For cooperative development and deployment, Joget DX works with Git. Due to this connection, deployment procedures may be automated and several users or teams can work on the same application at once [88].

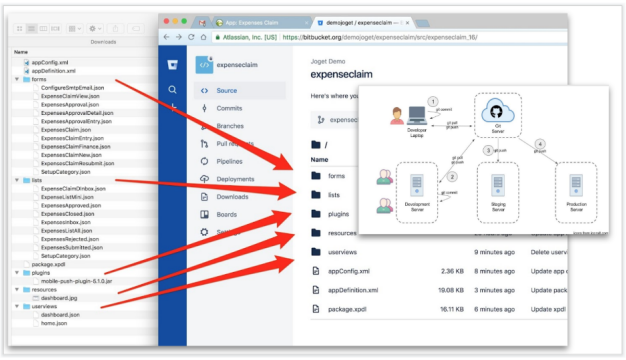

Figure 4.10: Joget DX Embedded Git Integration [88]

- **Artificial Intelligence (AI) and Smart Decisions:** The platform contains plugins for process choices and decision making, which may be linked to process routes. It also includes a no-code TensorFlow AI plugin to integrate pre-trained AI models into applications [88].

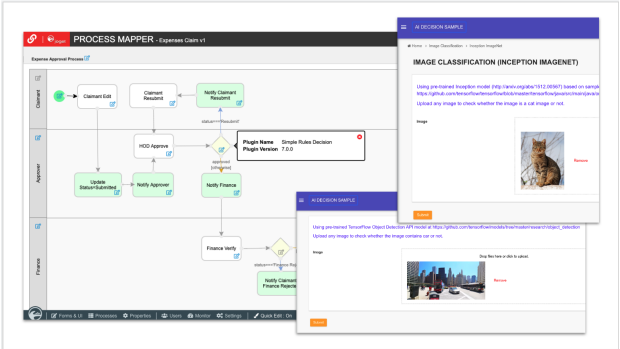

Figure 4.11: Joget DX Artificial Intelligence (AI) and Smart Decisions [88]

- **Extensibility via Add-On Builders and Enhanced Workflow:** Pluggable Add-on Builders are now supported by Joget DX, enabling the platform to be expanded with other visual builders. Additionally, improved workflow plugins are included in this feature so that workflow forms and behavior may be customized [88].

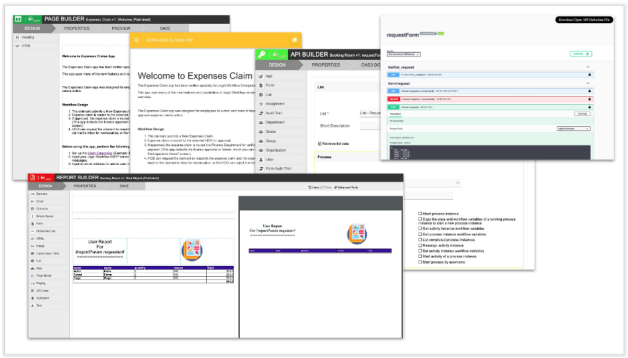

Figure 4.12: Joget DX Extensibility via Add-On Builders and Enhanced Workflow [88]

- **Progressive Web Apps (PWA) and User Experience (UX):** The platform supports Progressive Web applications (PWA) features including offline support and push notifications, and it comes with a new default Progressive style for applications that was inspired by Google's Material Design [88].

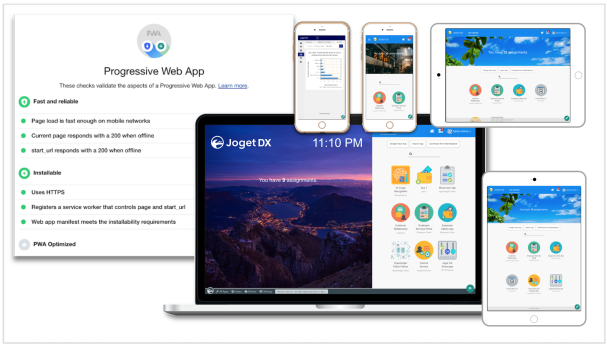

Figure 4.13: Joget DX Progressive Web Apps (PWA) and User Experience (UX) [88]

- **DevOps and Application Performance Management (APM):** There are built-in application performance management (APM) tools in Joget DX that provide real-time warnings and automated monitoring of system and application performance [88].

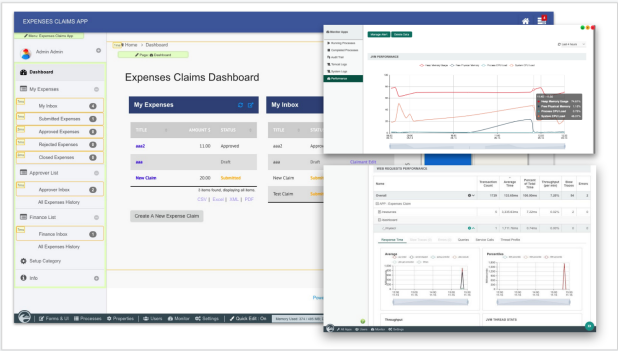

Figure 4.14: Joget DX DevOps and Application Performance Management (APM) [88]

Limitations: Some of the challenges and limitations associated with Joget DX platform are as follows:

- **Customization Complexity:** Even though Joget DX has a great deal of customization options, customizing apps to meet the demands of a particular project might require a little expertise in technology, particularly when adding extra features or creating intricate processes that go beyond the platform's built-in capabilities. For users with complex and specialized needs, this could be difficult [88].
- **Database Access Limitations:** Due to security concerns in a shared server environment, database access is restricted for users who are subscribers to the Starter and Standard On-Demand subscriptions. Users must upgrade to the On-Demand Dedicated Server package in order to access the database, suggesting a constraint in the lower-tier subscription options [89].
- **App Limitations:** Users with different subscription packages can only create a certain number of apps or workflows on the Joget DX platform. For example, users with the Standard package can only create 20 apps, while users with the Starter package can only create 10 apps. This could limit the scalability and flexibility of application development for users with large requirements [89].
- **Subscription Model Constraints:** Users may be limited in terms of scalability, access to advanced functionality, and customization choices depending on their selected plan by the platform's subscription model, which offers many packages with differing features and constraints. This may reduce the platform's usability for users with a range of demands and financial situations [89].
- **Learning Curve:** Moving to a new platform might be difficult, especially for users used to traditional development approaches, even if Joget DX is recognized for its ease of use, simplicity, and low learning curve. A little initial investment in familiarization and training may be necessary to fully grasp the possibilities of the platform and adjust to a low-code environment [90].

While there are many benefits to Joget DX as a flexible low-code platform, such as its rapid application development, ease of use, and extensive feature set, users may run into issues with subscription-based limitations, app limitations, database access restrictions, the learning curve that comes with switching platforms, customization complexity, and customization complexity. Solving these issues could improve the platform's usability even further and make it more appealing to a wider range of users.

4.5.3 Mendix

Mendix Inc. developed the Mendix Platform, an all-inclusive low-code development platform that lets companies design, integrate, and deploy large-scale applications with little to no code [91]. Its visual development platform allows users to create, build, test, and deploy programs without requiring extensive programming knowledge. Mendix has been acknowledged as a leader [91] in the 2023 Gartner Magic Quadrant for Enterprise Low-Code Application Platforms, demonstrating the widespread praise the company has received for its broad feature set and capabilities.

Businesses can accelerate their software development process with the features provided by the Mendix platform. Some of the key features are as follows:

- **Web and Native Mobile App Development:** Mendix makes it possible to create native mobile apps based on React Native and completely responsive online applications that meet a variety of user requirements [92].
- **Custom Code Capabilities:** Users can upload additional code as needed, allowing for the building of backend plugins in Java/Kotlin, frontend widgets in React.js, or updates to mobile apps in Swift/Java/React Native [92].
- **Scalability:** With options for both vertical and horizontal scalability, the platform can handle tens of thousands of connections at once, making it suitable for businesses of all sizes and scaling strategies. [92].
- **Full Lifecycle Management:** Mendix guides clients through every step of the software development lifecycle, including idea mapping, app design, deployment, product launch, maintenance, and process optimization [92].
- **Artificial Intelligence and Machine Learning:** Mendix utilizes machine learning and artificial intelligence (AI) technology. One such example is Mendix Assist, an AI co-developer that expedites design work, lowers error rates, and boosts output overall [92].
- **Cloud-Native and On-Premise Infrastructure Support:** Mendix seamlessly supports both cloud-native and on-premise infrastructures, allowing users to choose their preferred approach [92].

- **Process Automation:** The platform offers a process-first approach, allowing automation to speed up workflows, increase business results, provide visibility into processes, enable the installation of new solutions, and provide complete control over the IT environment [92].

Platform Architecture: The Mendix platform's architecture is an advanced and durable framework created to facilitate the development of high-productivity applications in an enterprise-grade, low-code, cloud-native environment. Mendix is a platform for building and deploying applications that is scalable, flexible, and efficient. It does this by utilizing modern cloud-native principles [93]. Core principles behind Mendix platforms architecture is described below:

- **Model-Driven Development:** Mendix takes a model-driven approach that allows developers to focus on application logic rather than traditional code-based development using visual modeling tools. This methodology improves collaboration, accelerates development cycles, and reduces coding complexity [93].
- **Microservices and Containerization:** By leveraging microservices architecture and containerization, Mendix ensures scalability, agility, and portability in application delivery. By breaking applications into smaller, independent services, developers can easily scale components and effectively manage dependencies [93].
- **Stateless Server Architecture:** The stateless nature of Mendix servers enables seamless vertical and horizontal scaling, ensuring high availability and performance across different workloads. This architecture enables demand-based resource allocation and efficient use of computing resources [93].
- **Openness and Extensibility:** Mendix promotes openness and extensibility across its platform, allowing developers to customize applications, integrate with external systems via APIs, and use the Model SDK to manage app metadata. This flexibility allows companies to tailor solutions to their specific needs and integrate seamlessly into existing infrastructure [93].
- **Alignment with Twelve-Factor App Principles:** Mendix follows the 12-Factor App methodology and adheres to best practices for developing cloud-native applications. These principles cover aspects such as codebase management, configuration handling, dependency management, and more, ensuring consistency and efficiency in application delivery [93].

Mendix Platform is a comprehensive platform-as-a-service (aPaaS) solution for corporate application design, development, deployment, and administration. Developers and administrators may access the platform via the Developer Portal, which provides access to applications and services for requirements management, development, and deployment in app and app service operation and administration [94]. To expedite development, the

platform offers hundreds of publicly available building blocks through the Mendix Marketplace and Mendix Studio Pro. Apps and building blocks can be shared amongst organizations by configuring the Mendix Marketplace for private use. With Mendix Studio Pro, Mendix apps, and the Developer Portal, users can collaborate online on this platform. The figure 4.15 below provides an overview of the main aspects of the Mendix Platform..

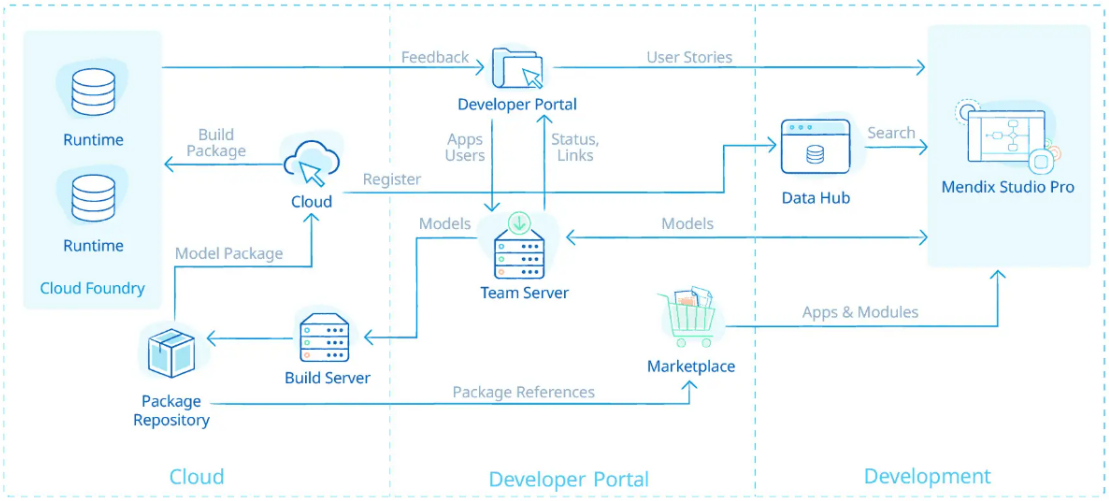

Figure 4.15: Key Components of Mendix Platform [94]

- **Developer Portal:** An interactive platform for app design, development, and deployment is provided via the Mendix Developer Portal [94]. This web site has features for online social collaboration (via Buzz), notification services, and a list of people who are actively involved in your company and who might be asked to participate in app initiatives and social collaboration.
- **Mendix Studio Pro:** Mendix offers an integrated development environment (IDE) for multi-user modeling called Mendix Studio Pro. Creating an integrated and unified modeling environment is the main objective of Mendix Studio Pro, which enables business analysts and IT developers to work closely together to model the different application components. For entirely offline work, Mendix Studio Pro features an integrated build service and runs locally on the developer's PC [94].
- **Team Server:** Application models are maintained and versioned centrally on the Team Server. The prominent open-source software configuration management system Subversion (SVN) inspired the creation of Team Server. The team server may be set up locally or hosted on Mendix Cloud [94].
- **Build Server:** On the Team Server, each application artifact is versioned and stored in a project directory. The Build Server creates deployment packages from artifacts

(including models, style sheets, and special Java classes). The build server may be launched using the Mendix Runtime or developer portal. The build server also does package validation and determines if the particular build is a deployable package [94].

- **MxID:** MxID is a user management and provisioning solution that uses the OpenID standard and can be integrated with the Active Directory and Single Sign-On (SSO) protocols. It acts as the Developer Portal's login server, which is the main entry point for Mendix apps [94].
- **Marketplace:** Though it may also be configured as a closed corporate marketplace, the Mendix Marketplace is a public marketplace for applications and application modules that lets business leaders and end users know which apps are accessible inside their organization [94].

Limitations: Like all other low/no-code platforms, Mendix also has some limitations. Some of the limitations are as follows:

- **Limited Customization:** Customization of the themes on the Mendix platform is limited. Users don't have the option to tailor the appearance of their application as they want. Also, the default UI gadget provided by the platform may not meet the specific design requirements of complex enterprise applications [95].
- **Vendor Lock-in:** Adopting the Mendix platform ties an organization to the Mendix ecosystem. Migrating an application built with the Mendix platform is really challenging and costly, which in a way creates a vendor lock-in situation [95].
- **Incorporating Advanced SQL Features:** When it comes to using complex SQL operations, Mendix could find it difficult to completely integrate the features of SQL, which could restrict the platform's potential in comparison to traditional code-based development [95].
- **Limited Scalability:** Mendix can be difficult to scale, especially in situations involving large-scale applications or high usage; extra setup and knowledge are needed to guarantee peak performance [95].
- **Styling Pages, Forms, etc., Within Mendix:** Styling pages, forms, and other elements inside Mendix is time-consuming, requiring a good CSS foundation and potentially limiting user interface modification [95].
- **Limited Control Over Infrastructure** Due to Mendix's Platform-as-a-Service (PaaS) nature, users' ability to manipulate the underlying infrastructure is restricted. Organizations with particular infrastructure preferences or stringent regulatory standards may find this lack of control concerning [96].
- **Pricing:** With its comparatively higher price tag than other low-code development platforms, Mendix's pricing model may pose a problem for smaller budgets [96].

