Journal of Software Engineering Research and Development, 2025, 14:4, doi: 10.5753/jserd.2025.5969

Democratizing AI Development: A Feature-Based Categorization of API Platforms, Development Frameworks, LCNC and AIaaS Platforms for LLM-Based Applications

Dimitrios Tolis [University of Turku | dimitrios.tolis@utu.fi]
Juuso Rytilahti [University of Turku | jubery@utu.fi]
Oshani Weerakoon [University of Turku | osweer@utu.fi]
Panu Puhtila [University of Turku | papuht@utu.fi]
Erkki Kaila [University of Turku | ertaka@utu.fi]
Tuomas Mäkilä [University of Turku | tusuma@utu.fi]

Abstract

In recent years, LLM-based AI development platforms have gained widespread adoption, enabling both IT professionals and citizen developers to create AI-powered applications. However, the landscape remains fragmented, with a variety of API-based platforms, AI development frameworks, Low Code/No Code (LCNC) platforms, and Domain-Specific AI as a Service (AIaaS) solution, each offering varying levels of accessibility and customization. Due to the recency of the interest in LLM-based AI development platforms, there is limited systematic research categorizing these tools based on their functionalities and intended user groups. This paper addresses this gap by proposing a structured, feature-based categorization framework, distinguishing between platforms based on criteria such as primary target group, degree of customization, and level of abstraction. Methodologically, we apply a feature-driven analysis grounded in documented capabilities and design affordances across a representative set of tools, and we operationalize the two core dimensions (customization and abstraction) through an anchored ordinal scoring rubric to produce a visual map of categories and overlaps. However, further empirical research is needed to validate the attitude of users towards the different tools in the categories. By providing a clearer understanding of AI development tools, this research supports more informed decision-making and contributes to the democratization of AI adoption across industries.

Keywords: *Categorization, LLM-based AI development, Citizen Developers, API-based platforms, AI frameworks, Low Code/No Code (LCNC) platforms, Domain-Specific AIaaS solutions (AIaaS)*

1 Introduction

The rapid adoption of Generative AI has led to an increasing demand for development platforms that enable both technical and non-technical users to build and integrate LLM-based AI solutions. The market now offers a variety of tools, including API-based platforms, AI development frameworks, Low Code/No Code (LCNC) platforms, and Artificial Intelligence as a Service application (AIaaS), aiming to different levels of expertise and development needs. However, as AI adoption is growing, selecting the most suitable development tool for specific requirements is challenging. At the same time, a major challenge that exists across Europe and globally is a widespread lack of AI-related skills (Moch & Oberdieck, 2024).

While AI presents opportunities to enhance efficiency and innovation across industries, it also highlights the urgent need for software developers to continuously update their skill sets to keep pace with rapid technological advancements. This skills shortage threatens the full realization of AI's benefits and poses risks to economic development and the viability of the global digital economy (Wisskirchen *et al*., 2017). Furthermore, as AI becomes more embedded in everyday technologies and applications, individuals without AI literacy and technical skills often express concerns and resistance to adoption (Long, 2021) (Ng *et al*., 2023). This growing skills-gap further highlights the importance of AI development tools that can be used both by IT professionals and citizen developers, i.e., users with little to no experience in coding, by offering more accessible and adaptable solutions.

However, the landscape of the LLM-based AI development remains fragmented, with platforms varying widely in their customization capabilities, accessibility, and industry applications. Despite the growing supply of AI development tools, there is limited systematic research categorizing them based on their features and intended user groups. To address this gap, this study proposes a structured categorization framework, systematically examining key technological and usability attributes that distinguish these platforms.

To guide this research, two key questions were formulated: "*RQ#1: What systematic approach can be used to categorize API-based platforms, AI development frameworks, Low Code/No Code (LCNC) platforms, and domain-specific AIaaS solutions?*" and "*RQ#2: What are the key features and functionalities that justify the proposed categorization?*".

To answer RQ#1, this study presents a categorization framework that distinguishes LLM-based AI development platforms into four distinct categories: (1) LLM-Based Platforms with API Access, (2) Developer-Centric AI Frameworks for Building Custom Applications, (3) LLM-Based No Code / Low Code Platforms for Citizen Developers, and

(4) Domain-Specific AIaaS LLM-based Platforms. This categorization was guided by a feature-driven perspective. RQ#2 was addressed by identifying key functionalities that justify the placement of tools within each category.

The paper contributes to the systematic categorization of AI development tools, offering researchers, developers, and organizations a clearer framework for evaluating platform options and capabilities. The findings highlight critical trade-offs between usability and flexibility, reinforcing the need for further empirical research, for example, task-based usability studies and field evaluations to validate the framework's applicability.

2 Literature Review

In a recent survey (McKinsey & Company, 2024) it was found that Al adoption worldwide has increased dramatically, reaching 72% of the 1363 respondents in 2024, increasing from 55% in 2023, after years of little meaningful change, although people without AI literacy and skills express concerns and resistance (Long, 2021). At the same time, the widespread lack of AI-related skills across Europe and the globe is a major challenge (Moch & Oberdieck, 2024). While AI presents opportunities to enhance efficiency and innovation in all industries, it also requires experienced software developers who must continuously update their skillset to keep pace with technological advancements. This skills shortage threatens not only the full realization of the benefits of AI, but also economic development and the viability of the global digital economy (Wisskirchen *et al*., 2017). Interestingly, to fill this gap, a growing trend has emerged: the 'citizen developers', non-IT professionals who can design and develop their own software solutions with minimal or no programming skills, using user-friendly LCNC tools and platforms (Binzer & Winkler, 2022; Esposito *et al*., 2023). However, at the organizational level, there is a lack of familiarity with the technology and its usage (Prinz *et al*. 2024), so that the citizen developer approach can be implemented.

Following the rapid and growing interest in AI applications development, there is a consequent growth in the development of tools that can facilitate this demand. Programming languages are updated with AI-specific libraries, new frameworks emerge, and new API platforms with many features are offered, together with the majority of the LLMs. Lately, new LLM-based LCNC tools are being introduced to simplify and democratize software creation by enabling users without extensive technical expertise to build applications through visual interfaces and predefined components, thereby empowering "citizen developers" (Viljoen *et al*., 2024). LCNC platforms create an environment where individuals and teams can build applications using visual interfaces, like drag-and-drop tools, instead of relying heavily on traditional coding (Esposito *et al*., 2023), with minimal or no programming code. According to a Gartner report (Gartner, 2024), the low-code application platform market in general is projected to reach $16.5 billion by 2027, with a compound annual growth rate of 16.3% from 2022 to 2027, with a growing demand for generative AI (GenAI) capabilities (Risk and Opportunity Index: Low-Code Application Platforms, 2024). These trends emphasize the pivotal role of LCNC in driving digital transformation across industries (Ajimati *et al*., 2025).

The rise of citizen developers, empowered by LCNC platforms and relevant tools, is transforming the traditional software development process. Ajimati *et al*. (2024) highlight that Citizen Development raises greater operational efficiency, reduces IT dependency, and empowers employees to address specific organizational challenges through tailored solutions. Benefits include improved alignment between business and IT, cost reductions, and the ability to drive digital transformation initiatives. However, challenges remain, such as cultural resistance, governance issues, and the risk of technical debt if applications are poorly designed or managed.

Although the recent interest in Generative AI and LLM-based applications is growing rapidly, there is still a lack of systematic reviews on the topic. Weber (2024) has proposed a taxonomy on LLM-integrated applications at a systems level, categorizing LLM components with the software systems. Strobel *et al*. 2024 classify Generative AI applications based on their functional roles, focusing on AI development platforms, introducing a classification based on user accessibility, customization, and integration capabilities to better serve both IT professionals and citizen developers, based upon 5 key characteristics: Generator, Reimaginator, Synthesizer, Assistant, and Enabler. Other ontological efforts, such as Artificial Intelligence Ontology (Joachimiak *et al*. 2024) provide structured classifications for AI methodologies, primarily focusing on conceptual AI knowledge rather than AI development tools.

Webb (2024) has mapped the gen-AI products by user experience, utilizing 3 primary dimensions: RAG/Large context, structured generation, and Real-time. Russo (2024) derived a theoretical model of AI adoption in software engineering: the Human-AI Collaboration and Adaptation Framework, which aims to ease AI tool design and provide a framework to create effective organizational implementation strategies. Passerini et al. (2025) state that potential and limitations between human-LLM interaction remain largely unexplored and should be emphasized in future research efforts. Distinguishing the different tooling based on stakeholder needs will therefore be beneficial for the research community.

In comparison to these earlier taxonomies, which have focused on specific AI technologies, our paper presents a larger framework which is not specifically focused on only one aspect of AI industry, but rather aims to illustrate the relationship between the use cases and stakeholder groups (Citizen developers, IT-professionals, different domain experts etc.), which make use of and integrate AI technologies to their business models. Thus, our categorization provides an easily accessible way to compare various AI technologies

and their usability in providing added value in different situations. As the interests and motivations of these various stakeholder groups vary quite widely, this kind of framework can be utilized as a guideline in tool selection, enhancing the development process significantly.

Feature-oriented classification has been used in software engineering to compare tools and derive structured taxonomies based on observable capabilities rather than perception-based adoption models. Prior work on taxonomy construction emphasizes the need for explicit classification criteria and decision procedures (Britto et al., 2016), while tool-oriented studies show that capability and feature-based coding can support transparent comparison when dimensions are operationalized and grounded in documented evidence (Ozkaya,2018, 2019; Cowie et al., 2022). This paper aligns with that tradition by defining a feature-driven coding rubric and reporting traceable evidence (Appendix B) alongside anchored scoring definitions (Appendix C), enabling readers to follow and reassess the classification rationale for each tool.

3 The Categorization Framework

To answer "RQ#1: What systematic approach can be used to categorize API-based platforms, AI development frameworks, Low Code/No Code (LCNC) platforms, and domain-specific AIaaS solutions?", the categorization initially was guided by three main criteria: the intended primary target group, the degree of customization, and the level of abstraction.

The primary target group, IT-Professionals or citizen developers, has been applied as a first criterion to identify which tools best suit different levels of expertise and application needs. IT professionals, such as software engineers and data scientists, in general, are expected to require highly customizable platforms with API access and advanced programming capabilities to optimize performance and scalability. Similarly, developer-centric AI frameworks, including LangChain and Rasa (da Costa *et al*., 2024), offer substantial flexibility through their modular components, enabling structured AI development with greater efficiency, but still require significant programming proficiency.

Citizen developers, in contrast, need tools that prioritize ease of use, automation, and minimal coding. This group, including professionals, business users, and educators, is expected to make use of visual tools. Platforms such as Flowise and Langflow offer visual drag-and-drop interfaces, pre-defined templates, and automated workflows with limited or no customization functionality, reducing the complexity of AI application development and making it accessible to users with limited or no coding skills. Furthermore, Domain-Specific AIaaS LLM-based Platforms, such as Salesforce Einstein AI Solutions, are designed for end users in specific sectors, such as healthcare, finance, marketing, and education, providing AI-powered functionalities without requiring any coding (No Code) or model adaptation.