4.6 Generative AI

Generative artificial intelligence (GenAI) has appeared as a game-changing technology that can transform the way we create and work with digital content. Fundamentally, GenAI uses machine learning models, mostly unsupervised and semi-supervised, to create new content out of the existing data. The birth of GenAI can be traced to the development of artificial neural networks in the 1950s and 1960s [97]. The first algorithm that was employed to use the computing power to classify the labeled data into categories was the Perceptron algorithm, which was first used in 1958. Thirty years later, the backpropagation algorithm was created in 1986, which allowed the multilayer perceptron networks to learn in non-linear ways; thus, the turning point in the development of the neural networks was realized [97].

Although it was only in the 2010s that GenAI started to close the gap, The Transformer architecture that was presented in 2017 solved the problem of the previous recurrent models of memory limitations, and consequently, the neural networks were able to manage context and relevance with ease. Efficiency improvement, together with the availability of large datasets and the growing power of computers, was the main factor in the creation of highly efficient language models such as GPT (Generative Pre-trained Transformer). GenAI has justly shown its importance in the NLP field [98]. The study on OpenAI GPT-3 and ChatGPT proves that these models are able to perform like humans in both the academic and professional fields, such as the 90th percentile SAT and the bar exam. These phrases will be the mainstay for the development of the text, which will be like a human because of the word prediction that is related to the preceding context [98].

The impact of GenAI has gone beyond text generation, and it has also contributed to other fields. The generative adversarial networks (GANs) have made the synthetic images very realistic, and the DALL-E and Midjourney models can produce images from the textual descriptions. RunwayML and Pika, which are video generation tools, can make short videos from text prompts or still images [98]. The development of GenAI will lead to the evolution of different industries and, at the same time, the birth of new possibilities in content creation, problem-solving, and decision-making. On the other hand, the rapid development of GenAI has also raised concerns about its misuse, for example, of deepfakes or other forms of misinformation. The problems that will increase while using GenAI will be the main reason for the generation and implementation of GenAI after the technology has developed [98].

Over the years, there has been the evolution of several key milestones and foundational models in the field of generative AI that have significantly contributed to the advancement of the technology. Some of them are described below:

- **Generative Adversarial Networks (GANs):** The two neural networks that comprise

Generative Adversarial Networks (GANs), which Ian Goodfellow and associates created in 2014, are a generator and a discriminator. They are simultaneously taught inside a zero-sum game framework, in which the discriminator determines the authenticity of the synthetic data produced by the generator [99]. With this competitive path, the approach enables the construction of supremely realistic outputs, which has a significant impact on domains like image manufacturing and development.

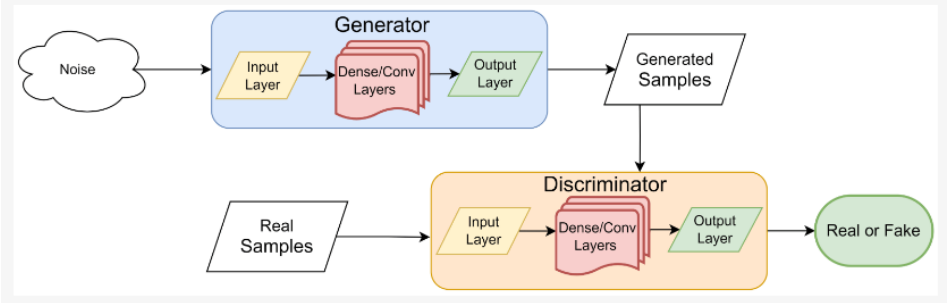

Figure 4.16: Typical Structure of Generative Adversarial Networks Model [99]

- **Variational Autoencoders (VAEs):** Conversely, VAEs were first proposed in 2013 by Kingma and Welling, sharing the same foundation as GANs. VAEs are a specific kind of generative model that recognizes an input code and then generates new pictures or samples based on that code. They consist of a Decoder function that pulls the output from the Latent space after the Encoder function maps the input into a Latent space. VAEs are utilized in numerous areas, including data augmentation, text production, and image generation [99].

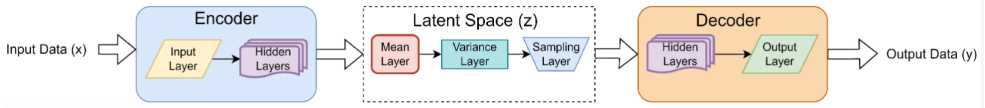

Figure 4.17: Typical Structure of Variational Autoencoders (VAEs) Model [99]

- **Transformers and Large Language Models (LLMs):** Transformers and Large Language Models (LLMs) have driven an astonishing revolution of Natural Language Processing through the marvelous self-attention mechanisms that allow the long-range dependencies in the texts to be effectively handled, and make parallel processing and scalability possible. These well-known LLMs that are based on Transformer architecture, such as the GPT-3 and BERT, display remarkable capability in generating text that sounds natural, they understand the context and perform such complex language tasks as proving their application in content creation, customer support, software development, and healthcare [99]. These models generated a new level of NLP benchmark challenging the possibilities of innovation and reaching new heights

of research. However, the debate continues, as researchers grapple with the problems of model interpretability, ethics, and efficient deployment.

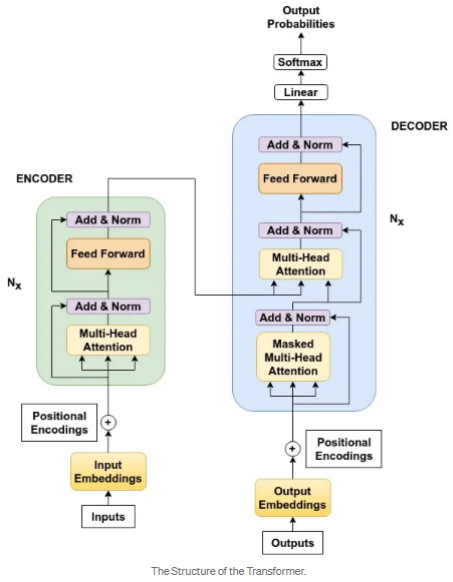

Figure 4.18: Typical Structure of Transformers Model [99]

4.6.1 Key Features and Benefits of Generative AI:

Some of the key benefits and features of generative AI technology are described below:

- **Intelligent Content Generation:** Generative intelligence will take the creative process to the next level by an imaginative second with the production of content suitable for the needs of the user – and it will traverse ways from a captivating marketing copy to attractive stories and articles [100]. It creates these human-like sentences from its extensive collection of teaching materials and utilizes context and the target audience to generate unique texts.

- **Accelerated Decision-Making:** Among the many advantages generative AI models have over humans, one of them is that they can quickly scan large amounts of data, identify patterns to make sense of, and offer the important advice that decision-makers require to make the right decisions [100]. They are capable of accomplishing tasks such as market trend analysis, risk assessment promotion, and strategic planning, which are added advantage that can help the business remain ahead of the trend.

- **Enhanced Creativity and Innovation:** Only one person or a team can use the generative AI as a tool to begin to explore their thinking and even generate both ideas and

concepts. The generative AI models would be a framework for thinking and a stimulus for creativity. The artists, the authors, and the designers can be more creative by knowing and realizing the limits of their works, or they can create new things by displaying their works [100].

- **Automation of Repetitive Tasks:** Generative AI has the ability to create content, including content marketing material, reports, and even source code, which in turn saves time as well as resources. The technology empowers workers to work on tasks that require high skill and require human creativity and cognition in general [100].
- **Personalization and Customization:** AI models can blend in generative technologies to know about what the client wants, which leads to a level of satisfaction and involvement. Installed AI chatbots and voice assistants that are powered by generative AI are the sophisticated technologies that are providing personal assistance to customers.
- **Accelerated Product Development:** Generative AI can accelerate the product development process by generating and iterating on design ideas, finding out potential risks at an early stage, and optimizing product features. Business stakeholders can utilize these insights to make informed decisions, optimize their operations, and gain a lead in the market [101].
- **Advancements in Healthcare:** Generative AI is being applied in the healthcare industry to expedite drug discovery, enhance illness detection, and customize treatment regimens. These algorithms may find interesting drug candidates, spot early illness indications, and provide medical practitioners with personalized suggestions by evaluating enormous volumes of medical data [101].

Although generative AI has many advantages, there are drawbacks as well, including the possibility of biased results, security issues, and moral dilemmas. It is critical to address these issues and make sure that generative AI systems are developed and used responsibly as technology advances.

4.6.2 Current State and Applications

In the last couple of years, generative AI has seen significant development, going from just a concept to a practical solution with a wide range of use cases across different sectors. The engineering and technology revolution is also changing the way we are creative, design and solve complex problems.

Text Generation: Large language models like GPT-3 and its different variants have shown remarkable capabilities in generating text that is human-like for different kinds of applications like content creation, storytelling, dialogue systems, and answering questions [102].

These models use self-attention and transformer architectures to extract long-range relationships from textual input [103].

Image Generation: Nowadays, generative adversarial networks (GANs) and diffusion models can produce photo-realistic pictures with the same quality as a real image [104]. For example, StyleGAN produces top-grade face images, while DALL-E is capable of generating various images using text descriptions as inputs [105]. These models have a prospect for use in design and creative fields, advertising, and multimedia content generation.

Audio and Video Generation: Although still at an early stage, generative models show that they have the potential to produce audio waveforms for text-to-speech conversion, music creation, and audio editing applications by their performance [106]. Another aspect that is being investigated in video generation is video-to-video synthesis or generative adversarial video generation which are two approaches that can be applied in various fields such as virtual reality and animation [107].

Music and Art: Generative AIs compose music and create visual art through the features with which to demonstrate their working abilities on the different platforms used by people for expression.

Software Development: Some of the AI tools like GitHub Copilot are helping developers achieve this by generating code snippets, suggesting, improvements, and automating repetitive tasks, which are assisting in the delivery of next-generation cutting-edge technology even at a significant speed [108].

Challenges: Indeed, generative AI technology can transform our lives for the better, but in addition to generating significant opportunities, it also brings important challenges that must be solved. The following are key challenges to of using this revolutionary technology:.

- **Fairness and Bias:**therefore, The basis of generative AI lies in the ambiguity of content creation and therefore fairness checking is quite a challenging task. While large language models cannot be assessed for fairness; traditional machine learning models can [109]. Disentangling and quantifying nuances, such as a slightly more irritated tone when making materials aimed at women than men, is a tough one to handle.
- **Privacy Concerns:** The broad features of generative AI create challenges not only for conventional data breaches but also for unidentified data training, which violates privacy in some cases [109]. Mitigating these risks by managing training data to steal personal information and applying strategies for detection and preventing

duplication of sensitive content is highly crucial.

- **Toxicity and Inappropriate Content:** The technology of generative AI can produce designs that are offensive, heartbreaking, or inappropriate. To solve this issue, the 'guardrail' models should be developed to filter out not-allowable information, training data, and produced outputs, including using human-annotated training data to help these models [109].

- **Hallucinations and Factual Inaccuracies:** Generative AI might have the ability of imitating content in a convincing manner but likewise might be the case with inaccurate information which is called hallucinations. In order to address this issue there is need for information that educates the users about the abilities as well as the limits of generative AI [109]. Integrate LLMs with databases that are verifiable and sources that can be used for fact-checking. And provide clear disclosures that tell the user on the capabilities the AI possesses so the user can make an informed choice.

- **Intellectual Property and Creativity Issues:** Intellectual property right is the primary concern because of the uncertain line between inspiration and imitation when generative AI reproduces content and styles [109]. Applying techniques to avoid or minimize using protected materials in creative products, along with an integration of technical, regulatory, and legal procedures, will be useful to overcome this problem.

- **Plagiarism and Academic Integrity:** When generative AI is used for writing essays and other academic or professional tasks, worries about plagiarism and originality come up. If models for the detection are designed for distinguishing between the human created content and material generated by AI in the future, the generative models and the detection methods may find themselves in an arms race [109].

- **Disruption to Traditional Work:** The efficiency of generative AI in automating jobs and producing content has generated questions concerning job loss while underscoring the importance of workforce adaptability [109]. Retraining, upskilling, and continuous learning are required if human intelligence is to harmonize with that of machines.

Addressing these challenges is really crucial, and it requires a multi-disciplinary strategy that involves high-quality training data, implementing robust ethical frameworks, collaborations between researchers, policymakers, industry stakeholders, and civil society, as well as educating users about the capabilities and limitations of this transformative technology. By properly handling these challenges, it is possible to utilize this groundbreaking technology for responsible development and deployment.

4.7 Generative AI and Low/No-Code Development

The domain of generative artificial intelligence (AI) is advancing quickly, offering fresh capabilities for the progress and revolution of application development, particularly in the context of low-code/no-code (LCNC) platforms [110]. By enabling users to construct programs using visual interfaces and pre-built components, even those with little to no coding knowledge, LCNC platforms aim to democratize software creation. Traditional systems, however, were limited in terms of more automation and context-aware help.

These difficulties may be resolved and the LCNC platforms might be further enhanced via generative AI, which entails teaching machine learning models to produce new material such as text, graphics, audio, and even code [111]. The power of large language models, computer vision models, and multimodal models can be employed by the LCNC platforms to provide natural and expressive interactions with users, automate parts of the development process, and also give intelligent recommendations [105].

LCNC platform integration with generative AI has the potential to change the way app development is performed, as it would make it more accessible, efficient, and customized to user needs. The integration comes with several problems, such as trust and reliability of the code or components that have been produced by the AI, addressing biases or limitations in the developed models, and developing suitable metrics for evaluation and user experience associated with this process of AI-assisted development [112].

As the field of generative AI is evolving rapidly, it is essential to delve into the possible applications, advantages, and risks related to integrating these technologies into LCNC systems.

4.7.1 Benefits and Features of Integrating Generative AI

GenAI with LCNC platforms has numerous features and capabilities that change the very nature of software development.Some of the core benefits and features of integrating genAI into low/no-code platforms are:

- **Automated Code Generation:** One of the most advantageous use cases for genAI with low/no-code platforms is code creation based on natural language or user input. These platforms might be dependent on large language models like GPT-3 and/or multimodal models like DALL-E, which allow the user to define the desired application as an outline, from which the AI system would generate and execute the necessary code or configure the necessary components[101]. This can make the software development process easy for non-technical users.
- **Enhanced User Experience:** Generative AIs can be applied as a method for UX/UI design for LCNC platforms to save time needed for generation of a predictive model

and components. Design tools can also be used to identify the right structure, color, and elements that need to be used in an application, and such applications are also differentiated from their use of AI and are also usable [101]. This leads to useful, and applications being user-friendly.

- **Multimodal Application Development:** As generative AI has the ability to handle multiple modalities like text, images, and audio, they can enable users to explain their desired requirements and applications in natural language, visual inputs which makes the development process more natural, expressive and intuitive for the users [101].

- **Personalization and Customization:** With the implementation of generative AI technology into low/no-code platforms, it is possible to enhance the personalization features of these platforms. AI can help to meet specific user requirements and needs by analyzing the user's data and behavior which can enhance user satisfaction and engagement. These features are particularly useful for creating personalized marketing campaigns, customer service chatbots, and e-commerce solutions that meet user's preferences [101].

- **Real-Time Error Detection and Correction:** Another important feature is the automated reporting and correcting errors in real time with the help of AI [101]. Generative AI applications can highlight any errors that are made while the user constructs the application and suggest potential ways to solve the problem. This saves a lot of time on developing and testing programs and correcting observed errors.

- **Accelerated Development Cycles:** Generative AI allows developers to perform repetitive and complex tasks much faster. Coding and debugging are two time-consuming, repetitive processes that AI can automate. This frees up the developer to concentrate on the higher-level aspects and decision-making process of software development. [101]. This allows businesses to develop their applications much quicker and in turn they can respond better and faster to the changing market needs.

- **Improved Accessibility and Inclusivity:** When GenAI is integrated with LCNC platforms, non-technical users may actively participate in application creation, which enhances accessibility [101]. This democratization of software development allows a larger number of people to create creative apps without significant coding experience.

Overall, the integration of generative AI into LCNC platforms has the potential to significantly enhance the capabilities and user experience of these platforms, making application development more efficient, intuitive, and accessible to a broader range of users.

4.7.2 Current Scenario:

Some of the major technology companies and startup has already began to integrate generative AI technology into their low/no-code development platforms. Some of them are:

- Microsoft has integrated Large Language Models and code generation capabilities into its Power Apps platform [113].
- Google and Amazon have also taken similar initiatives for their platforms App Maker and Honeycode, respectively [114].
- Startup companies including Appian and Mendix have investigated genAI for automatic UI design and component recommendation [115].
- Salesforce's Einstein platform also integrated AI so that it can offer predictive analytics, natural language processing(NLP), and automated recommendations. It helps users build apps that are smarter by automating data analysis and providing actionable insights, thus enhancing the decision-making process and user experience [116].
- Github Copilot also integrated generative AI for code generation. It suggests the user with the code and automating the repetitive tasks in the development cycle [117].

However, integration of generative AI into these low/no-code developments is still at an early stage but with the features and possibilities this technology can bring, all the companies and platforms are working on the implementation of this technology in their platform to provide their consumers more satisfactory user experience.

4.7.3 Key Challenges and Issues

Integrating generative AI into LCNC platforms has a lot of potential benefits, but it also comes with its fair share of challenges that need to be addressed

- **Privacy Concerns and Data Security:** Keeping data safe and protecting privacy when feeding information into generative AI models are big challenges when it comes to incorporating GenAI into LCNC systems [118]. When using GenAI for tasks like generating code and developing applications, it's super important to pay close attention to data privacy and security [110].
- **Varying Outcomes and Reliability:** Because generative AI is inherently unpredictable, it can produce different results, so it's important to carefully consider specific use cases to ensure accurate outcomes. While GenAI can automate tasks and offer code snippets, it can be challenging to consistently guarantee correct results in various situations due to the unpredictable nature of AI-generated code [119].

- **Balancing Automation and Human Expertise:** Another key issue is striking a balance between automation and human expertise in software development [119]. While GenAI can help create code and automate tasks, it can't completely replace the need for human developers, especially AI in designing complex applications
- **Learning Curve and Adoption:** Natural language-driven implementation of AI tools on low-code/no-code platforms may require users to adapt to new ways of interacting with software development tools. For widespread adoption and successful implementation of this technology, it is important to overcome the learning curve associated with using GenAI for code generation and application development [110].
- **Compliance and Governance:** Establishing appropriate governance policies and guaranteeing compliance with data protection requirements are important factors when integrating GenAI into LCNC systems. In order to mitigate risk and ensure safe ethical use of Generative AI in the software development process it is important to pay attention to issues related to compliance, data security, and governance requirements [119].
- **Scalability and Computational Requirements:** Training and automating an AI model on a large scale can be computationally challenging, consuming large amounts of energy and computational resources. Because LCNC systems are designed to serve a wide variety of users, it is important to ensure flexibility and manage resources [109].

Addressing these fundamental challenges and issues of integrating generative AI into low/no-code platforms is essential to leveraging the full potential of AI technologies, as well as data security, reliability, and compliance, and these new tools' effective use in software development guarantees

Chapter 5

Analysis

This chapter explores both the buisnesss and technical aspect of low code and gen AI. In the first sub section, the mechanisms behind Low code development platforms will be explained. The life cycle of LCD will be defined in order to understand where the implementation of gen AI is most appropriate. In the second subsection, the business and strategic impact of low/no-code platforms and generative AI is explored. We conducted a study of the market, SWOT analysis, and Porter's Five Forces framework to evaluate the competitive environment and identify potential possibilities and difficulties with these technologies. We discuss technical elements of low/no-code platforms and generative AI. We explored the structures, development life cycles, and integration capabilities of generative AI, identifying optimal stages for its implementation. The section will end with the specification of the requirements.

5.1 Technical Analysis

5.1.1 Low code platform Architecture

Low code platforms are typically deployed either through a cloud-based delivery model, known as Platform-as-a-Service (PaaS), or as on-premises solutions[120]. Architecturally, they generally consists of several distinct layers (see figure 5.1), below they are listed and explained:

- Application Layer: This layer houses a user-friendly graphical environment which contain among other things toolboxes and widgets tailored for defining user interfaces. Additionally, it hosts mechanisms for user authentication and authorization. Users interact with this layer to leverage its modeling tools and abstractions, facilitating the development of applications and the specification of their functionalities[120].
- Service Integration Layer: This layer serves to facilitate seamless integration with and utilization of diverse services, including application programming interfaces (APIs)

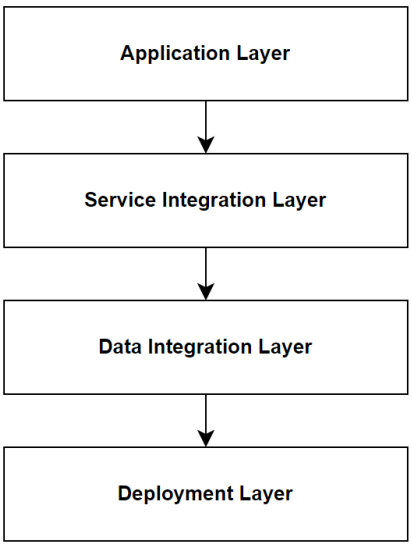

Figure 5.1: Layered Architecture based on [121]

and authentication mechanisms. It acts as a bridge between the platform and external services, enabling efficient communication and data exchange[120].

- Data Integration Layer: This layer is responsible for facilitating the integration, management, and manipulation of data sourced from several origins, this layer plays a crucial role in ensuring data consistency and accessibility. It provides the necessary infrastructure for harmonizing data from disparate sources and making it available for use within the platform[120].

- Deployment Layer: Responsible for orchestrating the deployment of developed applications, this layer ensures their smooth transition into a dedicated cloud or on-premise environment. Working in cooperation with the service integration layer, it manages the containerization and orchestration of applications, optimizing their performance and scalability[120].

Together these layers form the foundational infrastructure of low code platforms, enabling users to seamlessly develop, integrate, manage, and deploy applications with ease and efficiency.

To further deepen our comprehension of Low code platforms, let us look at the different components that make up these platforms. Figure 5.2 showcases the different entities and their interactions. We can see 2 lines dividing the figure, making it consist of three sections:

- Application Modeler (bottom section):

  This tier comprises the graphical environment that enables users to design applications using various modeling constructs and abstractions (drag drop). When the application model is finalized, it is transmitted to the platform's back-end for further processing. This includes the generation of a fully functional application that is tested and prepared for deployment on the cloud[121].

- Server-Side Infrastructure (middle section) This tier handles the server-side functionalities essential for the platform's operation. This includes model management operations such as code generation and optimization. It also involves managing interactions with various services like database systems (both SQL and NoSQL), microservices, API connectors, model repositories of reusable artifacts, and collaboration tools. The platform ensures that all necessary microservices are created, orchestrated, and managed without user intervention, relieving developers from the responsibility of handling technical details such as authentication, load balancing, business logic consistency, data integrity, and **{ajeethrabalamurugan_2023_auditing}**, **\cite {a2020_scihub}**.

- External Services (top section) The third tier encompasses the external services integrated with the platform. These include APIs and other external systems that the platform interacts with through specific connectors designed to handle the consumption and management of these services on the back-end[121].

5.1.2 Low code development life cycle

The life cycle of low code development is like that of the traditional software development, however, one can say it is built on a shorter and a more efficient foundation[122]. In a generalized manner, below are the stages of the low code development life cycle:

- Requirements Gathering: In this stage, requirements related to the user experience, the functionality of the program, and business needs are gathered [122].

- Design: This stage involves the development of the program's architecture, user interface, and other design elements. Wireframing and prototyping may also be done [122].

- Development: This stage focuses on the actual development of the application. Visual editors and drag-and-drop tools are frequently used in low-code systems to make programming easier [122].

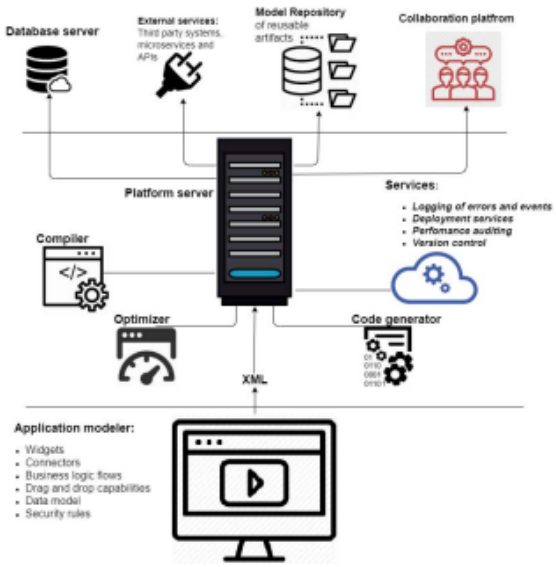

Figure 5.2: What a low code platform is made of [121]

- Testing: The program is tested to find and fix bugs, problems, and usability concerns. To facilitate this process, low-code systems usually provide automated testing tools [122].
- Deployment: After the application has been approved and tested, it is deployed into production environment [122].

The low code life cycle can follow various development methodologies, however, low code software development aligns well with agile development methodologies, as both prioritize customer satisfaction and the continuous delivery of incremental improvements [123]. The fundamental motivation behind LCSD is to build applications, gather user feedback, and quickly incorporate changes [124]. This approach naturally complements agile principles, which emphasize iterative development and frequent stakeholder involvement [123].

The delivery of software through a low-code methodology involves multiple phases of software development. While low-code development can integrate various software development approaches, fast delivery practices are commonly used due to the streamlined nature of low-code platforms[120].

The research discusses the application of several frameworks in low-code development. For example, the integration of low code within Scrum and Kanban methodology[121], [125], [126]. Furthermore, several sources have mentioned the application of Rapid Application Development (RAD) in relation to low-code development [127], [128], [124]. The reason being RAD appears to be effective for low-code development in terms of quick

reviews, feedback, and adapting to constant requirements change. Nevertheless, it might not be the best choice for projects with big teams, lengthy development timelines, or a shortage of highly qualified engineers[129]. In addition, these papers fail to go in depth and provide a detailed descriptions of the development life cycle steps specific to RAD within low-code development[120].

Despite the diversity of methodological approaches focused on low-code development, many are often limited to the functionalities of specific platforms. A more comprehensive methodological framework is required to support development processes across various low-code platforms[122]. Although, it is evident that, despite variations in the naming and focus of different phases, all approaches consistently include phases dedicated to requirements, testing, and post-deployment activities[120]. Given the various research papers linking agile methodology to low-code development [123], and the fact that multiple low-code development platforms are equipped with features that support agile development [121], [130], [131], this approach will be adopted and discussed in the upcoming section.

The figure 5.3 showcases the stages involved in the agile development process in both traditional (outer cycle) and low code software development (inside cycle). As seen in the figure the low-code development life cycle includes phases similar to those in agile methodologies. This iterative process ensures that applications are developed and refined quickly, incorporating user feedback at every stage. Monitoring and maintenance features, such as automatic error reporting are often built into low-code platforms[122], further enhancing their efficiency and reliability.

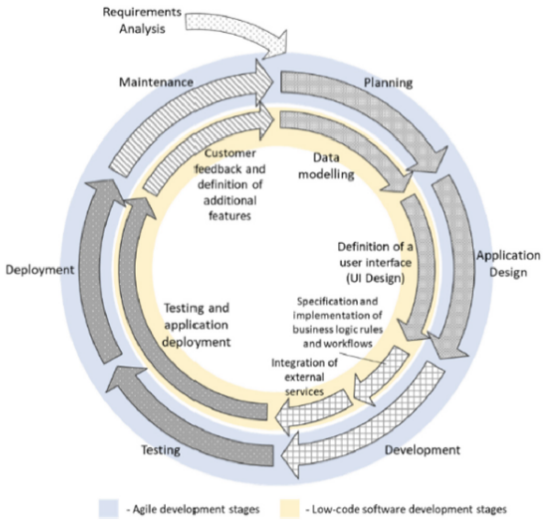

Figure 5.3: LCD life cycle stages[131]

The phases in the diagram are listed and explained below [121], [123], [122] are established:

- Data Modeling

  In the initial stage of data modeling, the developer utilizes the requirements collected by the business analyst or gathered during meetings with end-users. These requirements are then validated and transformed into a comprehensive data model. This model serves as the foundation for the application's functionalities and informs the subsequent design of the user interface.

- User Interface Design

  Following the establishment of the data model, the process advances to the second stage: user interface development. In this phase, the UI is crafted based on the predefined data model, ensuring that the interface aligns with the application's data structures and user needs.

- Business Logic Implementation

  The third stage involves the implementation of business logic. Here, the developer defines the rules and behaviors for each data entity and outlines the application's workflow. This ensures that the application operates according to the specified requirements and business rules.

- Integration of External Services

  In the fourth stage, the focus shifts to integrating external services by the use APIs. This integration is crucial for enhancing the functionality of the low-code application and ensuring seamless interaction with other systems.

- Testing and deployment

  The fifth stage encompasses testing and deployment. The developed application undergoes testing to identify and resolve any issues and then deployed.

- Customer feedback and definition of additional features

  Finally, it is essential to collect user feedback to identify areas for future improvement or additional features that may arise. This ongoing process of updating the application ensures that it continues to meet user needs and adapts to any changes in requirements.

5.1.3 Stages where gen AI can come to play

Having identified the various stages of the low code development, the next step is to investigate where generative AI can be used. the question is, in what stages of the development life cycle would genAI be helpful? Ideally we would have liked to get first hand answers by interviewing experts from a company specialized in low code developed solutions, that plan unfortunately failed as the company refused our request. Consequently, we will rely on research papers with similar agenda as ours, as this is a recent topic, research is rather limited. To understand where gen AI can be used, we will first attempt to identify the challenges or limitations of low code platforms at each each stage to better understand the potential of gen AI[132],[133] at the various stages of the development life cycle. Our focus will on the challenges experienced by the end user (Low code developer) rather than the limitations of the platforms themselves.

Domain Modelling

In this early stage of the development process, the developer could experience difficulties in correctly translating the gathered requirements into the low code platform. Here, determining the data structure, relationships, constraints, data storage and more fall on the user[134]. In addition, changing requirements are to some extent another challenge faced by the developer [131]. The possible improvements that gen AI can bring to these challenges is by automatically generating draft models based on the gathered requirements[122].

Limitations and Challenges	**Generative AI Improvements**
• Complexity in defining accurate domain models. • Ensuring relationships and constraints are correctly represented. • Adapting models to evolving requirements.	• Gen AI can generate domain models from natural language descriptions and existing templates, reducing complexity. • Gen AI can validate relationships and constraints to ensure accuracy. • Gen AI can suggest model adjustments to accommodate requirement changes.

User Interface Definition

In this phase, the developer could face challenges related to the friendliness of the interface[135]. Another already mentioned issue is customization since low code platforms tend to have pre-built components limiting the options the developer have. This can be mitigated by having gen AI generate UI components based on the developer's textual input.

Limitations and Challenges	Generative AI Improvements
• Balancing user-friendly design with functionality. • Limited customization options in low code platforms.	• Gen AI can generate user interface prototypes based on user requirements. • **G**en AI can recommend design customization.

Business Logic Specification

In this phase of the development stage, developers can face challenges with implementation of business logic, and require programming knowledge[123]. Gen AI can help with workflow generation and optimization[134], in addition to code generation that can help citizen developer and reduce development time[3].

Limitations and Challenges	Generative AI Improvements
• Complexity in defining intricate workflows and rules. • Ensuring business logic aligns with requirements.	• Gen AI can analyze and optimize workflows for efficiency and performance. • Gen AI can verify business logic against specified requirements and suggest corrections.