Finally, the level of abstraction further refines this categorization by assessing how much technical complexity is hidden from the user. API-based platforms, which expose raw AI functionalities through direct API calls, represent the lowest level of abstraction, as they require developers to implement all aspects of AI integration manually. Developer-centric frameworks provide a slightly higher level of abstraction by offering pre-built components that streamline AI development while still necessitating programming skills. LCNC platforms operate at an even higher level of abstraction, as they replace traditional coding interfaces with graphical environments that allow users to create AI applications with minimal or no programming knowledge. At the highest level of abstraction, domain-specific AIaaS solutions eliminate technical barriers entirely by embedding AI functionalities within industry-specific applications, ensuring that users can leverage AI without needing to understand the underlying mechanisms.

Based on the analysis presented in this paper, we propose the following categorization framework:

Category 1: LLM-Based Platforms with API Access, in which there are platforms that provide direct access to company-owned or open-source LLMs with extensive customization options through APIs and integrations, enabling users to create advanced, highly tailored applications. Suitable for developers requiring full control over AI workflows and integrations.

Category 2: Developer-Centric AI Frameworks, platforms are designed to provide developers with the flexibility and control needed to build complex, customized AI workflows, pipelines, and multi-step applications. These tools are aimed at users with coding expertise who require SDKs, API libraries, and modular components.

Category 3: LLM-Based No Code / Low Code Platforms, through which technical or non-technical users (citizen developers) may quickly create LLM-powered applications using visual interfaces, templates, and basic customization options with minimal or no coding. These platforms balance ease of use with limited flexibility, suitable for quick and easy deployment.

Category 4: Domain-Specific AIaaS LLM-based Platforms, which are industry-focused solutions that offer pre-configured workflows and tools optimized for specific business use cases. These platforms provide minimal customization, aiming to deliver out-of-the-box solutions for niche applications.

Hybrid Category: A Multi-Affiliation Perspective. A key limitation of any discrete classification system is its ability to handle platforms that span multiple categories. In the rapidly converging AI development market, this hybridity is increasingly common. Major providers intentionally design ecosystems that serve multiple user groups with both low-

level and high-level capabilities. A platform like Google's Vertex AI, for instance, offers both raw API access for developers (Category 1) and a no-code agent builder for business users (Category 3).

To address this challenge and enhance the framework's utility, we adopt a multi-affiliation perspective, which allows a single platform or ecosystem to be associated with more than one category. Rather than forcing a platform into a single "dominant" category - a subjective exercise that risks oversimplification - this perspective acknowledges versatility and provides a richer view for practitioners deciding which platforms best fit their needs.

Vertex AI is a prime example of a hybrid platform. Its core is a powerful MLOps suite with a "Model Garden" providing API access to over 200 models and tools for custom training, aligning it with Category 1. Simultaneously, its "Agent Builder" and "AutoML" features provide a no-code environment for citizen developers, giving it a strong affiliation with Category 3. An organization can adopt Vertex AI as a unified platform where a technical team operates in Category 1 space, while a business team leverages its Category 3 capabilities.

Another example is the Rasa Ecosystem (Dual Affiliation by Product: Category 2 & Category 3). The Rasa ecosystem achieves hybridization through product differentiation. Rasa Open Source is a quintessential Category 2 developer-centric framework, requiring coding expertise for building custom assistants. In parallel, Rasa Studio is explicitly marketed as a no-code, drag-and-drop AI assistant builder for business users, placing it firmly in Category 3. The ecosystem demonstrates a multi-affiliation strategy, targeting two distinct user groups with tailored products.

3.1 Methodological Grounding and Decision Rules for the Feature-Based Approach

To provide methodological clarity and reproducibility for the feature-based approach, we explicitly operationalize the classification as a structured coding and decision process grounded in feature-oriented tool evaluation in software engineering. Similar capability- and feature-based tool inspections have been used to compare software tools. For example, Ozkaya (2019) analyzes 58 UML modelling tools against practitioner-oriented capability groups, and Ozkaya (2018) evaluates architectural languages with an emphasis on language features and tool support. Comparable feature surveys appear in other domains, such as the systematic search and feature analysis of web-based SLR tools in medicine (Cowie et al., 2022) and the review-style overview of power system analysis software tools in a rapidly evolving industry (Bam & Jewell, 2005). Our motivation (Section 1) and positioning within existing literature (Section 2) align with these strands of work, as they share a common aim: to support understanding and selection through explicit feature-based comparison.

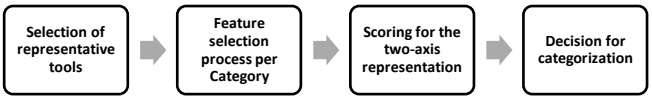

Figure 1. The 4-step process illustrates the methodology. Selection of representative tools is described in 3.2, feature selection in 3.3., scoring for the two-axis representation in 3.4 and the final category selection process for each tool is defined in 3.5.

The process followed four steps (Figure 1). To strengthen traceability and manage platform volatility, we fixed an evidence window (January–June 2025) and recorded per-tool evidence notes, including the date checked, source URL, a brief category justification (Appendix B.1–B.4), and a scoring for the two-axis representation.

3.2 Selection of Representative Tools for each Category

The selection of AI development tools for this study, although non-exhaustive, followed specific inclusion criteria to ensure a representative sample of tools relevant to LLM-based applications. The sampling window was from January 2025 to June 2025, thus representing the current state-of-the-art as well as possible, considering that the phenomena observed in this paper are in constant development and change. We prioritized platforms that showed:

1. Industry adoption signals, such as public API usage in production contexts, notable customer references, sustained release cadence, and community engagement, such as issues/PRs and GitHub stars.
2. Presence in industry reports and technology reviews, indicating relevance beyond niche use, and
3. Sufficient documentation depth, such as developer guides, API/SDK references, and changelogs, to enable consistent and replicable feature coding.

Where tools exhibited overlapping capabilities, we retained them and applied the multi-affiliation rule described in Section 3.5, assigning a primary category based on dominant (C/A) placement and marking a secondary affiliation when the tool lies near an adjacent boundary, supported by evidence notes in Appendix B.

Exclusion criteria also were applied. General automation tools without explicit LLM integration were excluded as non-relevant. Also, Machine Learning as a Service (MLaaS) platform, such as AWS SageMaker were excluded as MLaaS platforms focus on model training and deployment not the LLM applications which is the focus of this study. Also, generic programming languages like Python or JavaScript were excluded as being too generic and multipurpose.

3.3 Features Selection Process per Category

To address *"RQ#2: What are the key features and functionalities that justify the proposed categorization?"* we followed an iterative, feature-first approach. We built an initial "list of features" by an empirical scan of recurring features described in representative tools documentation and mapped these capabilities to the needs of different user groups. For example, IT Professionals require customization, SDK/CLI depth, and infrastructure control, while LCNC tools for Citizen Developers emphasize visual builders, templates, and onboarding. We then refined the list by reviewing representative tools and consolidating recurring functional and non-functional features (Table 1).

When selecting the list of features to be included in the categorization, we observed similar kinds of classifications for software development. Taheri & Sadjadi (2015) describe a classification for agile software development tools based on several categories, but with features mentioned as an essential aspect. Similarly, our classification is highly focused on the features offered by the tools. In the survey conducted by Azizian et al. (2011), the developers listed features and ease-of-use as the most satisfactory factors, and factors such as price and usability as the least satisfactory.

To validate and refine these features, a comprehensive review of a sample of well-known tools within each category was conducted, such as OpenAI API (Category 1), LangChain (Category 2), Flowise (Category 3), and Salesforce Einstein AI (Category 4) and provided a baseline of commonly implemented features. This analysis revealed patterns in common functionalities, such as API extensibility in LLM-based platforms, modular SDKs in developer-centric frameworks, and drag-and-drop interfaces in low-code/no-code platforms.

Additionally, an iterative approach was applied, where the initial list of features was refined, as specific tools were analyzed in greater detail and new characteristics were identified, such as the importance of community support and interactive tutorials for citizen developers or scalability for IT professionals, were incorporated throughout the process. This approach ensured that the selected features were both practically relevant and theoretically sound, addressing the diverse needs of the user groups while reflecting the state-of-the-art capabilities in LCNC tools for LLM-based applications.

To provide a verifiable trail with minimal overhead, we include for each tool an Appendix entry with the date checked, source URL, and a brief category justification (see Appendix B.1-4).

Table 1. Features Selection Criteria: Selection criteria for key features used to analyze AI development platforms across categories, based on the needs of primary user groups, common tool characteristics, and iterative refinement aligned with a feature taxonomy.

Category	Primary Target Group Needs	Common Characteristics of Existing Tools	Iterative Refinement
LLM-Based Platforms with API Access (Category 1)	IT professionals: Fine-grained control (APIs/SDKs), broad integrations, perf tuning, flexible deployment, observability, enterprise security/compliance.	Robust API documentation, authentication mechanisms, performance optimization, and advanced configurations.	Added focus on features supporting enterprise-grade scalability and debugging tools based on tool-specific analysis (e.g., OpenAI, Anthropic).
Developer-Centric AI Frameworks (Category 2)	IT professionals: RAG/agents, modular pipelines, eval/tracing, CI/CD-ready, local/hosted model support, clear SDK/docs	Modular SDKs, plugin architectures, pre-built components, and debugging environments.	Iterated with additional emphasis on multi-agent workflows and compatibility with popular development environments (e.g., Rasa).
LLM-Based No Code / Low Code Platforms (Category 3)	Citizen developers: Visual builders, templates/wizards, turnkey connectors, hosted runtime, RBAC/guardrails, strong onboarding/community.	Drag-and-drop builders, visual workflow design, pre-configured templates, and workflow automation.	Updated list to include community support resources and interactive tutorials after reviewing tools like Botpress AI and Zapier.
Domain-Specific AIaaS LLM-based Platforms (Category 4)	Citizen developers and business users: Prebuilt industry workflows, domain templates, policy presets, CRM/ ERP/ helpdesk integrations.	Pre-trained models, domain-specific templates, minimal setup requirements, and integration options.	Expanded to include features like AI-powered analytics and industry-specific documentation (e.g., Salesforce Einstein GPT, Jasper).

Table 2. Categorization Matrix for API-based platforms, AI development frameworks, Low Code/No Code (LCNC) platforms, and domain-specific AIaaS solutions, with means and min-max ranges for each category,

Category	Primary Target Group	Number of tools	Customization Mean	Customization Range (0-3)	Abstraction Mean	Abstraction Range (0-3)
Category 1: LLM-Based Platforms with API access	IT-Professionals	12	2.92	2–3	0.50	0–1
Category 2: Developer-Centric AI Frameworks	IT-Professionals	9	3.00	3–3	1.00	1–1
Category 3: LLM-Based No Code / Low Code Platforms	Citizen Developers	13	1.54	0–2	2.15	2–3
Category 4: Domain-Specific AIaaS LLM-based Platforms	Citizen Developers	8	0.75	0–2	3.00	3–3

3.4 Scoring for the two-axis representation

To support the two-axis representation (Figure 2), we applied an anchored scoring rubric to each tool (Table 2). Specifically, each tool was assigned ordinal scores for Customization (C) and Abstraction (A) on a 0–3 scale (0 = low, 3 = high), based on observable characteristics documented in primary sources (see Appendix C for the scoring matrices and Appendix B for traceability). The purpose of this scoring is not to produce precise measurements, but to provide a transparent and repeatable way to position tools along two high-level dimensions that are central to our categorization.

Customization (C) captures the degree of control a user retains over the solution, including composition, extensibility, and deployment. Indicators include the depth of API/SDK access, the ability to implement programmable orchestration and custom logic, and the availability of self-hosting or controlled deployment options. Tools closer to Category 1 and Category 2 typically score higher on this axis because they enable deeper implementation control.