Integration with External Services

As for the integration with external services, developers can face difficulties with 3rd party services integration and compatibility. In some cases, knowledge in software development can be needed [131]. Gen AI can help with

Limitations and Challenges	Generative AI Improvements
• Ensuring compatibility with external systems. • Managing and maintaining API connections. • Programming knowledge.	• Gen AI can automate the generation and management of APIs, simplifying the integration process. • Gen AI can verify compatibility and suggest necessary adjustments. • Automatic code generation

Testing and Deployment

In the testing phase issues ranging from running automated testing and 3rd party testing methods[123]. Gen AI can relate In general there seem to be lack of documentation that causes confusion in several stages. Gen AI can help with writing text cases, help the developer interpret error messages[122].

Limitations and Challenges	Generative AI Improvements
• Ensuring thorough testing coverage. • Identifying and resolving deployment issues.	• Gen AI can generate and execute test cases to ensure thorough testing coverage. • Gen AI can help interpret/resolve error messages.

As we have seen, gen AI can beneficial in all stages of the development process, however, depending on the context of its application, it is crucial not to rely on generative AI as a standalone tool in any phase of the development process in its current state. The optimal approach is to integrate generative AI with developers to validate all outputs. Despite its immense usefulness, there have been instances where generative AI has produced incorrect results ([136], [137]). Therefore, involving developers alongside generative AI5.4 is essential for achieving optimal results and enhancing the low-code software development process.

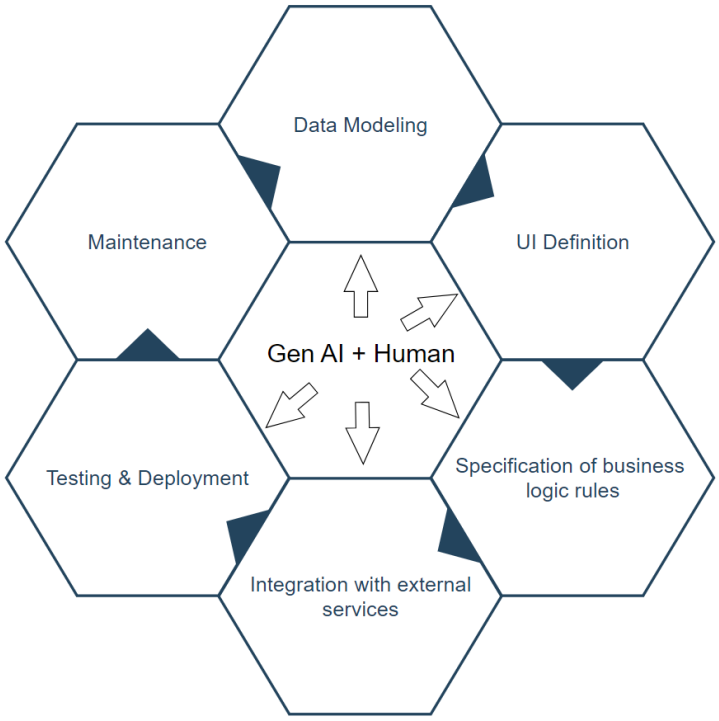

Figure 5.4: Gen AI in the low code life cycle

Many of the suggested gen AI improvements can take form as a ChatGPT-based chatbot, integrating it directly into the low-code platform. This chatbot can assist developers at the various stages of the development process, providing support and guidance based on textual input. By leveraging the capabilities of ChatGPT, developers can receive real-time suggestions, troubleshooting assistance, and conceptual clarifications, enhancing the overall efficiency and effectiveness of the development workflow. This integration ensures that developers have continuous access to AI-driven support, helping to address and overcome the mentioned challenges and limitations.

Considering the layered architecture we have already introduced earlier 5.1, the low code platforms are built to seamlessly integrate with external services. The integration of the

GPT model into the chat bot would happen through the use of the OpenAI API[138].

5.2 Business Analysis

5.2.1 Stakeholder Analysis

Stakeholder analysis is an essential process that builds the framework for successful project execution by identifying and engaging key individuals or groups having a vested interest in the project's outcome. Understanding stakeholders' needs, expectations, and influence is essential for gaining support, managing expectations, and aligning initiatives with company objectives. Stakeholders might include a wide range of individuals, including investors, team members, clients, partners, and end users, each with their own perspective and contribution to the project.

The stakeholder analysis section of the report seeks to offer a complete assessment of the important stakeholders engaged in the project, including their level of influence and interest in the project's success. Prioritizing communication tactics, engagement techniques, and decision-making processes is made easier by classifying stakeholders based on their power and interests.

Key Stakeholders of the Idea: For our idea of integrating Generative AI in low/no-code platforms, we identified the key stakeholders associated with this. Their individual impact and contribution to the idea is described below:

- **End Users:** For our conceptual low/no-code development platform integrated with Generative AI, end users are pivotal stakeholders. Their active engagement will significantly influence the proposed platforms design, functionality and market success. They participate in direct user feedback, beta testing, and the promotion of the platform through the creation of advocates and the documentation of successful use cases. User feedback is essential for fine-tuning platform functionality, proving its applicability, and directing development efforts in a way that reflects real-world use. Additionally, end-user interaction data is critical for training and improving the AI models that enable the platform's unique functionality. Furthermore, their contentment and recommendation can bring more users, create a long-term user base, and raise fresh revenue through subscriber growth and sales. This way, users not only get the product but at the same time contribute to its completion, which is the core of the platform's evolution and success.
- **Software Developers and IT Staff:** Another important stakeholder group for this conceptual idea are software engineers and IT professionals. They not only keep the system running well, but they also initiate innovations and guarantee that the platform remains active and serves the needs of its users. Such experts in technology are

in charge of system installation, software upgrades, security checks, and making improvements to improve the user experience. They also help bridge the gap between non-technical users and complicated technology solutions by offering training and support services to ensure that everyone can fully utilize the platform's capabilities. And their position is vital not just in terms of the platform's technical integrity, but also in its evolution to fulfill both present and future demands, making it a relevant and powerful tool.

- **Project Managers:** Project managers are also very important stakeholders in the idea. They are the ones who do the behind-the-scenes tasks to make sure that the projects stay within budget as well as to maintain strategic objectives. Through coordination among development and marketing teams, the project managers manage every step of platform development and deployment so that these steps are aligned. They are effective in timelines, resource allocation, and people management, and can tackle any problems that could arise and finish the work smoothly. They serve as an important channel of communication for keeping the stakeholders informed and up to date about any changes or new developments. Their monitoring capability is critical not only in this regard but also in terms of accommodating feedback and market dynamics, thus the project remains flexible and tuned to the users' demands.

- **Business Analysts:** Business analysts are one of the crucial stakeholders in the building of platforms wherein the Generative AI technology is integrated with the low/no-code capabilities. They are the link between the possibilities of technology and business goals, whereby they decode the complex data and also the feedback from the users into actionable information. They are supposed to really understand both the market conditions and customer needs for creating functionalities that really work for the users. Through detailed market research and feasibility studies, business analysts guarantee that the platform does not only satisfy present market requirements but also provide for the future. They help to prioritize feature development following strategic importance and ROI that, at the end of the day, ensure the platform's commercial success. The analytical skills of business analysts enables them to serve the role of providing solid data-driven foundation that helps in decision-making process; this makes them vital players in the process of aligning business strategies with technological advancements.

- **Quality Assurance/Testers:** QA (Quality Assurance) testers are another critical stakeholder in the platform idea. They make sure that the software is not only operational but also aligned with the design specifications, taking into account the users' requirements for usability, functionality, and reliability. Through repetitive testing of every aspect of the platform, QA testers track down bugs and issues before they affect end-users, ensuring smooth functionality. They perform rigorous testing, which helps keep standards high and, in the bargain, prevents expensive errors that could

spoil the platform's reputation and user satisfaction. In addition, QA testers also contribute to the platform's development by giving feedback that can be used to make the platform more innovative and efficient. Their work guarantees that new releases are highly stable and boast superior functionality, creating a pleasant experience for users and increasing the credibility of the platform in the face of strong competition.

- **AI Researchers and Data Scientists:** AI researchers and data scientists are also essential stakeholders in developing the platform. They are the driving force of the system given that they develop and adjust the AI models that automate and improve certain features. Through the use of advanced machine learning algorithms as well as data analysis methods, they verify that the AI parts are not only innovative but also practical and efficient enough for their end users. Their work consists of constant experimenting and iterating to harvest AI precision and intelligence, which makes the system more cognitive and interactive as time passes. Additionally, AI researchers and data scientists scrutinize users data to get insight for the development of new features or improvements which are based on the real world usage of the platform and affect the evolution of the platform. Their efforts play a key role in promoting technological advancement and market competitiveness, which ultimately leads to an elevated user experience and gratification.

- **Regulatory and Compliance Officers:** Regulatory and Compliance Officers are also very important stakeholder for the development of the conceptual platform. They ensure that the platform maintains industry regulations, data protection laws, and ethical guidelines, which are crucial for maintaining trust and legal compliance. Their involvement is essential for identifying risk factors and taking efficient measures to mitigate them to safeguard the platform from legal issues and its reputation. They also play a vital role in keeping the platform up-to-date with changes in legislation, which is a challenging task as the technological and regulatory evolution of AI is happening rapidly. They not only protect the organization but similarly, they make the users confident that their data is dealt with carefully and securely through the whole process of development and deployment by guaranteeing compliance.

After determining the stakeholders for our idea, we categorized them based on their influence, interest, and levels of participation using the power-interest grid. There are four categories for the power-interest grid mapping and they are [139]:

- High Power, High Interest: "Manage Closely"
- High Power, Low Interest: "Keep Satisfied"
- Low Power, High Interest: "Keep Informed"
- Low Power, Low Interest: "Monitor"

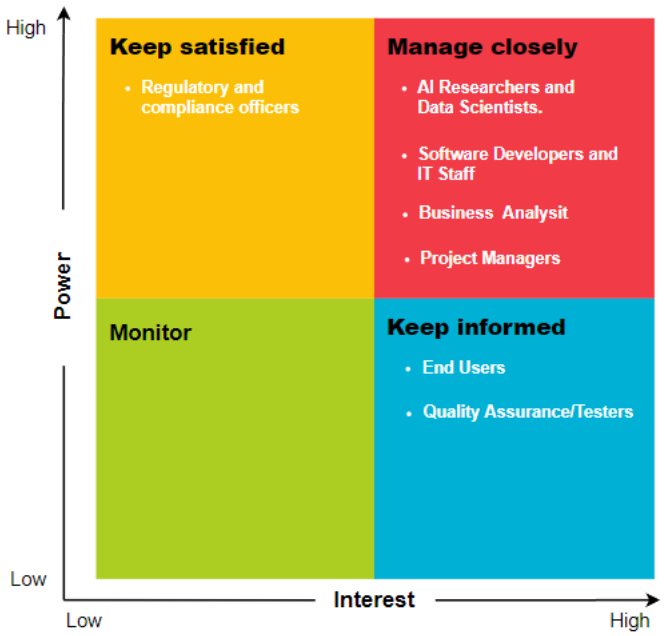

Figure 5.5: Power Interest Grid

For our idea of the platform, stakeholders like AI researchers and data scientists, developers and IT staff, business analysts, and project managers fall under the high-power, high-interest grid. Then comes the high-power, low-interest grid, where we placed regulatory and compliance officers as we had to keep them satisfied. Lastly, we placed end users and quality assurance/testers in the low-power, high-interest grid, and we have to keep them informed. An explanation of the position in the grid is described below:

- **High Power, High Interest:- Manage Closely :** These are the people who has the highest power and interest in the project so it is required to manage them closely. For our idea of the Generative AI integrated platform, stakeholders like AI researchers and data scientists, software developers and IT staff, business analysts, and project managers were placed in this group based on their influence and interest in the idea. Their technical expertise will be crucial for the success of implementing the idea. They play a crucial role in decision-making, so they are required to be managed closely. AI researchers and scientists will innovate, which will shape the platform's core functionalities; software developers and IT staff will ensure the platform's robustness and integration capabilities, business analysts will play the role of translating the user's needs into actionable plans; and project managers will need to look at the project resources and timelines. It is essential to keep them engaged through regular updates, strategic involvement, and robust feedback. As they hold an influ-

ential role as well as high interest in the idea and are required to manage closely, we placed them in this grid.

- **High Power, Low Interest:- Keep Satisfied:** In this section, we need to place the stakeholders who are required to keep in loop, as they need to be kept satisfied even though they have a lower interest in the project idea but hold high power. They need to be handled carefully, and their power can be used in an undesirable way if they are not kept satisfied. For this reason, we placed regulatory and compliance officers into this grid. For successful implementation and the future, we need to keep them satisfied by fulfilling all the existing rules and regulations. Their role is crucial for ensuring that the platform meets legal standards. We need to focus on keeping them well-informed and compliant, ensuring their needs are met through periodic updates and streamlined communications.

- **Low Power, High Interest:-** Last comes the low-power, high-interest quardant for our idea. We placed the end users, such as business users, developers, and quality assurance/testers, in this, as we need to keep them adequately informed and communicate with them to ensure that their is no issue arising. They are deeply invested in how the platform performs and functions because it will directly affect their work or the quality of the product they are making using the platform. They need to be kept well-informed and involved. Regular communications, updates, and feedback sessions are required to handle these stakeholders, as their feedback greatly helps to enhance the platform's features and functionalities. Quality assurance/testers are also required to be regularly updated and informed, as they need to be well aligned with the upcoming features, detailed changelogs, and development plans so that they can properly plan their testing needs and requirements.

5.2.2 Market Analysis

In this section, we will dive into a comprehensive analysis of the low-code/no-code development platform industry, especially focusing on the integration of generative AI into this platform's code generation and design tasks. We will try to provide a detailed overview of the current market landscape, growth trends, key players, adoption patterns, and future prospects for this market segment. By analyzing the market segment, competitive landscape, and future opportunities, we will shed light on the market potential and implications for the successful implementation of the idea.

Market Size and Growth Rates: The LCNC development platforms and generative AI industry are currently evolving and expanding rapidly because of the increasing demand of fast and effective application development solutions and integration of AI into application generation processes.

It is expected that the low-code development platforms market will expand globally from USD 28.75 billion in 2024 to USD 264.40 billion by 2032, with a growth rate (CAGR) of 30.90% [140]. Gartner predicts that sales of low-code development solutions globally will reach around USD 31 billion by 2024, indicating growth over previous years. [141]. As a

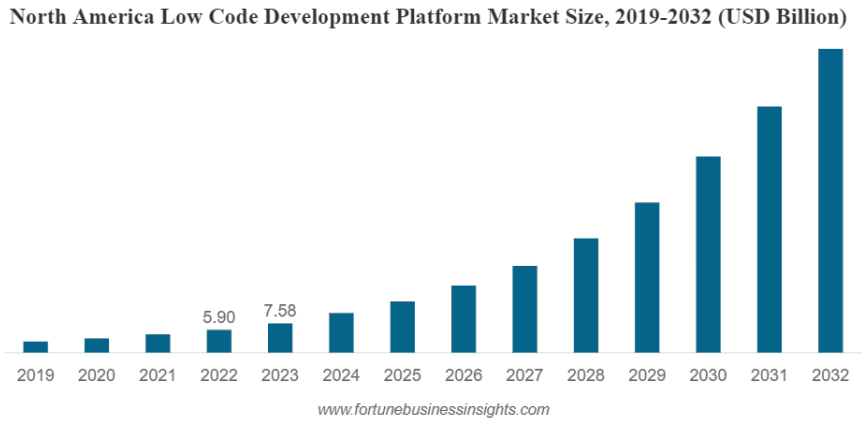

Figure 5.6: Low Code Market Size [140]

result of the development of generative AI technologies, the cloud computing industry is expected to grow from USD 0.58 trillion in 2023 to USD 1.24 trillion, according to Gartner's forecast [142]. The adoption of code intelligence is predicted to drive market growth as companies invest in this platform for benefits such, as enhanced data analytics, improved user experience, cost efficiency, and faster development processes.

Historical Data and Future Projections: It is anticipated that the low-code market would expand by around 40% due to anticipated increases in investment [142]. By 2025, 70% of apps created by businesses will use low/no-code technologies, according to Gartner [142]. This is a significant increase over prior years.

The adoption of Generative AI models and tools has surged rapidly, with major cloud providers enhancing their infrastructure and capabilities to support clients in leveraging Generative AI for business purposes [118]. The increasing embrace of low-code AI is anticipated to drive market expansion as organizations invest in platforms that offer data analytics improved user experiences and expedited development processes.

Integration of Generative AI and Low/No-Code Platforms: The integration of generative AI and low-code platforms enables users to change how the program will behave without the need to handle underlying intricate information [118]. It can be input in the form of suggestions such as "Can I have a code snippet for changing the format of dates?" or "Can you design a flow that updates inventory automatically?" This way, it brings efficiency to coding and fosters creativity.

This fusion re-creates the efficiency of the software interface, because tasks that once required tedious configuration, or programming, can now be done with commands. Through natural language prompts, users can dictate the behavior of software without necessarily having advanced technical knowledge, and thus innovation has been made easier [118].

Challenges and Considerations: Although the design of generative AI and integration with low/no-code platforms is promising, there are issues to be discussed and overcome. At this stage, privacy is a very important issue when dealing with data that is fed to generative AI models and finding the right balance between the provision of useful insights and protection of data entering the model is often a difficult task [142].

Since generative AI algorithms are stochastic in their nature, the application may produce different results at different times, so one needs to be vigilant when choosing an AI solution for a particular task [118]. Leaders need to be attentive to the development of the governance layer as ensuring data integrity and security remain vital.

5.2.3 SWOT Analysis

To strategically evaluate our idea of integrating Generative AI into low/no-code development platforms, it was important to conduct a SWOT analysis. This helped us identify the internal strengths and weaknesses of our proposed idea as well as the external opportunities and threats it can face in the long run. It allowed us to identify the core advantages that our idea holds, such as increased accessibility and innovation in software development, while also identifying potential challenges like technical complexity or market acceptance issues.

- **Strengths:** In this section of the analysis, we identified the main advantages and strengths we have for our idea of integrating generative AI into low/no-code platforms to enhance their capabilities. Based on our analysis, we have identified the core strengths associated with the idea, like that it is an innovative integration that enhances the capabilities of traditional low/no-code platforms, making it more powerful, efficient, and attractive to users. It also has the potential to make software development accessible to a larger audience, including non-technical users, by simplifying the software development process. Another strength of the idea is rapid prototyping, as low/no-code development platform makes it easy to build applications without starting from scratch, rather using drag-and-drop features and built-in tools. and with the integration of generative AI technology, users can create applications using natural language commands/text. It can automate the repetitive workload by automatic code generation, testing, etc. This kind of platform also reduces the time consumption of the development process as well as the development cost, which are also strengths of the idea.

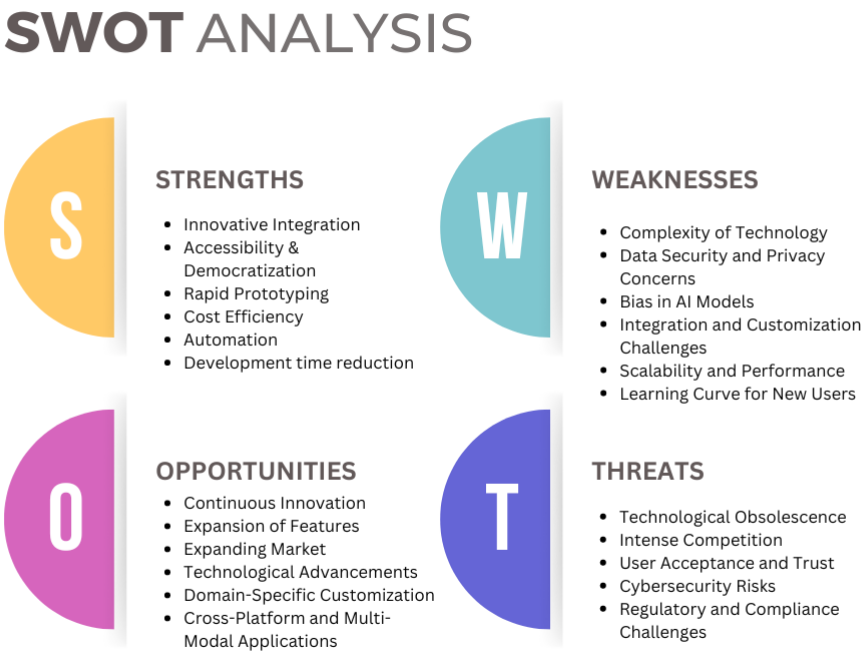

Figure 5.7: SWOT Analysis

- **Weakness:** We also identified the weaknesses related to the idea, and one of the main weaknesses is the complexity of technology. It requires high-level expertise in the sectors of AI, data science, and machine learning, and as the field is advancing rapidly, it is also difficult to keep up with the regular updates and grasp new things. Also, the platform's dependency on user-provided data for GenAI training has potential security and privacy concerns. Biases in the AI model are another weakness of the idea, as we have to train the AI model with the data for generating output, but if it is trained on biased data, then the output of the system will also be biased, which is a concerning factor. It is also a difficult task to integrate generative AI technology and maintain the platform with regular updates and functionalities. As the technology is new, the learning curve of new users can also be a weakness.
- **Opportunities:** The rapid advancement of the technological sector brings new opportunities to explore and work on. For our idea, we also identified the opportunities that lie in the integration of GenAI technology into these platforms. It is a field of continuous innovation, as new things and ideas are coming regularly with better aims and goals to achieve. It provides an opportunity to expand the market with the expansion of new features and capabilities in the platform. As the low/no-code and generative AI market is continuously growing and expanding it provides more

and more opportunities to work on. The combination of GenAI and low/no-code platforms enables faster development and innovation. It also enables opportunities for new use cases as well as domain-specific customization.

- **Threats:** The low/no-code market is evolving rapidly, with many new companies or existing large companies working with similar integration, which poses a threat in the long run for the market, which is crowded with various options for users. Technological limitations are also another threat to the idea, as generative AI is continuously evolving and might lead to uncertainties in its long-term viability and integration challenges. The increase in accessibility, also increases the possibility of misuse, as people can make applications that might violate privacy or spread false information or scams. Newer advancements in the evolving technological world also pose concerns about the idea becoming obsolete. There are also threats to cybersecurity. Intense competition in the existing market is another big threat and requires continued innovation to stay ahead in the market. On top of these, another threat lies in user acceptance and trust of the platform. It is difficult to gain the trust and acceptance of users. Also, there are regulatory and compliance challenges that need to be properly followed so that the solution always fulfills the legal requirements. These regulations and compliances also evolve timely, which can be a threat in the long run as well.

5.2.4 Porter's 5 Forces

Through Porter's Five Forces analysis, we can get a good idea of the nature of the competitive forces that will be imposed on the implementation of an idea. It assists companies in learning the competitive situation of their industry and detecting opportunities for expansion and innovation. It is relevant for our thesis idea to have a better understanding of the current market trends for the successful implementation of the idea. From the analysis of the power of a new entrant, the bargaining of a supplier side and a buyer side, competitive rivalry, and substitutes, we will be able to learn what is proper in the market situation and where there are opportunities and risks. It helped us to investigate the feasibility and possible effect of the combination of generative AI with low/no-code development platforms and to come up with a strategy that considers the competition and market trends. A detailed description of the analysis is described below:

- **Threat of New Entrants:** In this section of the analysis, we need to find the possibility of new entrants in the market, which also integrates generative AI into low/no-code development platforms. Due to the rapid advancement of AI technology, the increasing availability of development tools and services and the potential of low/no-code platforms can attract new entrants with innovative solutions. But integrating generative AI into these low/no-code platforms required significant technical knowledge, expertise in AI, data science, and development costs. With access to proprietary data

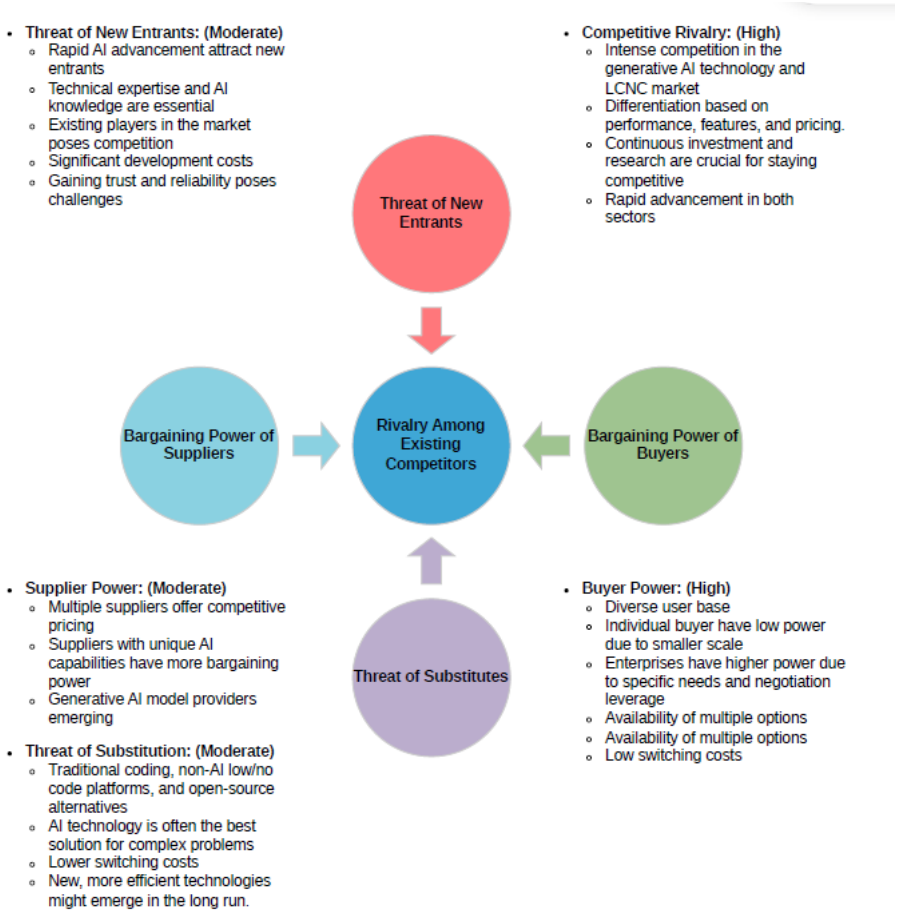

Figure 5.8: Porters 5 forces Analysis

and unique capabilities, new entrants can quickly enter the market with their unique features and compete with the existing players. However, gaining the trust and reliability of the user as a new entrant is also a challenge. But existing market competitors like OpenAI, Google, etc. who are already working with generative AI technology can pose a threat to the idea in the future. So comparing all of the factors, it can be expected that the threat of new entrants in the market is moderate or medium.

- **Bargaining Power of Suppliers:** As technology is rapidly evolving and more and more suppliers are already available on the market, users have multiple options in their hands to choose from based on their requirements and capabilities. For this reason, the bargaining power of suppliers in the AI technology field is also moderate. High competition among the suppliers helps to keep the prices of the products or services down and gives users more bargaining power. However, suppliers can have more bargaining power with unique AI capabilities, features, and proprietary data.

- **Bargaining Power of Buyers:** The bargaining powers of buyers can vary from moderate to high due to the diverse user base. As there are different user bases, like individuals, small or medium enterprises, and large enterprises, the power of bargaining also varies. Individual users have a lower level of bargaining power due to the smaller scale, and small or medium enterprises and large enterprises have a higher level of bargaining power with their specific needs and requirements and can negotiate on price or customization. Also, to keep the customer base happy and loyal to the platform, businesses need to fulfill buyer demands and offer competitive pricing to maintain their market share. Moreover, due to the availability of multiple options to choose from, they have the power to switch to a new one with relatively lower switching costs.

- **Competitive Rivalry:** Rivalry among the competitors in this market also ranges from moderate to high. Competition is intense in the generative AI technology market, as it is currently blooming everywhere with its capabilities and features. Many startups and big companies are implementing this technology into their relevant field to increase their efficiency and productivity. Moreover, the existing LCNC platform market is already competitive. However, the main factors that will differentiate the competitors will be their performance, unique features, and capabilities. Also, pricing and business models will be important competitive factors. To be in the competitive market, it is essential to keep up with the rapid advancement in this field and provide customers with better service and more reasonable pricing options than the available competitors. Continuous investment and research are required to stay ahead of competitors in the competitive landscape.

- **Threat of Substitutes:** The threat of substitutes for our idea is moderate or medium. There are some substitutes available in the market, like traditional software development, existing low/no-code platforms without generative AI integration, and some open-source platforms with basic generative AI functionalities. User can shift to any of these based on their needs or scenarios. But most often, AI technology is the best solution for complex problems rather than the traditional development process. As technology evolves, new, efficient technologies might appear on the market in the long run.

5.2.5 Value Proposition Canvas

It is essential to understand how the proposed idea's solutions fit with the demands of the consumer. For this reason, we created a value proposition canvas for our idea, which helped us visualize and define the value that our proposed innovative platform intends to provide to its users. This framework is particularly important for our analysis, as it helped us to systematically break down the features and offerings of the platforms against the real demands and challenges faced by the potential users.

The value proposition and the customer profile are the two primary parts of the value proposition canvas. By analyzing the jobs that need to be done by the users, the pains they currently face and wish to avoid, and the gains they want to achieve, the customer profile helped us to understand the user segment and the requirements they have deeply. On the other hand, the value proposition segment helped us identify the user pains our proposed platform with generative AI integrated should resolve and also create the gains to fulfill their needs and requirements. A detailed description of the two sections of the value proposition canvas is described below:

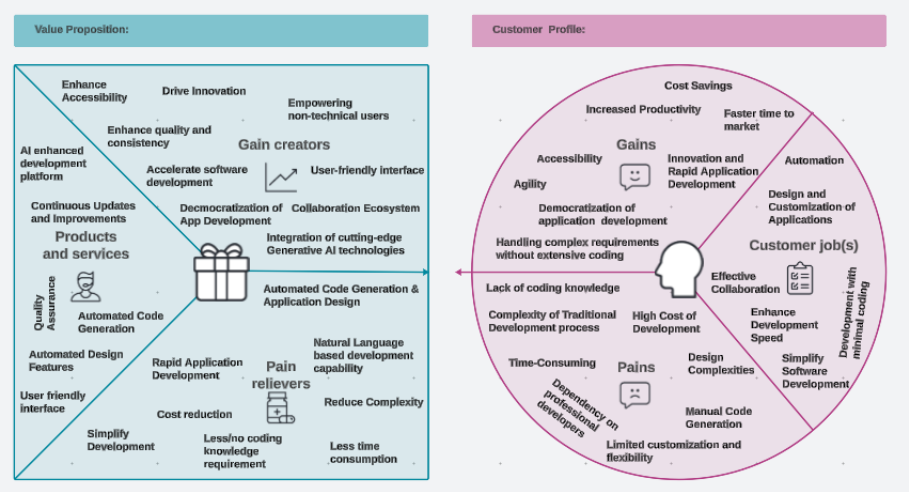

Figure 5.9: Value Proposition Canvas

- **Customer Profile:** In this section of the canvas, we identified the customer jobs that need to be done, the pains they face while performing the jobs, and also the gains they want to achieve from using the generative AI integrated platform.

  - **Customer Jobs:** One of the main things that need to be done is to simplify the software development process so that both technical and non-technical users can participate in the development process. Businesses want to speed up their development cycle so that they can reach the market faster. They need to have features for designing and customizing applications. Another important job is development with minimal coding, which reflects the aim of automation. Another customer job that needs to be done is having the opportunity to collaborate with others for the development process effectively.