Abstraction (A) captures the extent to which complexity is encapsulated through higher-level constructs. Indicators include the presence of visual builders, reusable templates, guided workflows, and turnkey domain functionality that reduces the need for low-level implementation decisions. Tools closer to Category 3 and especially Category 4 typically score higher on this axis because they prioritize usability and rapid configuration over deep customization.

Example Scoring Rubric:

C (Customization): 0 = parameter/template only | 1 = sandboxed scripting/logic gates | 2 = extensible runtime (custom code + dependencies) | 3 = full code/infrastructure control.
A (Abstraction): 0 = raw API primitives | 1 = managed platform layer (console, evaluation tools) | 2 = visual builder/orchestrator | 3 = turnkey domain product.

3.5 Deciding on the Final Categorization

After scoring each tool on Customization (C) and Abstraction (A), as a first step we assigned a primary category based on the tool's dominant feature profile, as defined by the category descriptions in Section 3. In practical terms, the C/A scores provide an explicit, structured way to interpret this dominant profile: tools with high customization and low abstraction naturally align with Category 1. Developer frameworks with high customization and moderate abstraction align with Category 2. LCNC platforms with higher abstraction and limited-to-moderate customization align with Category 3. And domain-specific AIaaS platforms with very high abstraction and low customization align with Category 4. To make category boundaries and overlaps transparent, we then positioned all tools on a feature-based map using their (C, A) scores. The map visualizes both the within-category variation (min–max ranges) and the proximity of some tools to adjacent categories, which motivates the multi-affiliation concept discussed in Section 3. The resulting visualization is presented in Figure 2.

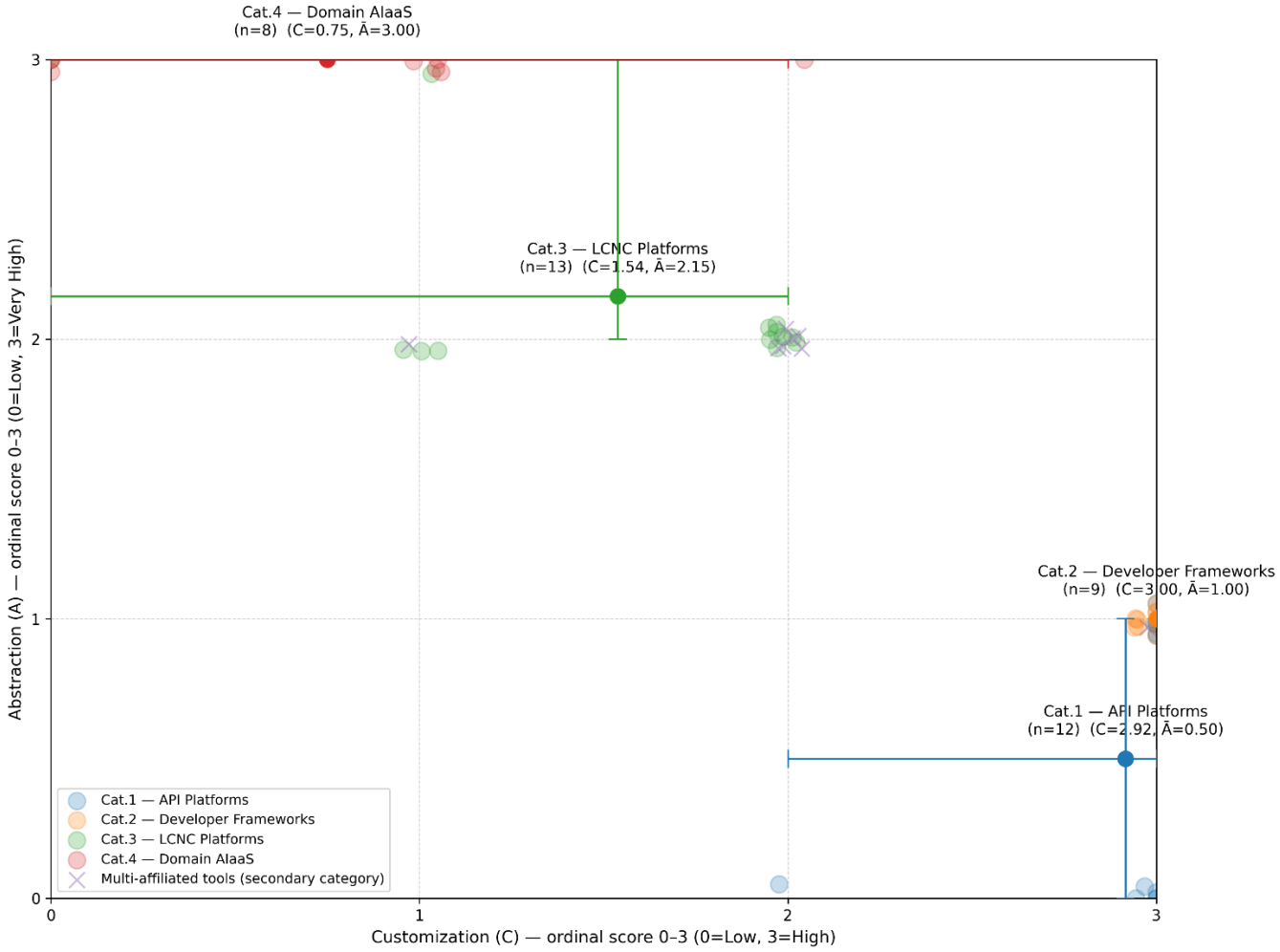

Figure 2. Feature-Based Map of Categories with Tool-Level Scores, Means and Ranges. Each tool listed in Appendix B was assigned ordinal scores for Customization (C) and Abstraction (A) using anchored definitions (0–3). Points represent tool-level scores; category markers indicate means (mean C and mean A across tools in the category), and error bars indicate within-category min–max ranges. Tools with a secondary affiliation are highlighted with the X symbol to operationalize overlap under the multi-affiliation rule.

4 Analysis of Categorization Results per Category

This chapter examines the key features that differentiate AI development platform categories, focusing on their technical capabilities, usability, and intended users. Additionally, the analysis evaluates how well the categorization aligns with real-world use, considering adoption trends, user challenges, and industry needs.

4.1 Technical and Functional Features of Category 1: LLM-Based Platforms with API Access

Table 3 provides an overview and description of the key technical and functional features of platforms in Category 1: LLM-Based Platforms with API Access.

Category 1 consists of API-based platforms which demand programming expertise, as their primary purpose is to enable developers to build tailored AI solutions through APIs and cloud-based hosting (for a comparative review see the categorization matrix in Appendix A.1). These platforms maximize capability (fine-grained API control, performance tuning, deployment flexibility) while demanding higher implementation effort relative to visual or pre-configured platforms. Their advanced capabilities, including extensive API control, security compliance, and cloud deployment, make them powerful tools for businesses and researchers.

Their capabilities are also tested in various scenarios, see for example Li *et al*. (2023), benchmarking the performance of the different models' capabilities on operating on a large set of different APIs to measure their effectiveness. Furthermore, these platforms and their API's can be seen as a backbone of the platforms and tools described in other categories defined in this study. Most utilize LLMs directly or, in the case of more developer-centric frameworks, have built-in plug-and-play support for the APIs of the most popular LLM-based platforms. For a practical example, see Morales-Chan *et al*. (2024). However, the technical expertise required to configure and optimize these models places them beyond the reach of non-technical users, ensuring that they remain tools designed for IT professionals and AI engineers, see for example Parthasarathy *et al*. (2024).

One of the defining characteristics of these platforms is their high level of configurability, allowing users to fine-tune models, optimize performance, and integrate AI into complex enterprise systems. Most platforms offer fine-tuning tools, though some, like Anthropic, rely on external solutions such as Amazon Bedrock. Alongside fine-tuning, these platforms also provide various model sizes, enabling developers to balance efficiency and computational power based on specific application needs.

While they share core functionalities, these platforms exhibit significant differences in their features. Modality support varies greatly, with some platforms supporting text, image, audio, and embeddings, while others are limited to text-based processing. Also, for prompt engineering support, platforms like OpenAI and Vertex provide tools for helping developers optimize their prompts, while others like Mistral provide only documentation and best practices. Additionally, some platforms, such as OpenAI and Anthropic, operate as closed-source solutions, offering their in-house built models only, whereas platforms like Vertex AI and Amazon

Table 3. Features of Category 1: LLM-Based Platforms with API Access

Feature	Description
LLM Origin (Developed by Company, Third-party, or Both)	Indicates whether the platform offers LLMs developed by the company, LLMs from other providers/companies, or a combination of both.
Open Source / Closed Source / Both	It indicates whether the LLM platforms offered are open source, closed source, or a mix. This affects flexibility and integration options.
Input/Output Modality (Text, Image, Video, Audio, Embeddings)	The platform's capability to process and generate different data formats (e.g., text, images, video, audio). This parameter determines the scope of potential applications and use cases in production environments.
Model Fine-Tuning Tools	Availability of tools or capabilities to fine-tune the LLM for specific applications, allowing customization with custom organizational data or behaviors.
Prompt Engineering Support	Indicates whether the platform provides tools or features to assist users in creating, refining, and managing prompts to optimize AI model performance.
Multiple Model Size Options	Availability of model variants with different parameter counts and computational requirements, enabling optimization for various deployment scenarios and performance requirements.
Data Privacy and Security Controls	Implementation of technical security measures like encryption, access controls, and data retention policies to protect sensitive information during processing and storage.
Compliance with Industry Standards	Adherence to industry standards or regulations (e.g., GDPR, HIPAA), relevant for use in regulated industries.
Consultancy and Support Services	Availability of professional services to assist with customization, integration, or troubleshooting for enterprise deployments.

Bedrock offer both proprietary and open-source options, providing developers with greater flexibility in selecting and deploying AI models. Hugging Face differs entirely by positioning itself as an open-source AI hub centered around research, community collaboration, and model hosting rather than proprietary AI services (Ai *et al*. 2023, Castaño *et al*. 2024, Jain 2022, Jones *et al*. 2024).

4.2 Technical and Functional Features of Category 2: Developer-Centric AI Frameworks

Table 4 provides an overview and description of the key technical and functional features of platforms in 2: Developer-Centric AI Frameworks.

Category 2 consists of developer-centric AI frameworks, designed to provide structured support to developers for building and optimizing AI-driven applications (for a comparative review, see the categorization matrix in Appendix A.2). These frameworks abstract many complex operations, making it easier for developers to integrate large language models (LLMs) into their workflows without handling low-level API interactions manually. By providing structured execution, retrieval-augmented generation (RAG), agent-based architecture, context and memory management, and observability tools, they enable developers to build scalable, intelligent applications with greater efficiency (Topsakal & Akinci 2023). Examples of such tools from literature, enabled by these technologies, would be MindGuide, developed by Singh *et al*. (Singh *et al*. 2024), or Intelligent Spark Agents, a framework for creating AI-powered Spark agents by Wang and Duan. (Wang & Duan 2024).

Category 2 frameworks deliver high functional leverage via structured workflows, RAG primitives, agent orchestration, context/memory management, and observability. They improve developer effort through SDKs, templates, and integrate external data sources (Holkar *et al*. 2024), (Morales-Chan *et al*. 2024).