  - **Pains:** In this part, we identified the pains or issues they face in completing the jobs that are required to be done. One of the main pains for the users is their lack of programming knowledge, which separates them from the development

of software. Traditional development is a complex process that is also time-consuming. Due to a lack of coding/development knowledge, the development of applications can be costly, and customers need to depend on professional developers. Moreover, designing an application and building it with manual coding is complex and has many challenges.

- **Gains:** Though there are several pains associated with jobs that require to be done by the users, there is still an opportunity for gains in the idea of integrating generative AI into low/no-code platforms. One of the main gains is making software development accessible to a broader audience. As programming or coding knowledge is not required to develop applications using these platforms, it creates opportunities for non-technical users to participate in the development process, which also democratizes software development. With the features of low/no-code platforms with generative AI capabilities, application development becomes faster, businesses can reach the market with their product very quickly, and it can improve code quality and enhance the user experience through AI-driven automation. Users can handle complex requirements without extensive coding. In a nutshell, productivity and efficiency can increase. This kind of platform can also reduce the cost of development.

- **Value Proposition:** In this section of the canvas, we tried to identify how our generative AI-integrated platform aligns with the offerings that meet the needs and exceed the expectations of the users. This segment mainly focuses on the benefits and features that make the platform attractive to users. Each element of this segment highlights how the platform not only addresses the challenges faced by the users but also adds value by enhancing the user experience and outcomes.

  - **Product and Services:** In this part of the segment, we outline the core features and offerings of our proposed platform. It will be a development environment enhanced by AI that enables users to create applications with little to no coding. The platform will automatically generate code based on simple user inputs, making the design task more easy for them. It will have a user-friendly interface so that user can comfortably use the platform for their application creation. It will ensure that the application created with the platform meets user requirements. The platform needs to be continuously updated with the newest features and improvements so that it has the newest facilities available for its users.

  - **Pain Relievers:** Here, the main focus was to identify how our platform with generative AI capabilities can address specific user pains with the software development process.For instance, the complexity and time needed for development may be greatly decreased with the use of generative artificial intelligence.

Users can use simple natural language to communicate with the system and provide their requirements and the system will automatically generate code and design for them, which reduces the complexities of the design task and manual coding as well as their knowledge of programming. The platform's automated code generation and intuitive design tool features may remove technical hurdles and learning curves from the development process, making it more accessible and less challenging for new users. It will also relieve the pain of depending on developers for developing applications and the cost associated with that.

– **Gain Creators:** This element of the value proposition segment emphasizes the additional benefits that our proposed platform can offer to our users. The platform will not only simplify the development process but also enhance user productivity and enable rapid prototyping. user can develop applications without extensive coding skills, rather in their natural language. This capability empowers non-technical users to participate in the development process. Users can quickly test and iterate on their ideas, which will accelerate the innovation process and bring applications to market faster. Additionally, organizations utilizing the platform may get a competitive edge by optimizing apps for performance and user experience thanks to its AI-driven insights.

5.2.6 Business Model Canvas

To ensure the long-term success and viability of our proposed low/no-code platform integrated with generative AI capabilities, we need to have a well-defined business model. For this reason, we made a business model canvas for our idea that serves as a roadmap. It outlines the key elements that will enable us to deliver value to our target users of the platform and achieve sustainable growth. It helped us define our customer segments, the value proposition we could offer, the channels through which we will reach the users, the cost required to implement and maintain the idea, the key resources and activities related to the idea, as well as the revenue stream that will fuel our operations. By carefully considering these elements and continuously updating our business model based on the market feed, we can serve a platform that will empower users of all skill levels to build innovative applications and participate in the software development process. A detailed description of all the elements of the business model canvas is given below:

- **Customer Segments:** In this element of the canvas, we outline the customer/user segment that we can work with our proposed platform. Basically, the users who will be using the platform for the developments. For our proposed platform, the customer segment includes enterprises that are looking to accelerate digital transformation, professional developers seeking to increase their productivity, citizen developers or non-technical users who lack coding knowledge, and independent software

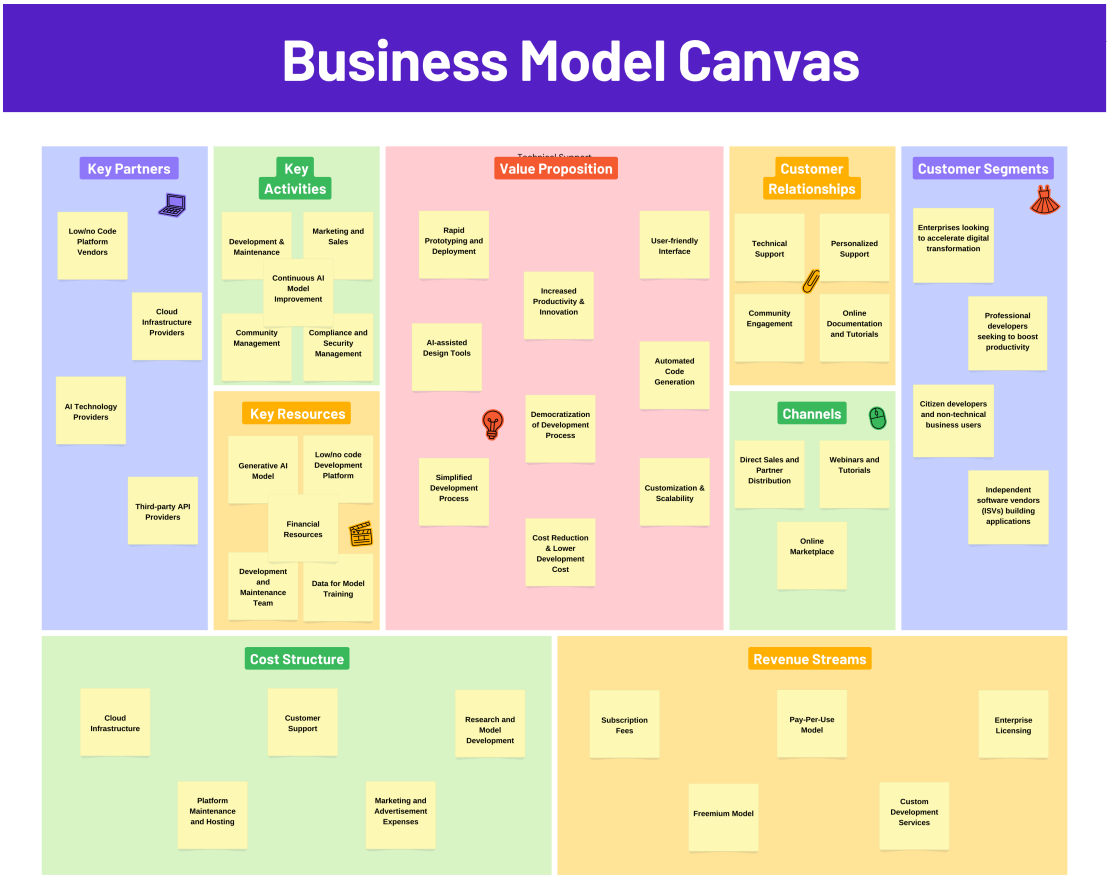

Figure 5.10: Business Model Canvas

vendors for building applications. Identifying and targeting these customer segments is crucial for the successful adoption and implementation of Generative AI integration.

- **Customer Relationships:** Here we identified the ways through which we can maintain the connectivity and relationship with the user of the platform. For our idea, we will try to give the user the required technical support as well as personalized support to navigate and build applications on the platform. Moreover, community engagement initiatives can also help to build a collaborative user base.

- **Channels:** In these elements of the canvas, we need to identify the channels through which users will receive the value proposition of the proposed platform. The platform can be accessed online, providing a central hub for development activities. Direct sales to enterprises, partnerships with platform vendors, and an online marketplace for individual developers can work as channels as well. Webinars and workshops can also be arranged to reach and engage with the target users effectively.

- **Value Proposition:** Here, we identified the value that the proposed integrated platform can provide to its users. These value propositions include accelerating software development by automating code generation and design tasks, simplifying the development process, empowering non-technical users to build applications efficiently, enhancing the quality and consistency of software output, and providing an intuitive, user-friendly interface so that users can seamlessly develop applications. With the availability of AI-assisted tools, productivity, and innovation opportunities can also be increased, which will enable rapid application development and reach to the market faster. This kind of platform will also reduce the time and cost required to develop an application in traditional ways. These value propositions intend to address the needs and desires of the target customer segment effectively.
- **Key Activities:** Key activities for the successful implementation of the idea include developing AI models for code generation and design, integrating AI capabilities into the platform, providing user training and support, and continuous update and improvement of the AI model based on user feedback. Marketing and sales of the product efficiently also fall under the key activities. Moreover, managing the compliance and security aspects as well as community management falls under the key activities. These activities are essential for ensuring seamless integration of the idea.
- **Key Resources:** Key resources for the proposed platform include a generative AI model, low/no-code development platform, data for training the model, and a development and management team that will work to develop and maintain the functionalities and features of the platform. Financial resources like investment and funding on research and development for new and updated AI technology or models also fall under the key resources.
- **Key Partners:** Key partners include low/no-code platform vendors, cloud infrastructure providers, AI technology providers, and third-party API providers for other features and functionalities to include in the platform. It is crucial to collaborate with these partners to effectively leverage their expertise and resources for the integration of generative AI into the development platform.
- **Cost Structure:** The cost structure includes the expenses required for the development of the AI model, cloud infrastructure, research for new innovations and development, customer support and success, platform maintenance, hosting, and marketing and advertisement-related expenses. It is really crucial to understand and manage the cost structure to ensure the financial sustainability of the idea.
- **Revenue Streams:** Revenue streams can be generated from subscription fees based on usage and features, a freemium model that allows users to try basic functionalities for free before committing to a subscription; and and custom development services for enterprises that require tailored solutions. The key to increasing sustainability and profitability is income stream diversification.

5.3 Requirements

The following section will specify the requirements of the various tasks that can be carried out by integrating a GPT-based chatbot into low-code platforms. Requirements are generalized and based on the previous sections—state-of-the-art, business, and technical analysis.

Requirement	Description	Rationale
Assistance and Guidance	The GPT-based chatbot should provide help and guidance when developers encounter difficulties based on a conversational manner.	To further enhance democratization of software development and support developers in overcoming obstacles quickly.
Enhanced Code Automation	The GPT-based chatbot should automate the generation of complex code structures that adapt to user needs and project contexts.	To reduce manual coding requirements and speed up development processes, making the platform more efficient and accessible.
Dynamic UI/UX Suggestions	The GPT-based chatbot will provide actionable recommendations for UI/UX designs based on user behavior and industry standards.	To assist developers in creating user-friendly interfaces that enhance user satisfaction and engagement.
Customization	The GPT-based chatbot should provide features that enhance the flexibility of low-code platforms, allowing developers to adapt and extend pre-built components and templates to meet unique project requirements.	To ensure that the AI-generated content aligns with the unique requirements of different projects, enhancing the flexibility and usability of the low-code platform.
AI-Enhanced Integration Tools	Develop AI tools that simplify the process of integrating the LCNDP with other systems and technologies.	To enhance the capability of LCNDPs to handle complex integrations smoothly, thus extending their applicability.
Automated Testing and Quality Assurance	AI should automatically generate and execute test cases to ensure application functionality and reliability.	To maintain high quality standards with reduced manual intervention, increasing the reliability of deployed applications.
Intelligent Error Handling and Debugging	Implement AI-driven mechanisms to identify, analyze, and rectify errors in real-time.	Enhances productivity by reducing downtime and manual debugging efforts, thus improving code quality and reliability.

Table 5.6: Functional Requirements

Chapter 6

Discussion & Future Work

In this discussion chapter, we will reflect on the decisions made throughout the thesis and discuss the process of conducting a technical and business analysis for our thesis report. Furthermore, we will explore potential future directions and aspects of this research.

In our background chapter, we explored the evolution of software development methodologies by highlighting how traditional methods like the waterfall model and iterative methods like the spiral model provided a structure for software development but lacked flexibility. Agile methodology emerged to handle the lack of traditional methodologies and introduced more adaptability and user-centric approaches to software development. However, even these methodologies had issues, such as the dependency on skilled developers and extensive coding, which highlighted the need for more accessible and efficient methodologies for software development. This need paved the way for the rise for low/no-code development and generative AI.

Then in our state-of-the-art chapter, we explored the current state of low/no-code development and generative AI technologies that helped us to understand how these platforms can democratize software development by allowing users with limited or no coding knowledge to the software development processes. We got to understand their functionalities such as visual development environments and pre-built components, model-driven development approach which makes the development process faster and less costly. In addition to this, the exploration of generative AI technologies and their capabilities, we could grasp a better understanding of how they can enhance the capabilities of low/no-code platforms. Together, these technologies can make software development more accessible and efficient, promoting innovation and enabling larger number of people to participate in the development process.

The stakeholder analysis helped us to identify the relevant stakeholders associated with the possible integration of low/no-code development platforms and generative AI. Our

business analysis provided us insights of the possible business opportunities and challenges associated with the possible collaboration of these two technologies. We could understand the strengths, weaknesses, opportunities, and threats of the integration by performing the swot analysis. We also performed porter's five forces framework to understand the competitive market scenario and possible challenges. In our value proposition canvas, we identified the key challenges and issues that the users or businesses face now regarding software development and how these two technologies can come as a remedy to these issues and create value for the users. To analyze the business potential of integrating generative AI with low/no-code platforms, we created a business model canvas and identified its nine components.

In the technical analysis, we explored the workings of low-code platforms, which are built on a layered architecture where each layer handles specific operations. The key layers include the application layer, service integration layer, data integration layer, and deployment layer. We then defined the low-code development life cycle to understand how and where generative AI can be optimally integrated into the software development process. This lifecycle mirrors traditional software development but is designed to enhance development speed and efficiency.

We identified six stages based on the agile methodology and pinpointed potential limitations and challenges developers might encounter. After defining these challenges, we outlined how generative AI can provide solutions, concluding that generative AI can be beneficial throughout the entire development process, from requirement specification to deployment. We suggest that integrating a GPT-based chatbot as an AI assistant within the low-code platform could address many of these limitations.

However, several issues arose during our research. Ideally, we would have collected primary data from both experts (to identify the potential of generative AI) and end users (citizen developers) to understand the challenges from their perspective. Although we received a response from one company, we were unable to secure their collaboration for an interview. Additionally, due to time constraints, we did not conduct end-user testing of low-code platforms, leading us to rely on limited research given the novelty of the topic.

While we concluded that generative AI is essential throughout the development stages and identified its potential benefits, further research is needed to draw more concrete conclusions on the use of generative AI as a GPT-based chatbot, particularly concerning the validity and accuracy of its responses. Additionally, more exploration is needed to identify other implementation methods for generative AI and to offer additional recommendations on how it can enhance low-code platforms.

Another important area of research involves optimizing the way developers prompt the

chatbot, as this will be the developer direct interaction with the platform and this is crucial for effectively using the platform. Proper prompting techniques can significantly enhance the developer's experience and the overall efficiency of the low-code development process.

Moreover, our research lacks a detailed explanation of the technical implementation of generative AI within low-code platforms. Therefore, a key area for future research is to explore the technical aspects of integrating the GPT-based chat bot into low-code development environments.

Chapter 7

Conclusion

This thesis explored the transformative potential of low/no-code development enhanced by generative AI technology, addressing the main statement:

"How will the offering of low/no-code solutions, enhanced by generative AI technology, affect the future of software development?"

Following the main problem statement, some research sub-questions were also formulated to break it down into smaller parts so that we can effectively approach the problem statement. The first sub-question is as follows:

"How have the limitations of traditional and modern software development methodologies influenced the rise of low/no-code platforms?"