These frameworks are all LLM-agnostic, meaning they are not tied to a single provider, but instead allow developers to integrate a variety of models depending on their needs. A defining characteristic across this category is RAG support, which enhances AI applications by retrieving external knowledge at runtime, improving response accuracy, and reducing model hallucinations. While all frameworks in this category offer tools helpful for RAG applications, some of them, like Haystack and LlamaIndex, are more focused on it. Alongside RAG, all these frameworks incorporate context and memory management, allowing AI applications to maintain conversation history or task state across multiple interactions, essential for personal assistants, automated workflows, and long-form content generation.

Despite shared functionalities, these frameworks differ in focus and implementation. While all offer observability tools, LangChain stands out with LangSmith, a robust debugging and tracing solution for optimizing AI pipelines. Another key distinction is that while all frameworks are open-source at their core, some also provide enterprise cloud platforms, such as the LangGraph Platform and CrewAI, allowing developers to leverage managed infrastructure for production-scale deployment. Additionally, these are purpose-built for agentic workflows and multi-agent collaboration, as seen in CrewAI and LangGraph, whereas others such as LangChain and LlamaIndex, follow a structured, step by step execution model for more traditional AI applications.

Table 4. Features of Category 2: Developer-Centric AI Frameworks

Feature	Description
Open-Source Status	Indicates whether the framework is open source, closed source, or offers a mix, affecting adoption and customization options.
LLM Agnostic or LLM Specific	Indicates whether the framework is built to support multiple LLMs or is optimized for specific LLMs.
Local Model Support	Capability to run local LLMs and handle their specific requirements.
RAG Capabilities	Implementation of the Retrieval Augmented Generation architecture, encompassing document preprocessing, vector embeddings generation, and contextual knowledge retrieval mechanisms for enhanced model responses.
Agent Architecture	Framework support for implementing autonomous AI agents with reasoning capabilities, sequential task planning, and programmatic tool utilization for complex workflow automation.
Context Management	The system's ability to maintain conversation history and contextual information across multiple interactions in a session.
Prompt Engineering Tools	Tools for creating, managing, and optimizing LLM prompts.
Response Streaming	Capability to display model outputs in real-time, showing the response as it is generated (character-by-character or token-by-token).
Runtime Features	System reliability features for consistent performance, including automatic data storage, error handling, and request management.
Observability	Tools for monitoring, logging, and analyzing LLM operations.
Deployment Options	Offering variable options for deployment.
Enterprise Readiness	Enterprise-grade features, security controls, and professional support options.

4.3 Technical and Functional Features of Category 3: LLM-Based No Code / Low Code Platforms

Table 5 provides an overview and description of the key technical and functional features of platforms in Category 3: LLM-Based No Code / Low Code Platforms.

Category 3 consists of LLM-Based No Code / Low Code (LCNC) platforms that enable non-technical users to build AI applications through intuitive interfaces and automation tools with minimal or no coding (for a comparative review see the categorization matrix in Appendix A.3). These platforms simplify AI development by reducing the required knowledge for application development, making them ideal for citizen developers, offering improved efficiency in software development (Käss *et al*., 2023).

LCNC platforms emphasize rapid development via visual builders and guided workflows, letting non-IT professionals create AI apps with minimal code. Users can interact with LLMs through drag-and-drop interfaces, form-based workflows, or pre-configured prompts, significantly reducing the learning curve. For example, Flowise offers a very easy to learn and use environment, making it ideal for non-technical users, like educators, to use (Tolis *et al*. 2025). However, they often lack the flexibility and deep customization capabilities available in developer-centric frameworks aside from the said task (Shlomov *et al*., 2024). While some LCNC tools offer API endpoints and limited scripting options, most prioritize usability over technical depth. For example, IDA (Intelligent Digital Apprentice) is an LLM-powered no-code UI automation tool that guides non-technical business users to create web automation without programming skills (Shlomov *et al*., 2024).

Table 5. Features of Category 3: LLM-Based No Code / Low Code Platforms

Feature	Description
Open-Source Status	Indicates whether the platform is open source, closed source, or offers a mix, affecting adoption and customization options.
Development Interface	Primary development paradigm implemented in the platform's interface (e.g., visual node-based workflows, structured form interfaces, hierarchical list systems) for application construction without traditional programming.
LLM Agnostic or LLM Specific	Platform's architectural approach to model integration, indicating whether it supports multiple Large Language Models through standardized interfaces or is optimized for specific model implementations.
Local Model Support	Capability to run local LLMs and handle their specific requirements.
RAG Capabilities	Support for Retrieval Augmented Generation including document preprocessing, embeddings, and knowledge retrieval.
Workflow Orchestration	Support for creating and managing structured sequences of LLM operations and tasks.
Agent Architecture	Support for building AI agents with reasoning, planning, and tool-use capabilities.
Prompt Engineering Tools	Tools for creating, managing, and optimizing LLM prompts.
Pre-built Templates	Availability of ready-to-use templates and solution accelerators.
Response Streaming	Capability to process continuous model outputs in real-time, enabling progressive response generation and display.
Runtime Features	System reliability features for consistent performance, including automatic data storage, error handling, and request management.
Observability	Tools for monitoring, logging, and analyzing LLM operations.
Hosting Options	Available infrastructure options for system deployment and hosting.
Platform Extensibility	Support for extending core platform functionality through development tools, custom components, or third-party integrations.
Security Features	Implementation of security measures including encryption, API key management, and compliance options
Access Control & Collaboration	Features for user management, role-based permissions, and team collaboration

All platforms in this category share core features. They provide a visual no-code or low code interface, most of them drag-and-drop node-based, like Flowise, while others follow a conversation approach, using AI to build the solution, like Zapier. Prebuilt templates enable fast deployment by offering ready-to-use configurations for common AI applications, reducing setup time and technical complexity. With small exceptions, all of the platforms allow the creation of complex AI workflows, as well as agentic applications. The platforms in this category support RAG, to enhance AI responses, with Flowise and Botpress offering robust RAG capabilities with specialized features.

Despite these similarities, significant differences exist. Hosting options range from cloud-only solutions like Microsoft Copilot Studio, hybrid solutions like Flowise, where both self-hosting and cloud-hosting is possible, and Langflow where only self-hosting is an option. The open-source versus closed-source divide follows the same pattern, with examples like Flowise and Langflow are open-source frameworks, while Microsoft Copilot Studio and ChatGPT's Custom GPTs remain proprietary.

4.4 Technical and Functional Features of Category 4: Domain-Specific LLM-Based AIaaS Platforms

Table 6 provides an overview and description of the key technical and functional features of platforms in Category 4: Domain-Specific LLM-Based AIaaS Platforms.

Category 4 consists of Domain-Specific AIaaS LLM-Based Platforms, which provide AI functionalities tailored to specific industries such as marketing, sales, customer service, finance, healthcare, and education (for a comparative review see the categorization matrix in Appendix A.4). Unlike general-purpose AI development frameworks and platforms, these platforms integrate AI capabilities directly into predefined industry workflows and templates, automating the most useful and frequently required tasks. They apply a task-focused design which offers the environment and the tools business users to act as citizen developers and develop AI applications or tools without the need for deep technical expertise.

Domain-specific AIaaS platforms package pre-configured workflows and industry templates (e.g., CRM, support, marketing). Typically, they offer fast time-to-value with guided configuration, automating processes like AI-powered customer interactions for example Zendesk AI for customer support (Ehsani *et al*, 2023), marketing content generation such as Copy.ai (Hassan *et al*, 2024), or AI-driven sales assistance for routine tasks, such as Salesforce Einstein AI (Kaliuta, 2023).

Their trade-off is reduced customization versus frameworks and APIs, though some products expose deeper workflow editing and integrations. For example, Jasper AI and Copy.ai provide customizable marketing and content-generation templates, whereas Zendesk AI focuses on AI-powered customer service chatbots and ticketing automation. Despite this similarity, these platforms differ significantly in their customization capabilities. For example, Kore.ai and Ada AI, allow extensive workflow modifications, enabling businesses to tailor AI interactions to their unique requirements. Others, such as Synthesia (Genç, A *et al*, 2023), focus primarily on pre-configured AI models for video creation, offering limited control over underlying AI processes.

A key distinction within this category is the range of AI integration and business data exchange options. Platforms like Salesforce Einstein and HubSpot AI provide deep

Table 6. Features of Category 4: Domain-Specific LLM-Based AIaaS Platforms

Feature	Description
Industry AI Functions	AI capabilities pre-configured for specific industry tasks (e.g., automated content creation for marketing, customer service response generation, sales lead scoring).
Template Library	Pre-built, industry-specific templates and prompts (e.g., email templates, chat flows, marketing copy formats) ready for immediate use.
Workflow Configuration	Extent to which users can modify AI workflows (e.g., customizing response rules, adjusting decision paths, setting trigger conditions).
Output Controls	Available parameters for controlling AI outputs (e.g., tone adjustment, response length, brand settings, compliance filters).
Business Solutions Integration	Native integration capabilities with enterprise systems and business software, supporting operational workflows.
Data Exchange	Ability to import/export business data through standard formats (e.g., CSV, API endpoints, JSON, direct database connections).
Team Management	Collaborative features for team operations (e.g., shared content libraries, approval workflows, team performance tracking, role assignments).
Access Controls	Security features for user management (e.g., role-based permissions, audit logs, authentication methods, data access restrictions).
Performance Metrics	Business analytics tools tracking platform usage and effectiveness (e.g., response rates, user engagement, task completion rates, ROI metrics).
Compliance and Quality Assurance	Evaluates the platform's ability to adhere to industry-specific regulations and standards while ensuring the accuracy, consistency, and reliability of outputs e.g. GDPR, HIPAA and tools for optimizing the quality of generated content or services.

enterprise integration, allowing AI-powered insights to be embedded within CRM, sales, and marketing operations (Rios-Campos, C. *et al*, 2023). These platforms support advanced data exchange, enabling businesses to import/export structured data for AI-driven analytics. In contrast, Copy.ai and Jasper prioritize ease of use for content generation but offer more limited integration options beyond basic API access.

Despite the strengths of this category, challenges remain, particularly in balancing usability with customization. Some AIaaS platforms restrict user modifications to maintain ease of use, limiting adaptability for businesses with unique requirements. Overall, Domain-Specific AIaaS platforms target business users seeking AI-driven automation with minimal technical complexity. Their industry specialization accelerates time-to-value, but varying degrees of customization highlight trade-offs between usability and flexibility. As these platforms continue evolving, their ability to balance pre-configured AI capabilities with user-driven customization will be critical in defining their adoption and effectiveness across different business sectors.

Table 7. Summary of Non-Functional Factors Across AI Platform Categories

Category	Cost profile	Typical governance	Key risks to verify
LLM-Based Platforms with API Access (Category 1)	Low platform & usage; high engineering labor	Provider certificates vary; shared responsibility	Throughput/latency limits, data terms, lock-in
Developer-Centric AI Frameworks (Category 2)	OSS core & optional enterprise; medium labor	Your responsibility; enterprise add-ons available	Ops complexity, security hardening, split compliance
LLM-Based No Code / Low Code Platforms (Category 3)	Subscription & usage; low labor for scoped apps	Mixed: features & evolving certificates	Scale ceilings, exportability, role/audit depth
Domain-Specific AIaaS LLM-based Platforms (Category 4)	Subscription (bundled usage); minimal labor	Strong, sector-aligned	Vendor lock-in, limited extensibility, data portability

4.5 Non-Functional Features Considerations

While our categorization focuses on functional capabilities and target users, real-world selection is often decided by non-functional requirements. We highlight three clusters with the greatest impact across organizations: cost structure, scalability & infrastructure management, and governance & compliance. Throughout, vendor examples are illustrative, not exhaustive (Table 7).

Cost Structure and Economic Implications: When the same underlying models are used, raw API/usage costs are broadly comparable across Categories 1–3. What really diverges is time-to-value and the labor mix. Category 1 carries the lowest platform fees but the highest engineering cost, because teams must design and maintain authentication, rate-limiting, streaming, evaluation/observability and security hardening.

Category 2 trims this build time with structured components (RAG, agents, evals, orchestration) but still relies on professional software engineers; open-source cores reduce licensing, while optional enterprise add-ons (managed hosting, monitoring, support SLAs) re-introduce platform spend.

Category 3 flips the equation by shifting much of the assembly to citizen developers using visual builders and templates; subscriptions add to usage costs, but delivery compresses to hours or days for well-scoped use cases, reducing dependence on premium engineering capacity. Category 4 largely bundles engineering into the subscription: organizations pay for immediate business outcomes with minimal build effort, trading customizability for speed and shifting cost evaluation from engineering budgets to ROI-oriented business decisions.

Scalability and Infrastructure Management: The central consideration is the allocation of operational responsibility, i.e. who operates which components, and the extent of performance and deployment control retained by the adopting organization. Category 1 offers fully managed autoscaling that minimizes ops burden but limits deployment latitude - some vendors expose VPC/private link and regional options, yet fine-grained infra tuning remains constrained. Category 2 brings maximum control with maximum responsibility: self-hosting requires architecture, DevOps/SRE maturity (autoscaling, caching/queues, tracing, CI/CD, rollback), and sustained operational investment; "managed around the framework" services can offset parts of this at extra subscription cost.

Category 3 generally abstracts infrastructure; many tools offer both SaaS and self-hosting, while managed-only products maximize simplicity but can cap throughput or customization - excellent for rapid rollouts, provided you confirm scale ceilings and export/backup paths. Category 4 is vendor-operated and domain-optimized, yielding the lightest ops footprint; the trade-off is higher lock-in and less

flexibility for atypical latency/throughput targets or non-standard data-flow requirements. Across all categories, confirm observability depth (tracing, metrics, evaluation harnesses), environment options (VPC, on-prem, hybrid), and data-flow controls (PII handling, retention boundaries).

Governance and Compliance: Security, privacy, data residency/sovereignty, access control, auditability, and sector obligations drive real-world feasibility. Category 1 ranges from certification-forward providers (e.g., SOC 2, HIPAA BAA where applicable) to private-deployment-forward models that let customers meet compliance in their own environments; either way, it's a shared-responsibility regime for keys, DPIAs, and retention. Category 2 places compliance on the adopter: open-source cores provide flexibility but no inherent attestations; commercial/managed tiers layer on enterprise controls (SSO, RBAC, audit logs), yet you still compose and evidence end-to-end compliance across multiple components and environments.

Category 3 shows maturity variance: enterprise-oriented tools often document certifications and admin controls, while younger products emphasize governance features (SSO/RBAC/audit) ahead of formal audits - so verify artifact exportability, data-handling guarantees, and admin policy depth.

Category 4 typically maintains strong, sector-aligned governance and clear data-processing terms, which reduces organizational burden at the cost of deployment choice, still assesses data portability, human-in-the-loop controls, and vendor lock-in implications.

In addition, organizations should assess ethical risks, including bias/fairness testing, safety guardrails, red-teaming practices, and content-moderation policies, and require evidence of model-risk management processes alongside compliance attestations.

5 Discussion

To answer *"RQ#1: What systematic approach can be used to categorize API-based platforms, AI development frameworks, Low Code/No Code (LCNC) platforms, and domain-specific AIaaS solutions?"* a categorization framework for LLM-based development tools is proposed, in four distinct categories, which are derived from the observed differences in target user groups' needs, functionality, degree of customization and level of abstraction.

In summary, LLM-Based Platforms with API access (Category 1) offer raw model access and provide maximum flexibility to experienced developers, Developer-Centric AI Frameworks (Category 2) enable IT professionals with coding experience to design and deploy complex custom AI workflows and integrations, LLM-Based No Code / Low Code Platforms (Category 3) offer easy to use interfaces for rapid application development by citizen developers and the Domain-Specific AIaaS LLM-based Platforms (Category 4) offer ready to use AI-enhanced templates in specific sectors like finance, marketing, education or healthcare.

To address *"RQ#2: What are the key features and functionalities that justify the proposed categorization?"* the analysis examined key functional and non-functional features across numerous representative tools (e.g., API depth, SDKs, RAG support, agent orchestration, observability, deployment options, governance/compliance, security, and cost model). Appendix B documents per-tool source-and-date notes to provide a verifiable trail for the feature coding and category justification.

The primary strength of this framework is the ability to suggest a comprehensive and simple categorization of LLM-based AI development tools, in a fragmented and fast changing landscape. It provides a clear, multi-dimensional structure that considers target users (IT professionals vs. citizen developers) and technical depth (from raw APIs to fully packaged industry solutions). This makes it highly relevant across different AI development contexts – from IT Professionals teams to citizen developers – as each can locate the tools appropriate for their context.

Another strength is that the framework highlights critical trade-offs between usability and flexibility, offering insight into why no single tool fits all needs. By being explicit about these trade-offs, the categorization helps decision-makers balance short-term ease-of-use against long-term scalability.

In addition, grounding the framework in transparent feature-based coding and documented evidence (Appendix B), rather than vendor labels, improves reproducibility-by-inspection and decision-usefulness for practitioners selecting platforms under real constraints (e.g., latency, scale, observability, data protection, and lock-in risk). It effectively bridges theory and practice, which means it can be used not only to categorize tools, but also to anticipate adoption challenges or requirements for each category. Finally, the categorization is adaptable. As a high-level scheme, it can accommodate new tools or even be further extended with subcategories as the technology landscape evolves, without losing its core logic (the continuum from developer-centric to citizen developer-centric solutions).

While this framework provides a structured categorization, there are also several limitations. Due to the limited availability of scientific publications and systematic literature reviews on the topic, as the field of AIaaS and LCNC platforms is relatively new, most insights were drawn from tools documentation and practical implementations rather than established academic research. A key limitation is the absence of empirical studies linking measured usability and performance to specific feature bundles. Our categorization relies on documented capabilities and public artefacts (docs, SDKs, release notes). To address this limitation, future work should add task-based evaluations, user testing, and field case studies to quantify how features affect adoption and outcomes.

Another limitation is that some tools legitimately span multiple categories, creating category overlaps and blurred boundaries, especially as products evolve rapidly. For instance, Google Vertex AI combines API-level services for model access and deployment (aligning with Category 1: LLM-Based Platforms with API Access) and a LCNC agent builder for conversational workflows (aligning with Category 3: LCNC Platforms). As a result, assigning a single "final" category can under-represent a tool's versatility and may become outdated as platforms add or reposition features. To handle such cases transparently, we apply a multi-affiliation rule: each tool is assigned a primary category based on its dominant feature profile and position on the two-axis map, and a secondary affiliation is recorded when documentation within the evidence window indicates first-class support for the defining development modality of an adjacent category (reported in Appendix B and highlighted in Figure 2). Despite this mitigation, overlap judgments remain sensitive to evolving product scope and require periodic re-coding.

Additionally, the rapid evolution of AI platforms poses a limitation for any static categorization, whether primary or secondary. Tool feature sets change quickly: developer frameworks increasingly add low-code interfaces, while no-code platforms expand their extensibility and customization. Products frequently change branding, scope, or business model, and new tools emerge continuously in a volatile market. As a result, the proposed categories require periodic revision and re-coding to remain accurate over time.

6 Future Directions

Looking ahead, the categorization framework presented in this study serves as an initial structure, but further empirical validation is necessary. Future research should incorporate quantitative and qualitative methodologies, including user studies, case studies, and industry surveys, to assess the framework's applicability across different use cases and refine the categorization and its practical relevance. Similarly, longitudinal studies could track adoption trends, usability challenges, and retention rates providing deeper insights into how organizations integrate these tools over time.

As AI development needs continue to increase, deeper categorization levels may be needed with additional subcategories within each main category. For example, No-Code/Low-Code Platforms (Category 3) could be further divided based on the extent of configurability and customization options, while Developer-Centric AI Frameworks (Category 2) may require differentiation based on the emerging agents' workflow orchestration capabilities. Future research could explore granular categorization models to improve decision-making for organizations and developers selecting AI tools.

For example, beyond platform adoption, additional features, such as cost-effectiveness, security and ethics, could be considered to refine the framework. By continuously refining and validating this framework, researchers and practitioners can contribute to a more structured and actionable understanding of LCNC AI solutions.

In addition, studies could explore more how LCNC tools may meet the needs of citizen developers and their impact on AI adoption, examining barriers to adoption, learning curves, and their effectiveness in enabling non-technical users to create and deploy AI applications. Further research could investigate the role of LCNC platforms in AI literacy, workforce development, and bridge the AI skills gap. Finally, this categorization framework should be revisited and revised regularly, as the AI field is developing rapidly, and adjustments or adaptations might be needed.

7 Conclusion

This study proposed a structured, feature-based categorization of LLM-based AI development tools, addressing the growing complexity of available platforms. By distinguishing API-based platforms, developer-centric frameworks, LCNC solutions, and domain-specific AIaaS platforms, the paper provides a practical reference for both IT professionals and citizen developers. The framework highlights key trade-offs between customization and abstraction, and the two-axis representation supports transparent category placement while making overlaps explicit through a multi-affiliation approach. Although the tool set is indicative and the ecosystem evolves rapidly, the inclusion of anchored scoring definitions and per-tool evidence notes (Appendix B/C) provides a verifiable basis for comparison. Future work should extend the tool sample and incorporate empirical validation (e.g., task-based evaluations and field case studies) to test applicability across diverse use cases. As AI development tools continue to evolve, maintaining a structured and updatable classification will remain essential for navigating the landscape effectively.

References

Ait, A., Izquierdo, J. L. C., & Cabot, J. (2023, March). Hfcommunity: A tool to analyze the hugging face hub community. In 2023 IEEE international conference on software analysis, evolution and reengineering (SANER) (pp. 728-732). IEEE.

Ajimati, M. O., Carroll, N., & Maher, M. (2025). Adoption of low-code and no-code development: A systematic literature review and future research agenda. Journal of Systems and Software, 222, 112300. https://doi.org/10.1016/j.jss.2024.112300

Azizyan, G., Magarian, M. K., & Kajko-Matsson, M. (2011, August). Survey of agile tool usage and needs. In 2011 agile conference (pp. 29-38). IEEE.

Bam, L., & Jewell, W. (2005, June). Review: Power system analysis software tools. In *IEEE Power Engineering Society General Meeting, 2005* (pp. 139-144). IEEE.

Binzer, B., & Winkler, T. J. (2022). Democratizing Software Development: A Systematic Multivocal Literature Review and Research Agenda on Citizen Development. Lecture Notes in Business Information Processing, 463 LNBIP, 244–259. https://doi.org/10.1007/978-3-031-20706-8_17

Castaño, J., Martínez-Fernández, S., Franch, X., & Bogner, J. (2024, April). Analyzing the evolution and maintenance of ml models on hugging face. In Proceedings of the 21st International Conference on Mining Software Repositories (pp. 607-618).