Our research began by exploring the evolution of different software development methodologies in the background chapter where we highlighted the transition from traditional methodologies like waterfall and spiral models to modern methods like agile. This provided us with a better understanding of these models and how they often lacked flexibility and adaptability, which emphasized the need for more accessible and efficient way of developing software. Hence, the rise of low/no code development began, which aim to simplify and democratize the development process by making it accessible to non-technical users.

Then for the second sub-question:

"What are the core components and key features that define a development approach as "low/no-code" development?"

In our state-of-the-art chapter, we explored the functionalities and characteristics of low/no

code development platforms. We identified the core components of these low/no code platforms that include visual development environments, pre-built components, and drag-and-drop interfaces that make these platforms accessible to users with minimal coding and software development knowledge. We understood about the features of these platforms, how they make the development process easy with their functionalities as well as the challenges and limitation they currently face.

Our third research sub-question is:

"How can the integration of generative AI across the stages of the low code development process enhance its capabilities?"

To address this sub-question regarding the integration of generative AI, our initial focus was on defining the low-code development process. Thus, we examined the current landscape of software development methodologies within the low-code context. Our investigation revealed that low-code development aligns with agile principles but operates within condensed and more efficient phases. Building upon this understanding, we established six key stages: data modeling, user interface design, business logic implementation, integration of external services, testing and deployment, and maintenance.

Subsequently, we sought to pinpoint challenges encountered by developers to assess whether gen AI could alleviate these obstacles. Our analysis concluded with that gen AI integration can span the entirety of the development cycle. Through outlining the various tasks of generative AI within this process, it became evident that one viable implementation would be through a GPT-based chatbot. This approach offers users both a visual low-code environment and a textual interface via the GPT bot. Such integration has the potential to significantly enhance and democratize software development

The last research sub-question is as follows:

"How does integrating generative AI enhance the business value and competitive market positioning of low/no-code development platforms?"

To address this sub-question, we evaluated the market scenario and strategic implications of integrating generative AI with low/no-code platforms in our business analysis section. We conducted a market analysis, SWOT analysis, Porter's Five Forces Analysis and used the value proposition and business model canvas to understand the business impact. Our exploration revealed that generative AI integration into low/no-code development platforms can enhance the business value of these platforms by accelerating the development process, reducing costs, and improving software quality. This integration also positions these platforms competitively in the market by offering unique value propositions such

as enhanced productivity, greater innovation potential, and accessibility for non-technical users. These benefits create substantial opportunities for businesses to leverage these technologies for digital transformation and competitive advantage.

The integration of generative AI into low/no-code platforms represents a significant advancement in the field of software development. This combination has the potential to overcome current limitations, enhance productivity, and further democratize software development reshaping the future of software development. While the research shows promise, future studies should focus on the technical implementation of generative AI within low-code environments and gather primary data from industry experts and end-users to validate these findings.

Bibliography

[1] K. P., *Revolutionizing software development: The convergence of generative ai, no-code/low-code, and agile methodologies*, https://www.linkedin.com/pulse/revolutionizing-software-development-convergence-generative-patel-epwwc/, Accessed: 03-03-2024, Feb 14, 2024.

[2] N. Byers, *Empowering innovation: The rise of low-code and no-code development*, https://medium.com/@nathanbyers13/empowering-innovation-the-rise-of-low-code-and-no-code-development-e4d1d82a5ead, Accessed: 03-03-2024, Feb 19, 2024.

[3] P. Gomes and M. Brito, "Low-code development platforms: A descriptive study," Jun. 2022, pp. 1–4. DOI: 10.23919/CISTI54924.2022.9820354.

[4] G. V. Research, *Gvr report cover low-code development platform market size, share trends report low-code development platform market size, share trends analysis report by application type (web-based, mobile-based), by deployment type (cloud, on-premise), by organization size, by end-use, by region, and segment forecasts, 2023 - 2030*, https://www.grandviewresearch.com/industry-analysis/low-code-development-platform-market-report#, Accessed: 03-03-2024, 2023.

[5] N. Sabharwal, *How low-code development helps enterprises*, https://www.forbes.com/sites/forbestechcouncil/2022/10/11/how-low-code-development-helps-enterprises/, Accessed: 03-03-2024, Oct 11, 2022.

[6] Cyntexa, *How low code or no code revolutionize the future of software development*, https://cyntexa.com/blog/low-code-no-code-the-future-of-software-development/, Accessed: 03-03-2024, January 23, 2024.

[7] M. Brundage, S. Avin, J. Clark, *et al.*, "The malicious use of artificial intelligence: Forecasting, prevention, and mitigation," *arXiv preprint arXiv:1802.07228*, 2018.

[8] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[9] R. Ulfsnes, N. Moe, V. Stray, and M. Skarpen, "Transforming software development with generative ai: Empirical insights on collaboration and workflow," Feb. 2024.

[10] krishna philips, *The future of software development trends innovations*, https://medium.com/@philips202308/the-future-of-software-development-trends-innovations-a27b3fac9902, Accessed: 03-03-2024, Feb 19, 2024.

[11] P. Naur and B. Randell, *Software engineering: Report on a conference*. NATO Science Committee, 1969.

[12] W. Royce, "Managing the development of large software systems," in *Proceedings of IEEE WESCON*, IEEE, 1970, pp. 1–9.

[13] H. Benington, "Production of large computer programs," *IEEE Annals of the History of Computing*, vol. 5, no. 4, pp. 350–361, 1983.

[14] B. Boehm, "A spiral model of software development and enhancement," *ACM SIGSOFT Software Engineering Notes*, vol. 11, no. 4, pp. 14–24, 1986.

[15] J. Martin, *Rapid Application Development*. Macmillan Publishing Co., Inc., 1991.

[16] K. Beck *et al.*, *Manifesto for agile software development*, https://agilemanifesto.org/, 2001.

[17] T. Dingsøyr, S. Nerur, V. Balijepally, and N. Moe, "A decade of agile methodologies: Towards explaining agile software development," *Journal of Systems and Software*, vol. 85, no. 6, pp. 1213–1221, 2012.

[18] L. Bass, I. Weber, and L. Zhu, *DevOps: A software architect's perspective*. Addison-Wesley Professional, 2015.

[19] M. Brambilla, J. Cabot, and M. Wimmer, *Model-driven software engineering in practice*. Morgan & Claypool Publishers, 2017, vol. 2.

[20] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Professional, 2010.

[21] S. Hendrickson, S. Sturdevant, T. Harter, V. Venkataramani, A. Arpaci-Dusseau, and R. Arpaci-Dusseau, "Serverless computation with openlambda," in *Proceedings of the 8th USENIX Conference on Hot Topics in Cloud Computing (HotCloud '16)*, 2016.

[22] J. Saltz and K. Crowston, "Exploring the challenges of low/no-code development for non-it users," in *Proceedings of the 53rd Hawaii International Conference on System Sciences*, 2020, pp. 5984–5993.

[23] M. Bano, D. Zowghi, N. Sarkissian, W. Lowe, and K. De Salas, "Ai-assisted software development: A systematic literature review," *Journal of Systems and Software*, vol. 182, p. 111 075, 2021.

[24] Wikipedia, *Software development process*, https://en.wikipedia.org/wiki/Software_development_process, Accessed: 08-03-2024.

[25] L. Sherrell, "Waterfall model," in *Encyclopedia of Sciences and Religions*, A. L. C. Runehov and L. Oviedo, Eds. Dordrecht: Springer Netherlands, 2013, pp. 2343–2344, ISBN: 978-1-4020-8265-8. DOI: 10.1007/978-1-4020-8265-8_200285. [Online]. Available: https://doi.org/10.1007/978-1-4020-8265-8_200285.

[26] J. Adam, *What is the waterfall software development methodology and is it still relevant?* https://kruschecompany.com/waterfall-software-development-methodology/, Accessed: 22-05-2024, MARCH 29, 2024.

[27] M. Cohn, S. Sim, and C. Lee, "What counts as software process? negotiating the boundary of software work through artifacts and conversation," *Computer Supported Cooperative Work*, vol. 18, pp. 401–443, Dec. 2009. DOI: 10.1007/s10606-009-9100-4.

[28] K. Petersen, C. Wohlin, and D. Baca, "The waterfall model in large-scale development," in *Product-Focused Software Process Improvement: 10th International Conference, PROFES 2009, Oulu, Finland, June 15-17, 2009. Proceedings 10*, Springer, 2009, pp. 386–400.

[29] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, 1988.

[30] GeeksForGeeks, *What is spiral model in software engineering?* https://www.geeksforgeeks.org/software-engineering-spiral-model/, Accessed: 24-05-2024, 16 May, 2024.

[31] Javapoint, *V-model*, https://www.javatpoint.com/software-engineering-v-model, Accessed: 25-05-2024.

[32] O. Moravcik, T. Skripcak, D. Petrik, and P. Schreiber, "Approaches of the modern software development," *International Journal of Machine Learning and Computing*, vol. 1, no. 5, p. 479, 2011.

[33] A. I. Wasserman, "Modern software development methodologies and their environments," *Computer Physics Communications*, vol. 38, no. 2, pp. 119–134, 1985, ISSN: 0010-4655. DOI: https://doi.org/10.1016/0010-4655(85)90079-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0010465585900797.

[34] A. Moniruzzaman and D. S. A. Hossain, "Comparative study on agile software development methodologies," *arXiv preprint arXiv:1307.3356*, 2013.

[35] Wikipedia, *Rational unified process*, https://en.wikipedia.org/wiki/Rational_unified_process, Accessed: 27-05-2024.

[36] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*. Addison-Wesley, 1999.

[37] P. Kruchten, *The Rational Unified Process: An Introduction*, 3rd. Addison-Wesley Professional, 2003.

[38] B. Boehm, "A spiral model of software development and enhancement," *ACM SIGSOFT Software Engineering Notes*, vol. 11, no. 4, pp. 14–24, 1988.

[39] A. Manifesto, "Manifesto for agile software development," 2001.

[40] A. Alliance, *The 12 principles behind the agile manifesto*, `https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/`, Accessed: 28-05-2024.

[41] M. Clesham, *5 stages of the agile system development life cycle*, `https://www.brightwork.com/blog/5-stages-of-the-agile-system-development-life-cycle`, Accessed: 28-05-2024, November 9, 2023.

[42] N. Sekulic, *6 stages of the agile software development lifecycle*, `https://www.gitkraken.com/blog/6-stages-of-agile-development`, Accessed: 28-05-2024, April 28, 2023.

[43] F. Capital, *The most common and popular agile development frameworks, such as scrum, kanban, xp, and lean*, `https://fastercapital.com/topics/the-most-common-and-popular-agile-development-frameworks,-such-as-scrum,-kanban,-xp,-and-lean.html`, Accessed: 28-05-2024.

[44] P. Serrador and J. K. Pinto, "Does agile work? a quantitative analysis of agile project success," *International Journal of Project Management*, vol. 33, no. 5, pp. 1040–1051, 2015.

[45] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe, "A decade of agile methodologies: Towards explaining agile software development," *Journal of Systems and Software*, vol. 85, no. 6, pp. 1213–1221, 2012.

[46] K. Beck, *Extreme Programming Explained: Embrace Change*, 2nd. Addison-Wesley Professional, 2004.

[47] J. Holvikivi, "Agile and lean business development frameworks," *Journal of Software: Evolution and Process*, vol. 28, no. 11, pp. 935–946, 2016.

[48] M. Laanti, O. Salo, and P. Abrahamsson, "Agile methods rapidly replacing traditional methods at nokia: A survey of opinions on agile transformation," *Information and Software Technology*, vol. 53, no. 3, pp. 276–290, 2011.

[49] B. Fitzgerald, G. Hartnett, and K. Conboy, "Customising agile methods to software practices at intel shannon," *European Journal of Information Systems*, vol. 15, no. 2, pp. 200–213, 2006.

[50] M. Paasivaara, B. Behm, C. Lassenius, and M. Hallikainen, "Large-scale agile transformation at ericsson: A case study,"
Empirical Software Engineering, vol. 23, no. 1, pp. 255–289, 2018.

[51] M. Fowler, "The new methodology,"
Wuhan University Journal of Natural Sciences, vol. 6, pp. 12–24, 2001.
[Online]. Available: https://api.semanticscholar.org/CorpusID:20754389.

[52] D. C. Schmidt *et al.*, "Model-driven engineering,"
Computer-IEEE Computer Society-, vol. 39, no. 2, p. 25, 2006.

[53] EZOFIS, *The evolution of low-code platforms*,
https://www.linkedin.com/pulse/evolution-low-code-platforms-ezofis/,
Accessed: 25-03-2024, September 11, 2023.

[54] C. R. John Rymer, *The forrester wave™: Low-code development platforms, q2 2016*,
https://www.forrester.com/report/The-Forrester-Wave-LowCode-Development-Platforms-Q2-2016/RES117623, Accessed: 28-03-2024,
April 14th, 2016.

[55] S. Dipanshu, *Navigating the future: Predictions for low code and generative ai redefining technology in 2024*, https://medium.com/@shekhar.dipanshu6/navigating-the-future-predictions-for-low-code-and-generative-ai-redefining-technology-in-2024-da5f5fb7b7f9, Accessed: 28-03-2024, Jan 5, 2024.

[56] T. Kissflow, *The history of low-code platforms : How development changed*, https://kissflow.com/low-code/history-of-low-code-development-platforms/,
Accessed: 07-03-2024, 23 Feb 2024.

[57] IBM, *What is low-code?* https://www.ibm.com/topics/low-code,
Accessed: 19-03-2024.

[58] M. DiCesare, *Model-driven development: The foundation of low-code*, https://www.mendix.com/blog/low-code-principle-1-model-driven-development/,
Accessed: 31-03-2024, March 1, 2024.

[59] F. Alexander, *State of app development report results: The future looks bright*,
https://www.outsystems.com/blog/posts/state-app-dev-highlights/,
Accessed: 18-03-2024, October 19, 2023.

[60] R. K. John Rymer,
The forrester wave™: Low-code development platforms for add professionals, q1 2019,
https://www.forrester.com/report/the-forrester-wave-low-code-development-platforms-for-add-professionals-q1-2019/RES144387,
Accessed: 19-03-2024, March 13th, 2019.

[61] R. Ellis, *Why 83% of it leaders are banking on low-code*,
https://www.salesforce.com/news/stories/why-it-leaders-are-banking-on-low-code/, Accessed: 19-03-2024, JUNE 15, 2021.

[62] KPMG, *Shaping digital transformation with low-code platforms*, https://kpmg.com/xx/en/home/insights/2023/02/shaping-digital-transformation-with-low-code-platforms.html, Accessed: 19-03-2024.

[63] S. D. C. Misra, *Understanding low-code no-code (lcnc) platforms*, https://www.nic.in/blogs/understanding-low-code-no-code-lcnc-platforms/, Accessed: 19-03-2024, February 15th, 2024.

[64] M. Clausse, *How artificial intelligence is impacting low-code and no-code platforms*, https://www.planetcrust.com/how-artificial-intelligence-is-impacting-low-code-and-no-code-platforms?utm_campaign=blog, Accessed: 19-03-2024, November 6, 2023.

[65] DEVOPSdigest, *Differences between low-code and no-code development*, https://www.geeksforgeeks.org/low-code-vs-no-code-development/, Accessed: 06-04-2024, 24 Jan, 2023.

[66] OutSystems, *The transformative power of low-code apps*, https://www.outsystems.com/tech-hub/low-code/apps/#common-low-code-use-cases, Accessed: 15-04-2024.

[67] Q. E. Team, *10+ low-code use cases: When to use low-code development approach?* https://quixy.com/blog/low-code-use-cases/, Accessed: 15-04-2024, October 11, 2023.

[68] D. Sá, T. Guimarães, A. Abelha, and M. F. Santos, "Low code approach for business analytics,"
Procedia Computer Science, vol. 231, pp. 421–426, 2024, 14th International Conference on Emerging Ubiquitous Systems and Pervasive Networks / 13th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (EUSPN/ICTH 2023), ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2023.12.228. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050923022408.

[69] outsystems, *What is no-code and when should you use it*, https://www.outsystems.com/tech-hub/no-code/#how-does-it-work, Accessed: 19-04-2024.

[70] Low/Code, *Capabilities and limitations of no-code/low-code development platforms*, https://www.lowcode.agency/blog/capabilities-and-limitations-of-no-code-low-code-development-platforms, Accessed: 20-04-2024.

[71] N. E. Education, *5 pros and cons of no-code development*, https://northwest.education/insights/careers/5-pros-and-cons-of-no-code-development/, Accessed: 20-04-2024, September 21, 2023.

[72] M. S. Matt Sadowski, *No code / low code vs. custom traditional development*, https://themobilereality.com/blog/no-code-low-code-vs-traditional-development, Accessed: 20-04-2024, 14.03.2024.

[73] SAP, *What is low-code/no-code application development?* https://www.sap.com/products/technology-platform/low-code/what-is-low-code-no-code.html, Accessed: 30-03-2024.

[74] OutSystems, *Demystifying visual programming*, https://www.outsystems.com/tech-hub/app-dev/understanding-visual-programming-language/#role-of-visual-programming, Accessed: 31-03-2024.

[75] DEVOPSdigest, *2024 low-code/no-code predictions*, https://shorturl.at/esvDW, Accessed: 31-03-2024, January 04, 2024.

[76] D. Partida, *Trends in low-code/no-code*, https://www.datamation.com/trends/trends-in-low-code-no-code/, Accessed: 20-04-2024, May 8, 2023.

[77] G. Brisk, *10 no-code and low-code trends to look out for*, https://baserow.io/blog/low-code-no-code-trends, Accessed: 20-04-2024, January 18, 2024.

[78] D. A. James Oluwaleye, *Low-code and no-code development tools*, https://semaphoreci.com/blog/low-code-no-code-development-tools, Accessed: 20-04-2024, 7 Jun 2023.

[79] J. McKendrick, *Low-code and no-code development gets a makeover as priorities shift to ai*, https://www.zdnet.com/article/low-code-and-no-code-development-gets-a-makeover-as-priorities-shift-to-ai/, Accessed: 20-04-2024, Feb. 13, 2024.

[80] A. Ribeiro and A. R. da Silva, "Survey on cross-platforms and languages for mobile apps," in *2012 Eighth International Conference on the Quality of Information and Communications Technology*, Ieee, 2012, pp. 255–260.

[81] outsystems, *Deploying outsystems - overview*, https://www.outsystems.com/evaluation-guide/deploying-outsystems/, Accessed: 02-05-2024.

[82] ——, *The outsystems enterprise security posture*, https://www.outsystems.com/evaluation-guide/security/enterprise/, Accessed: 02-05-2024.

[83] ——, *Outsystems developer community*, https://www.outsystems.com/our-community/, Accessed: 02-05-2024.

[84] ——, *Release notes*, https://success.outsystems.com/support/release_notes/, Accessed: 02-05-2024, Dec 21, 2023.

[85] ——, *Platform architecture overview*, https://www.outsystems.com/evaluation-guide/architecture/, Accessed: 07-05-2024.

[86] outsystems community, *What are the disadvantages of outsystems?* https://www.outsystems.com/forums/discussion/62038/what-are-the-disadvantages-of-outsystems/, Accessed: 02-05-2024, Dec 21, 2023.

[87] Joget, *Joget dx introduction*, https://www.joget.org/product/joget-dx/, Accessed: 04-05-2024.

[88] Julian, *Introducing joget dx, the next generation open source digital transformation platform*, https://blog.joget.org/2018/12/introducing-joget-dx-next-generation.html?m=1, Accessed: 04-05-2024, December 07, 2018.

[89] J. Cloud, *Frequently asked questions*, https://www.jogetcloud.com/faq.html, Accessed: 04-05-2024, December 07, 2018.

[90] I. technology, *Joget dx 8 and its new feature*, https://iqratechnology.com/joget-dx-8-and-its-new-feature/, Accessed: 04-05-2024, December 07, 2018.

[91] mendix, *2023 magic quadrant™ for enterprise low-code application platforms by gartner® recognizes mendix as a leader for fourth consecutive time*, https://www.mendix.com/press/2023-magic-quadrant-for-enterprise-low-code-application-platforms-by-gartner-recognizes-mendix-as-a-leader-for-fourth-consecutive-time/, Accessed: 05-05-2024.

[92] J. Pietsch, *Discover top 15 mendix low-code platform features*, https://www.netguru.com/blog/mendix-features, Accessed: 05-05-2024, Apr 30, 2024.

[93] Mendix, *Architecture principles*, https://www.mendix.com/evaluation-guide/enterprise-capabilities/architecture-principles/#key-principles, Accessed: 07-05-2024.

[94] ——, *Platform architecture*, https://www.mendix.com/evaluation-guide/enterprise-capabilities/platform-architecture/, Accessed: 07-05-2024.

[95] M. Community, *Challenges and limitations of mendix*, https://community.mendix.com/link/space/app-development/questions/110047, Accessed: 07-05-2024.

[96] Mendix, *Mendix review*, https://www.trustradius.com/products/mendix/reviews?qs=pros-and-cons#overview, Accessed: 07-05-2024.

[97] B. Basgen, *A generative ai primer*, https://er.educause.edu/articles/2023/8/a-generative-ai-primer, Accessed: 17-05-2024, August 15, 2023.

[98] M. Webb, *A generative ai primer*, https://nationalcentreforai.jiscinvolve.org/wp/2024/03/04/generative-ai-primer/, Accessed: 17-05-2024, 4 March 2024.

[99] A. Bandi, P. V. S. R. Adapa, and Y. E. V. P. K. Kuchi, "The power of generative ai: A review of requirements, models, input–output formats, evaluation metrics, and challenges," *Future Internet*, vol. 15, no. 8, p. 260, 2023.

[100] T. U. Team, *10 benefits of generative ai: Increase productivity and creativity*, https://www.upwork.com/resources/generative-ai-benefits, Accessed: 17-05-2024, Sep 20, 2023.

[101] Dive, *Generative ai: Benefits, use cases, and examples*, https://www.letsdive.io/blog/generative-ai-benefits-use-cases-and-examples, Accessed: 17-05-2024.

[102] T. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[103] L. Ouyang, J. Wu, X. Jiang, *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[104] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[105] A. Ramesh, M. Pavlov, G. Goh, *et al.*, "Zero-shot text-to-image generation," in *International conference on machine learning*, Pmlr, 2021, pp. 8821–8831.

[106] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[107] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," *Advances in neural information processing systems*, vol. 29, 2016.

[108] A. JANSAUSKIS, *How AI tools impact the way we develop software: our GitHub Copilot journey*, https://www.emergn.com/insights/how-ai-tools-impact-the-way-we-develop-software-our-github-copilot-journey/, [Online; accessed 18-May-2024].

[109] A. Lee, *8 Ethical Challenges For Generative AI*, https://www.forbes.com/sites/amazon-web-services-asean/2024/05/17/8-ethical-challenges-for-generative-ai/, [Online; accessed 18-May-2024], May 17, 2024.

[110] E. Cetin, *How is the growth in GenAI changing low-code and no-code development?* https://www.ciklum.com/resources/blog/how-is-the-growth-in-genai-changing-low-code-and-no-code-development, [Online; accessed 20-May-2024], March 4th 2024.

[111] M. Chen, J. Tworek, H. Jun, *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[112] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning,"
ACM computing surveys (CSUR), vol. 54, no. 6, pp. 1–35, 2021.

[113] R. Cunningham, *Power Apps is empowering coders with next-generation AI capabilities*, https://powerapps.microsoft.com/en-gb/blog/power-apps-is-empowering-coders-with-next-generation-ai-capabilities/,
[Online; accessed 20-May-2024], 23 May 2023.

[114] J. Barr,
Introducing Amazon Honeycode – Build Web Mobile Apps Without Writing Code, https://aws.amazon.com/blogs/aws/introducing-amazon-honeycode-build-web-mobile-apps-without-writing-code/, [Online; accessed 20-May-2024],
24 JUN 2020.

[115] BOSTON, *Mendix Adds Powerful New AI and Machine Learning Capabilities to its Market and Technology-Leading Enterprise Low-Code Platform*, https://www.mendix.com/press/mendix-adds-powerful-new-ai-and-machine-learning-capabilities-to-its-market-and-technology-leading-enterprise-low-code-platform/, [Online; accessed 20-May-2024], June 22, 2023.

[116] Salesforce, *Salesforce Artificial Intelligence*, https://www.salesforce.com/eu/artificial-intelligence/,
[Online; accessed 20-May-2024].

[117] GitHub, *The world's most widely adopted AI developer tool.* https://github.com/features/copilot, [Online; accessed 20-May-2024].

[118] A. LOZINSKI,
The Growing Impact of Generative AI on Low-Code/No-Code Development, https://devops.com/the-growing-impact-of-generative-ai-on-low-code-no-code-development/, [Online; accessed 21-May-2024], OCTOBER 17, 2023.

[119] A. Ghoshal, *Why generative AI will turbocharge low-code and no-code development*, https://www.infoworld.com/article/3694173/why-generative-ai-will-turbocharge-low-code-and-no-code-development.html,
[Online; accessed 20-May-2024], APR 21, 2023.

[120] K. Rokis and M. Kirikova, "Exploring low-code development: A comprehensive literature review,"
Complex Systems Informatics and Modeling Quarterly, vol. 0, pp. 68–86, 2023.
[Online]. Available:
https://csimq-journals.rtu.lv/article/view/csimq.2023-36.04/3345 (visited on 05/21/2024).

[121] A. S. A. I. D. D. Ruscio, *Supporting the understanding and comparison of low-code development platforms. 2020 46th euromicro conference on software engineering and advanced applications (seaa) | 10.1109/seaa51224.2020.00036*, [Online; accessed 21-May-2024], 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9226356 (visited on 05/20/2024).

[122] J. Martins, F. Branco, and H. Mamede, "Combining low-code development with chatgpt to novel no-code approaches: A focus-group study," *Intelligent Systems with Applications*, vol. 20, p. 200 289, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S266730532300114X (visited on 05/21/2024).

[123] A. A. Alamin, S. Malakar, G. Uddin, S. Afroz, T. B. Haider, and A. Iqbal, "An empirical study of developer discussions on low-code software development challenges," *arXiv (Cornell University)*, May 2021. DOI: 10.1109/msr52588.2021.00018. [Online]. Available: https://ieeexplore.ieee.org/document/9463132 (visited on 05/21/2024).

[124] R. Waszkowski, "Low-code platform for automating business processes in manufacturing," *IFAC-PapersOnLine*, vol. 52, no. 10, pp. 376–381, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896319309152 (visited on 05/21/2024).

[125] R. Arora, N. Ghosh, and T. Mondal, "Sagitec software studio (s3) - a low code application development platform," in *2020 International Conference on Industry 4.0 Technology (I4Tech)*, 2020. DOI: 10.1109/I4Tech48345.2020.9102703.

[126] N. Jesse, "Agility eats legacy–the long good-bye," in *Proc. of 19th IFAC Conference on Technology, Culture and International Stability, North Holland: Elsevier*, Elsevier, 2019.

[127] A. Jacinto, M. Lourenço, and C. Ferreira, "Test mocks for low-code applications built with outsystems," 2020. DOI: 10.1145/3417990.3420209. [Online]. Available: https://doi.org/10.1145/3417990.3420209.

[128] C. Di Sipio, D. Di Ruscio, and P. T. Nguyen, "Democratizing the development of recommender systems by means of low-code platforms," 2020. DOI: 10.1145/3417990.3420202. [Online]. Available: https://doi.org/10.1145/3417990.3420202.

[129] K. Rokis and M. Kirikova, "An archimate-based thematic knowledge graph for low-code software development domain," in *European Conference on Advances in Databases and Information Systems*, Springer,

2023, pp. 465–476. [Online]. Available:
https://link.springer.com/chapter/10.1007/978-3-031-42941-5_40.

[130] R. Martins, F. Caldeira, F. Sa, M. Abbasi, and P. Martins,
"An overview on how to develop a low-code application using outsystems,"
in *2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*, IEEE, 2020, pp. 395–401.
[Online]. Available: https://ieeexplore.ieee.org/abstract/document/9277404?casa_token=axuaTBr637YAAAAA:H8_NViHAkm0TImJisvKV-B3ka82ocgkR7zwAj34_W-6tP5-85-19fP-M7HYOtm--RTpLQvMslTE (visited on 05/21/2024).

[131] K. Rokis and M. Kirikova,
"Challenges of low-code/no-code software development: A literature review,"
in *International Conference on Business Informatics Research*, Springer, 2022, pp. 3–17.

[132] A. Esposito, M. Calvano, A. Curci, *et al.*,
"End-user development for artificial intelligence: A systematic literature review,"
2023, pp. 19–34. [Online]. Available:
https://link.springer.com/chapter/10.1007/978-3-031-34433-6_2.

[133] Y. K. Dwivedi, N. Kshetri, L. Hughes, *et al.*, ""so what if chatgpt wrote it?" multidisciplinary perspectives on opportunities, challenges and implications of generative conversational ai for research, practice and policy,"
International Journal of Information Management, vol. 71, p. 102 642, 2023.
[Online]. Available:
https://www.sciencedirect.com/science/article/pii/S0268401223000233.

[134] J. Osman, *Understanding the lifecycle of a low-code project*, @appmaster$_io$, Jul. 2023.
[Online]. Available:
https://appmaster.io/blog/lifecycle-of-a-low-code-project (visited on 05/30/2024).

[135] Y. Luo, P. Liang, C. Wang, M. Shahin, and J. Zhan, "Characteristics and challenges of low-code development: The practitioners' perspective," Oct. 2021.
DOI: 10.1145/3475716.3475782. [Online]. Available:
http://dx.doi.org/10.1145/3475716.3475782.

[136] jogetworkflow,
Generative ai and no-code — a match made in heaven - jogetworkflow - medium,
Medium, Oct. 2023.
[Online]. Available: https://jogetworkflow.medium.com/generative-ai-and-no-code-a-match-made-in-heaven-ca6e5684d474 (visited on 05/30/2024).

[137] Z. Ji, N. Lee, R. Frieske, *et al.*, "Survey of hallucination in natural language generation," 2023.
DOI: 10.1145/3571730. [Online]. Available: https://doi.org/10.1145/3571730.

[138] *Introducing chatgpt and whisper apis*, Openai.com, 2024. [Online]. Available: https://openai.com/index/introducing-chatgpt-and-whisper-apis/ (visited on 05/31/2024).

[139] ProductPlan, *Stakeholder Analysis*, https://www.productplan.com/glossary/stakeholder-analysis/, [Online; accessed 10-May-2024], 2022.

[140] fortunebusinessinsights, *Low Code Development Platform Market Size, Share Industry Analysis, By Component (Platform and Services), By Deployment (Cloud and On-premises), By By Enterprise Size (Large Enterprises and SMES), Application Type (Web Cloud Based, Mobile Based, and Desktop Based), By Industry (BFSI, Healthcare, Education, IT and Telecommunication, Media Entertainment, Manufacturing, Government, Retail, and Others), and Regional Forecast, 2024-2032*, https://www.fortunebusinessinsights.com/low-code-development-platform-market-102972, [Online; accessed 21-May-2024], May 13, 2024.

[141] C. Dilmegani, *Low-Code No-Code: Difference, Benefits Challenges in 2024*, https://research.aimultiple.com/low-code-no-code/, [Online; accessed 21-May-2024], Mar 18, 2024.

[142] I. Sacolick, *How generative AI will change low-code development*, https://www.infoworld.com/article/3713500/how-generative-ai-impacts-low-code-development.html, [Online; accessed 21-May-2024], MAR 11, 2024.