Cowie, K., Rahmatullah, A., Hardy, N., Holub, K., & Kallmes, K. (2022). Web-based software tools for systematic literature review in medicine: systematic search and feature analysis. *JMIR Medical Informatics*, *10*(5), e33219.

Genç, A.C., Turkoglu Genc, F., Kaya, Z.N., Gönüllü, E.: AB1701 HOW TO MAKE A VIRTUAL PRESENTATION USING ARTIFICIAL INTELLIGENCE? Ann Rheum Dis. 82, 2088–2089 (2023). https://doi.org/10.1136/annrheumdis-2023-eular.6257.

Da Costa, L. A. L. F., Melchiades, M. B., Girelli, V. S., Colombelli, F., de Araújo, D. A., Rigo, S. J., Ramos, G. de O., da Costa, C. A., Righi, R. da R., & Barbosa, J. L. V. (2024). Advancing Chatbot Conversations: A Review of Knowledge Update Approaches. Journal of the Brazilian Computer Society, 30(1), 55–68. https://doi.org/10.5753/jbcs.2024.2882

Ehsani, K.L., Rhythm, E.R., Mehedi, M.H.K., Rasel, A.A.: A Comparative Analysis of Customer Service Chatbots: Efficiency, Usability and Application. In: 2023 Computer Applications and Technological Solutions, CATS 2023. Institute of Electrical and Electronics Engineers Inc. (2023). https://doi.org/10.1109/CATS58046.2023.10424303.

Esposito, A., Calvano, M., Curci, A., Desolda, G., Lanzilotti, R., Lorusso, C., & Piccinno, A. (2023). End-User Development for Artificial Intelligence: A Systematic Literature Review. In 29th American Conference on Information Systems (pp. 19–34). https://doi.org/10.1007/978-3-031-34433-6_2

Gartner Research. (2021). Gartner Forecasts Worldwide Low-Code Development Technologies Market to Grow 23% in 2021. https://www.gartner.com/en/newsroom/press-releases/2021-02-15-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-23-percent-in-2021

Gartner Research. (2024). Risk and Opportunity Index: Low-Code Application Platforms. https://www.gartner.com/en/documents/5459763

Hassan, A., Mohammed, F.A., Seyadi, A.Y.: Artificial Intelligence Applications for Marketing. In: Artificial Intelligence and Economic Sustainability in the Era of Industrial Revolution 5.0. pp. 607–618. Springer (2024). https://doi.org/10.1007/978-3-031-56586-1_43.

Holkar, A., Bhosale, S., Harpale, A., Pachangane, V.H. (2024), UNLOCKING THE DEPTH ANALYSIS IF PDF USING ARTIFICIAL INTELLIGENCE, LARGE LANGUAGE MODEL, LANGCHAIN. International Research Journal of Modernization in Engineering Technology and Science

Jain, S. M. (2022). Hugging face. In Introduction to transformers for NLP: With the hugging face library and models to solve problems (pp. 51-67). Berkeley, CA: Apress.

Joachimiak MP, Miller MA, Caufield JH, *et al* (2024) The Artificial Intelligence Ontology: LLM-assisted construction of AI concept hierarchies

Jones, J., Jiang, W., Synovic, N., Thiruvathukal, G., & Davis, J. (2024, October). What do we know about Hugging Face? A systematic literature review and quantitative validation of qualitative claims. In Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (pp. 13-24).

Kaliuta, K.: Integration of AI for Routine Tasks Using Salesforce. Asian Journal of Research in Computer Science. 16, 119–127 (2023). https://doi.org/10.9734/ajrcos/2023/v16i3350.

Käss, S., Strahringer, S., & Westner, M. (2023). Practitioners' Perceptions on the Adoption of Low Code Development Platforms. IEEE Access, 11, 29009–29034. https://doi.org/10.1109/ACCESS.2023.3258539

Li, M., Zhao, Y., Yu, B., Song, F., Li, H., Yu, H., ... & Li, Y. (2023). Api-bank: A comprehensive benchmark for tool-augmented llms. arXiv preprint arXiv:2304.08244.

Long, D. (2021). ACM Reference format: Duri Long, Takeria Blunt, and Brian Magerko. 2021. Co-Designing AI Literacy Exhibits for Informal Learning Spaces. Article, 5(CSCW2). https://doi.org/10.1145/3476034

McKinsey & Company. (2024). The state of AI in early 2024: Gen AI adoption spikes and starts to generate value. https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai. Accessed 6 Aug 2024. https://doi.org/https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai

Moch, E., & Oberdieck, T. (2024). Strategies for Securing and Further Developing AI Expertise: Measures to Avoid

a Shortage of Skilled Workers in the Artificial Intelligence Industry. International Journal of Academic Research and Reflection, 12(1). www.idpublications.org

Morales-Chan, M., Amado-Salvatierra, H. R., Medina, J.A., Barchino R., Hernández-Rizzardini R. , Teixeira, A. M. 2024, Personalized Feedback in Massive Open Online Courses: Harnessing the Power of LangChain and OpenAI API, Electronics.

Ng, D. T. K., Leung, J. K. L., Su, J., Ng, R. C. W., & Chu, S. K. W. (2023). Teachers' AI digital competencies and twenty-first century skills in the post-pandemic world. Educational Technology Research and Development, 71(1), 137–161. https://doi.org/10.1007/S11423-023-10203-6/FIGURES/2

Ozkaya, M. (2018). The analysis of architectural languages for the needs of practitioners. *Software: Practice and Experience*, *48*(5), 985-1018.

Ozkaya, M. (2019). Are the UML modelling tools powerful enough for practitioners? A literature review. IEt software, 13(5), 338-354.

Parthasarathy, V. B., Zafar, A., Khan, A., & Shahid, A. (2024). The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. arXiv preprint arXiv:2408.13296.

Passerini, A., Gema, A., Minervini, P., Sayin, B., & Tentori, K. (2025). Fostering effective hybrid human-LLM reasoning and decision making. Frontiers in Artificial Intelligence, 7, 1464690.

Prinz N, Huber M, Leonhardt J, & Riedinger C. (2024). Unleash the Power of Citizen Development: Leveraging Organizational Capabilities for Successful Low-Code Development Platform Adoption. In Proceedings of the 57th Hawaii International Conference on System Sciences. University of Hawaii at Manoa.

Rios-Campos, C., Vega, S.M.Z., Tejada-Castro, M.I., Zambrano, E.O.G., Perez, D.J.G.B., Calderón, E.V., Rojas, L.M.F., Alcantara, I.M.B.: Artificial Intelligence and Business. South Florida Journal of Development. 4, 3547–3564 (2023). https://doi.org/10.46932/sfjdv4n9-015.

Russo, D. (2024). Navigating the complexity of generative ai adoption in software engineering. ACM Transactions on Software Engineering and Methodology, 33(5), 1-50.

Shlomov, S., Yaeli, A., Marreed, S., Schwartz, S., Eder, N., Akrabi, O., & Zeltyn, S. (2024). IDA: Breaking Barriers in No-code UI Automation Through Large Language Models and Human-Centric Design. arXiv.Org, abs/2407.15673. https://doi.org/10.48550/arxiv.2407.15673

Strobel G, Banh L, Möller F, Schoormann T (2024) Exploring Generative Artificial Intelligence: A Taxonomy and Types

Taheri, M., & Sadjadi, S. M. (2015, July). A Feature-Based Tool-Selection Classification for Agile Software Development. In SEKE (pp. 700-704).

Tolis D, Mystakidis S, Christopoulos A (2025) Generative AI Applications in Education: A Low-Code Approach. Springer Nature Switzerland

Topsakal, Oguzhan & Akinci, T. Cetin (2023). Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast. International Conference on Applied Engineering and Natural Sciences

Viljoen, A., Altın, E. N., Hein, A., & Krcmar, H. (2024). Beyond Citizen Development: Exploring Low-Code Platform Adoption by Professional Software Developers. https://aisel.aisnet.org/amcis2024

Webb. M. 2024. Mapping the landscape of gen-AI product user experience. https : / / interconnected . org / home / 2024 / 07 / 19 / ai - landscape. [Accessed 18-9-2025].

Weber I (2024) Large Language Models as Software Components: A Taxonomy for LLM-Integrated Applications

Wisskirchen, G., Thibault Biacabe, B., Bormann, U., Muntz, A., Niehaus, G., Soler, G. J., & Von Brauchitsch, B. (2017). Artificial Intelligence and Robotics and Their Impact on the Workplace. IBA Global Employment Institute.

Appendix A: Categorization Matrices

A.1 Categorization Matrix of Category 1: LLM-Based Platforms with API access

Category 1	OpenAI Platform	Anthropic Console	Google AI Studio/ Gemini	Mistral	Cohere Platform	Vertex AI Platform	Amazon Bedrock	Azure AI Foundry	Hugging Face	Fireworks AI	Together AI	Groq
LLM Origin (Developed by Company, Third-party, or Both)	Developed by Company	Developed by Company	Developed by Company	Developed by Company	Developed by Company	Both	Both	Both	Third-party	Both	Both	Both
Open Source / Closed Source / Both	Closed Source	Closed Source	Closed source	Both	Closed source	Both	Both	Both	Open Source	Open Source	Open Source	Open Source
Input/Output Modality (Text, Image, Video, Audio, Embeddings)	I: Text, Image, Audio O: Text, Image, Audio, Embeddings	I: Text, Image O: Text	I: Text, Image, Video, Audio O: Text, Image, Embeddings	I: Text, Image O: Text, Embeddings	I: Text O: Text, Embeddings	I: Text, Image, Video, Audio O: Text, Image, Video, Audio, Embeddings	I: Text, Image, Video, Audio O: Text, Image, Video, Audio, Embeddings	I: Text, Image, Audio O: Text, Image, Audio, Embeddings	I: Text, Image, Video, Audio O: Text, Image, Video, Audio, Embeddings	I: Text, Image, Audio O: Text, Image, Embeddings	I: Text, Image O: Text, Image, Embeddings	I: Text, Image, Audio O: Text
Model Fine-Tuning Tools	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Prompt Engineering Support	Yes	Yes	Yes	Documentation Only	Yes	Yes	Yes	Yes	Yes	No	Documentation Only	No
Multiple Model Size Options	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Data Privacy and Security Controls	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Compliance with Industry Standards	Yes	Yes	No	Partial	Yes	Yes	Yes	Yes	Yes	Partial	Yes	Partial
Consultancy and Support Services	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

A.2 Categorization Matrix of Category 2: Developer-Centric AI Frameworks

Category 2	LangChain	LangGraph	Rasa	AutoGen (MS)	Bee Agent	Haystack	LlamaIndex	Semantic Kernel	CrewAI
Open-Source Status	Open Source	Both (Open Source & Commercial Platform)	Both (Open Source & Commercial Pro)	Open Source	Open Source	Open Source	Both (Open-Source Core Framework & Enterprise	Open Source	Both (Open Source & Enterprise Platform)
LLM Agnostic or LLM Specific	LLM Agnostic	LLM Agnostic	LLM Agnostic	LLM Agnostic	LLM Agnostic	LLM Agnostic	LLM Agnostic	LLM Agnostic	LLM Agnostic
Local Model Support	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RAG Capabilities	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Agent Architecture	Single-Agent	Multi-Agent	Single-Agent	Multi-Agent	Multi-Agent	Single-Agent	Multi-Agent	Multi-Agent	Multi-Agent
Context Management	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Prompt Engineering Tools	Comprehensive	Basic	Comprehensive	Basic	Basic	Comprehensive	Comprehensive	Comprehensive	Basic
Response Streaming	Yes	Yes	No	No	No	Yes	Yes	Yes	No
Runtime Features	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Observability	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Deployment Options	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Enterprise Readiness	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes

A.3 Categorization Matrix of Category 3: LLM-Based No Code / Low Code Platforms

Category 3	MS Co-pilot Stu-dio	Vertex AI Agent Builder	Chatgpt's Cus-tom GPTs	Flowi-se	Langflo w	Vec-torshift	Zapier AI	Dify	Bot-press	Voiceflo w	LLMSt ack/Pro mptly	Vellum AI	Rasa Studio
Open-Source Status	Closed Source	Closed Source	Closed Source	Open Source	Open Source	Closed Source	Closed Source	Open Source	Both	Closed Source	Both	Both	Both
Development Inter-face	Node-based	Form-based	Conver-sation based	Node-based	Node-based	Form-Based	Conver-sation based	Node-based	Node-based	Node-based	Form-based/N ode-based	Node-based	Node-based
LLM Agnostic or LLM Specific	LLM Specific	LLM Specific	LLM Specific	LLM Agnos-tic	LLM Agnos-tic	LLM Agnos-tic	LLM Specific	LLM Agnos-tic	LLM Agnos-tic	LLM Agnos-tic	LLM Agnos-tic	LLM Agnos-tic	LLM Agnos-tic
Local Model Sup-port	No	No	No	Yes	Yes	Yes	No	Yes	Yes	No	Yes	No	Yes
RAG Capabilities	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Workflow Orches-tration	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Agent Architecture	Yes	Yes	Limited	Yes	Yes	Yes	Limited	Yes	Yes	Yes	Yes	Yes	Yes
Prompt Engineering Tools	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
Pre-built Templates	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Response Streaming	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	No
Runtime Features	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
Observability	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Hosting Options	Cloud	Cloud	Cloud	Both	Self-hosted	Cloud	Cloud	Both	Cloud	Cloud	Both	Both	Both
API Endpoints	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Security Features	Yes	Yes	Limited	Yes	Limited	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Access Control & Collaboration	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

A.4 Categorization Matrix of Category 4: Domain-Specific AIaaS LLM-based Platforms

Category 4	Salesforce Einstein AI	HubSpot AI	Jasper	Copy.ai	Ada	Zendesk	Synthesia	Kore AI
Industry AI Functions	Industry-Specific	Industry-Specific	Industry-Specific	Industry-Specific	Industry-Specific	Industry-Specific	Industry-Specific	Industry-Specific
Template Library	Rich	Rich	Rich	Rich	Standard	Rich	Rich	Rich
LCNC Accessibility	Business-Ready	Business-Ready	Business-Ready	Business-Ready	Business-Ready	Business-Ready	Business-Ready	Business-Ready
Workflow Configuration	Highly Configurable	Guided Customization	Guided Customization	Guided Customization	Highly Configurable	Highly Configurable	Guided Customization	Highly Configurable
Output Controls	Advanced	Extensive	Standard	Standard	Standard	Extensive	Extensive	Advanced
Business Solutions Integration	Advanced	Advanced	Standard	Extensive	Extensive	Extensive	Standard	Enterprise
Data Exchange	Advanced	Advanced	Standard	Standard	Extensive	Extensive	Standard	Advanced
Performance Metrics	Advanced	Advanced	Standard	Extensive	Extensive	Extensive	Standard	Extensive
Access Controls	Advanced	Advanced	Standard	Standard	Standard	Advanced	Advanced	Advanced
Compliance and Quality Assurance	Advanced	Advanced	Standard	Standard	Advanced	Advanced	Standard	Advanced

Appendix B: Sources, Dates Checked and Rationale

B.1 Category 1 LLM-Based Platforms with API access

Product	Provider	Date checked (YYYY-MM-DD / ISO 8601 format)	Source / URL	Primary category	Secondary	Evidence & Justification
OpenAI Platform	OpenAI	2025-01-14	https://platform.openai.com/docs/overview	Cat.1	-	Official docs describe REST/SDK access, function calling, and streaming suitable for API-driven LLM apps.
Anthropic Console	Anthropic	2025-02-03	https://console.anthropic.com/	Cat.1	-	Console/docs provide API key management and model invocation endpoints for developer integration.
Google AI Studio / Gemini API	Google	2025-03-01	https://ai.google.dev/aistudio	Cat.1	-	AI Studio exposes Gemini API endpoints and credentials setup for programmatic use.
Mistral	Mistral AI	2025-01-27	https://mistral.ai/	Cat.1	-	Product pages link to model APIs and deployment options oriented to developer access.
Cohere Platform	Cohere	2025-02-18	https://cohere.com/	Cat.1	-	Platform materials describe text/embed APIs and enterprise connectivity suitable for API workflows.
Vertex AI Platform	Google Cloud	2025-02-24	https://cloud.google.com/vertex-ai	Cat.1	Cat.3	Documentation details extensive AI/LLM APIs; also offers no/low-code builders. So, Cat.1 primary with Cat.3 secondary.
Amazon Bedrock	AWS	2025-03-04	https://aws.amazon.com/bedrock/	Cat.1	-	Service overview presents unified API access to multiple foundation models for application integration.
Azure AI Foundry	Microsoft	2025-01-19	https://azure.microsoft.com/en-us/products/ai-foundry	Cat.1	-	Product/docs describe provisioning and calling model endpoints via Azure APIs for developers.
Hugging Face API	Hugging Face	2025-03-22	https://huggingface.co/docs/api-inference/	Cat.1	-	Inference API documentation enables programmatic model calls suitable for API-first usage.
Fireworks AI	Fireworks	2025-02-10	https://docs.fireworks.ai/getting-started/introduction	Cat.1	-	Getting-started guide shows API-based model serving and integration steps for developers.
Together AI	Together	2025-01-31	https://docs.together.ai/	Cat.1	-	Documentation provides endpoints, authentication, and usage examples for API integration.

Groq	Groq	2025-03-28	https://groq.com/	Cat.1	-	Product resources indicate hardware-accelerated LLM inference accessible via developer APIs.

B.2 Category 2: Developer-Centric AI Frameworks

Product	Provider	Date checked (YYYY-MM-DD / ISO 8601 format)	Source / URL	Primary category	Secondary	Evidence & Justification
LangChain	LangChain	2025-01-16	https://www.langchain.com/langchain	Cat.2	-	Framework docs present modular components (chains, tools, RAG, agents) for code-first assembly.
LangGraph	LangChain	2025-03-12	https://www.langchain.com/langgraph	Cat.2	-	Library enables graph-based, multi-step agent orchestration under developer control.
Rasa	Rasa	2025-02-19	https://rasa.com/docs/	Cat.2	Cat.3	Open-core conversational AI framework (NLU & flows) oriented to developer-built assistants.
AutoGen (Microsoft)	Microsoft	2025-01-25	https://www.microsoft.com/en-us/research/project/autogen/	Cat.2	-	Multi-agent programming framework let developers define agent roles, tools, and workflows.
BeeAI Framework	BeeAI	2025-03-05	https://framework.beeai.dev/introduction/welcome	Cat.2	-	Developer library for building and coordinating LLM agents with tool integration.
Haystack	deepset	2025-01-28	https://haystack.deepset.ai/	Cat.2	-	Code-first library for search/RAG pipelines with nodes, retrievers, and readers.
LlamaIndex	LlamaIndex	2025-02-07	https://www.llamaindex.ai/llamaindex	Cat.2	-	Framework provides programmatic indexes, retrievers, and pipelines for RAG in Python/JS.
Semantic Kernel	Microsoft	2025-02-21	https://learn.microsoft.com/en-us/semantic-kernel/overview/	Cat.2	-	SDK for planners, connectors, and skills to orchestrate LLM calls and tools in code.
CrewAI	CrewAI	2025-03-27	https://www.crewai.com/	Cat.2	-	Developer toolkit to create multi-agent “crews” with task assignment and tool usage.

B.3 Category 3: LLM-Based No Code / Low Code Platforms

Product	Provider	Date checked (YYYY-MM-DD / ISO 8601 format)	Source / URL	Primary category	Secondary	Evidence & Justification
Microsoft Copilot Studio	Microsoft	2025-01-23	https://learn.microsoft.com/en-us/microsoft-copilot-studio	Cat.3	-	Visual, no/low-code environment to design and deploy copilots/assistants without programming.
Vertex AI Agent Builder	Google Cloud	2025-02-10	https://cloud.google.com/products/agent-builder	Cat.3	Cat.1	Visual multi-agent builder for conversational flows; APIs available via Vertex AI.
ChatGPT's Custom GPTs	OpenAI	2025-02-01	https://openai.com/index/introducing-gpts/	Cat.3	-	No-code configuration of GPT-based assistants with instructions, knowledge, and actions.
Flowise	FlowiseAI	2025-02-14	https://flowiseai.com/	Cat.3	-	Node-based visual builder for LLM apps; supports self-hosting and custom extensions.
LangFlow	LangFlow	2025-02-26	https://www.langflow.org/	Cat.3	-	Drag-and-drop flow editor for LLM pipelines enabling non-developers to assemble apps.
Vectorshift	Vectorshift	2025-03-03	https://vectorshift.ai/	Cat.3	-	LCNC platform to build and deploy AI workflows and assistants via a visual interface.
Zapier (AI features)	Zapier	2025-03-09	https://zapier.com/	Cat.3	-	No-code automation with AI steps/connectors for building LLM-powered workflows.
Dify	Dify	2025-03-15	https://dify.ai/	Cat.3	-	Visual app builder for agents/RAG with optional developer extensions and self-hosting.
Botpress	Botpress	2025-03-21	https://botpress.com/	Cat.3	-	LCNC chatbot platform offering visual flows, channel integrations, and hosted runtime.
Voiceflow	Voiceflow	2025-03-29	https://www.voiceflow.com/	Cat.3	-	Visual conversation designer for assistants and prototypes aimed at non-technical teams.
LLMStack / Promptly	Promptly	2025-04-01	https://www.trypromptly.com/	Cat.3	-	Low-code platform to compose, test, and deploy LLM apps with a visual editor.

Vellum AI	Vellum	2025-04-03	https://www.vellum.ai/	Cat.3	-	LCNC tooling for prompt/version management and evaluations with production connectors.
Rasa Studio	Rasa	2025-01-15	https://rasa.com/product/rasa-studio/	Cat.3	Cat.2	Visual builder for Rasa assistants enabling flow design and configuration without coding.

Category 4: Domain-Specific AIaaS LLM-based Platforms

Product	Provider	Date checked (YYYY-MM-DD / ISO 8601 format)	Source / URL	Primary category	Secondary	Evidence & Justification
Salesforce Einstein AI	Salesforce	2025-01-22	https://www.salesforce.com/eu/artificial-intelligence/einstein-ai-assistant/	Cat.4	-	CRM-embedded assistants, domain workflows, and admin controls for business users (turnkey usage).
HubSpot AI	HubSpot	2025-03-30	https://www.hubspot.com/products/artificial-intelligence	Cat.4	-	AI features integrated into CRM workflows (content, sales/marketing automation) for non-technical teams.
Jasper	Jasper	2025-03-17	https://www.jasper.ai/	Cat.4	-	Marketing-focused generative AI with templates and brand governance; immediate, template-driven use.
Copy.ai	Copy.ai	2025-02-08	http://copy.ai/	Cat.4	-	Domain tool for marketing copy and campaign assets with prebuilt use-case templates.
Ada	Ada	2025-02-28	https://www.ada.cx/	Cat.4	-	CX automation platform providing AI chat, routing, and agent assist tailored to support operations.
Zendesk AI	Zendesk	2025-02-11	https://www.zendesk.com/service/ai/	Cat.4	-	Service desk–oriented AI (triage, intent, agent assist) embedded within Zendesk's platform.
Synthesia (L&D)	Synthesia	2025-01-18	https://www.synthesia.io/learning-and-development	Cat.4	-	AI video generation for learning/training with domain templates, avatars, and studio pipeline.
Kore.ai	Kore.ai	2025-03-02	https://kore.ai/	Cat.4	-	Contact-center/enterprise assistants with domain blueprints and management tooling for business users.

Appendix C: Customization and Abstraction Scoring Tables

Scoring Rubric

C (Customization): 0 = parameter/template only | 1 = sandboxed scripting/logic gates | 2 = extensible runtime (custom code + dependencies) | 3 = full code/infrastructure control

A (Abstraction): 0 = raw API primitives | 1 = managed platform layer (console, evaluation tools) | 2 = visual builder/orchestrator | 3 = turnkey domain product

C.1 Category 1: LLM-Based Platforms with API Access

Pure inference providers (Mistral, Cohere, Fireworks, Together) score A=0. Platforms with evaluation tools (OpenAI, Google AI Studio, Hugging Face) and hyperscalers with MLOps layers (Vertex, Bedrock, Azure) score A=1. Groq scores C=2 due to hardware constraints.

Tool	C	A	Analysis & Justification
OpenAI Platform	**3**	**1**	C=3: Full fine-tuning (SFT/RFT), Assistants API with code interpreter, function calling, batch processing. A=1: Playground and evaluation tools provide managed testing layer beyond raw API endpoints.
Anthropic Console	**3**	**0**	C=3: Extensive tool use capabilities, computer use, caching breakpoints; fine-tuning available via Bedrock/Vertex. A=0: Workbench is a minimal testing utility; core interaction is raw message-based API.
Google AI Studio / Gemini API	**3**	**1**	C=3: Hyperparameter tuning, context caching, safety settings, multimodal support via API. A=1: Structured prompt builder with code export provides an evaluation layer distinct from Vertex's full MLOps.
Mistral	**3**	**0**	C=3: La Plateforme enables fine-tuning of open-weights models, Agents API, MCP integration, self-deployment option. A=0: Console is API management interface; no visual builder or MLOps abstraction layer.
Cohere Platform	**3**	**0**	C=3: Specialized Rerank/Classify/Embed endpoints; fine-tuning for Command R models with hyperparameter control. A=0: API-first design; Coral dashboard provides minimal abstraction over raw endpoints.
Vertex AI Platform	**3**	**1**	C=3: Full custom training, Agent Development Kit, custom containers, hyperparameter tuning, model distillation. A=1: Model Garden, Feature Store, and Agent Builder create a managed MLOps layer.
Amazon Bedrock	**3**	**1**	C=3: Custom model training, continued pre-training, distillation on proprietary data via API. A=1: Managed Agents, Knowledge Bases, and Guardrails abstract deployment and governance complexity.
Azure AI Foundry	**3**	**1**	C=3: Deep fine-tuning of OpenAI/Phi models, Agent Framework, MCP support, Azure ML integration. A=1: Content Safety filters, Prompt Flow evaluation, and Studio interface provide a governance layer.
Hugging Face API	**3**	**1**	C=3: Full model hosting, custom containers via Inference Endpoints, extensive huggingface_hub control, model fine-tuning. A=1: Model Hub, Spaces, and Inference Playground provide a managed discovery and testing layer.
Fireworks AI	**3**	**0**	C=3: High-performance fine-tuning, LoRA serving, custom model deployment for open models via API. A=0: Developer-focused inference primitives; console is API management without workflow abstraction.
Together AI	**3**	**0**	C=3: Full and LoRA fine-tuning with detailed hyperparameter control; broad open model selection. A=0: Pure API infrastructure; minimal platform abstraction beyond endpoint management.
Groq	**2**	**0**	C=2: LPU hardware architecture restricts model compatibility; supports tool calling and LoRA but narrower model breadth. A=0: Deterministic inference API focused on speed; no managed abstraction layer.

C.2 Category 2: Developer-Centric AI Frameworks

All frameworks achieve C=3 (full code control) and A=1 (conceptual abstractions like Chains, Agents, Pipelines that simplify raw API interactions while requiring code to implement).

Tool	C	A	Analysis & Justification
LangChain	**3**	**1**	C=3: Full Python/JS control, custom runnables, tools, callbacks, 1000+ integrations, open-source MIT license. A=1: Chains, Runnables, and LCEL abstract raw API interactions into composable patterns.
LangGraph	**3**	**1**	C=3: Low-level state machine definition, cyclic graphs, fine-grained persistence, human-in-the-loop control. A=1: State and Node abstractions simplify agent orchestration; Studio adds a visual debugging layer.
Rasa	**3**	**1**	C=3: Full NLU pipeline control, custom Python actions, dialogue policies, open-source core framework. A=1: Domain/intent/entity abstractions and YAML configuration simplify conversational AI development.
AutoGen (MS)	**3**	**1**	C=3: Define agent classes, interaction behaviors, and custom tools in Python/.NET with layered architecture. A=1: Team, Chat, and conversable agent constructs abstract multi-agent coordination patterns.
Bee Agent Framework	**3**	**1**	C=3: IBM open-source framework (Apache 2.0), code-level agent definitions, A2A/MCP protocol support. A=1: Templates, memory optimization, and agent patterns provide reusable abstractions.
Haystack	**3**	**1**	C=3: Modular pipeline architecture, 100+ integrations, custom Python component creation. A=1: Retriever, Generator, and Pipeline abstractions significantly simplify RAG composition.
LlamaIndex	**3**	**1**	C=3: Deep control over data ingestion, chunking, embedding models, index structures, 300+ packages. A=1: Index, Query Engine, and Agent abstractions simplify data connection and retrieval patterns.
Semantic Kernel	**3**	**1**	C=3: Microsoft SDK for C#/Python/Java, native functions, enterprise codebase integration. A=1: Plugins and Planners abstract orchestration into familiar software engineering patterns.
CrewAI	**3**	**1**	C=3: Role-based agent definitions in Python, custom tool creation, Crews + Flows architecture. A=1: Crew, Agent, and Task abstractions simplify multi-agent collaboration patterns.

C.3 Category 3: LLM-Based No Code / Low Code Platforms

C=0 (no scripting), C=1 (sandboxed scripting), C=2 (extensible code with dependencies). A=2 (visual builders exposing logic), A=3 (turnkey builders hiding workflow).

Tool	C	A	Analysis & Justification
Microsoft Copilot Studio	**1**	**2**	C=1: Power Fx formulas for logic; extending beyond requires external Bot Framework SDK. A=2: Visual dialog canvas with topic/trigger model; exposes conversation flow structure.
Vertex AI Agent Builder	**2**	**2**	C=2: ADK export to Python/Java available; Cloud Functions enable custom tool integration. A=2: Agent Designer visual canvas with Playbook abstractions; exposes workflow logic.
ChatGPT Custom GPTs	**0**	**3**	C=0: Restricted to Instructions and Actions (OpenAPI specs); no scripting or variable logic. A=3: Conversation-based builder completely hides underlying workflow construction.
Flowise	**2**	**2**	C=2: Custom JS/Python nodes with npm/pip dependencies; open-source allows core modification. A=2: Node-based visual interface wrapping LangChain; exposes RAG and agent wiring.
Langflow	**2**	**2**	C=2: Custom Component nodes support full Python with external library imports via pip. A=2: Visual LangChain/LangGraph builder; users see and manipulate logical flow explicitly.
VectorShift	**2**	**2**	C=2: Python SDK with bidirectional sync between visual builder and code. A=2: Modular drag-drop builder abstracts integrations but shows pipeline structure.
Zapier AI	**1**	**3**	C=1: Code by Zapier provides sandboxed Python/JS (standard libs only, strict timeouts). A=3: Zaps handle auth/webhooks automatically; AI abstracts workflow creation.
Dify	**2**	**2**	C=2: Python/JS code nodes, plugin architecture, open-source with self-hosting option. A=2: Visual workflow canvas for RAG and agents; exposes orchestration logic clearly.
Botpress	**2**	**2**	C=2: Execute Code cards support NodeJS scripting and API interactions within platform. A=2: Visual Flow Editor abstracts NLU training but shows conversation design explicitly.
Voiceflow	**1**	**2**	C=1: JavaScript steps for variable manipulation but no external module imports allowed. A=2: Specialized conversation design canvas; visualizes branching and context flow.
LLM Stack / Promptly	**1**	**2**	C=1: Code processors exist but HTTP API only; no full SDK for deep extension. A=2: No-code builder shows pipeline structure; not fully turnkey abstraction.
Vellum AI	**2**	**2**	C=2: Code Execution Node supports Python/TS; workflows exportable as code via SDK. A=2: Visual builder focused on production workflows; exposes evaluation and routing logic.
Rasa Studio	**2**	**2**	C=2: Integrates directly with Rasa Pro framework; enables code-level customization via connection. A=2: Visualizes CALM conversation flows; abstracts state but shows dialogue structure.

C.4 Category 4: Domain-Specific AIaaS LLM-Based Platforms

All score A=3 as turnkey domain products. C=0 (pure configuration), C=1 (scripting in walled garden), C=2 (extensive platform customization).

Tool	C	A	Analysis & Justification
Salesforce Einstein AI	1	3	C=1: Copilot Builder allows invocable Apex actions and Flows within Salesforce ecosystem. A=3: Turnkey CRM AI; Trust Layer abstracts model safety and data grounding.
HubSpot AI	1	3	C=1: MCP servers, CLI tools, and Agent tools enable custom logic beyond configuration. A=3: Breeze AI suite embedded throughout Hubs with zero-setup deployment.
Jasper	0	3	C=0: Brand Voice and Knowledge Base are primary controls; API triggers workflows, not extends them. A=3: Abstracts prompt engineering into marketing outcomes like campaigns and content.
Copy.ai	0	3	C=0: Workflows chain predefined actions; no custom code logic injection possible. A=3: GTM Operating System abstracts AI into prospecting and content tasks.
Ada	1	3	C=1: HTTP Request blocks and conditional logic enable custom integrations beyond basic config. A=3: Automated CX agent with pre-trained intent recognition; turnkey deployment.
Zendesk AI	1	3	C=1: Sunshine Platform and extensive APIs allow custom app development within the ecosystem. A=3: Pre-trained on billions of CX interactions; Intelligent Triage is plug-and-play.
Synthesia	0	3	C=0: Template variables and Studio Avatars only; no logic scripting or model access. A=3: Complete abstraction of lip-sync, rendering, and audio synthesis pipeline.
Kore.ai	2	3	C=2: Script Nodes support JavaScript for context manipulation; NLP engine allows customization. A=3: XO Platform provides an end-to-end bot lifecycle with pre-built industry templates.