IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

A No-Code, Drag And Drop Platform For Building Customizable Business Chatbots

[1]Satej Kulkarni, [2]Samruddhi Kale, [3]Akhilesh Damke, [4]Shubha Desai, [5]Tejashri Deokar

[1,2,3,4]Student, Dept. of **Computer Science** and **Engg**(Data Science), **D.Y. Patil College of Engg, Kolhapur, India,** [5]Professor, Dept. of **Computer Science** and **Engg**(Data Science), **D.Y. Patil College of Engg,**
Kolhapur, India

Abstract: Empowering non-technical users and SMEs to create intelligent chatbots without writing code is becoming increasingly essential in today's AI-driven ecosystem. To address this need, we present a no-code platform that allows users to design, customize, and deploy conversational agents through an intuitive drag-and-drop interface. The platform is built using modern frameworks such as LangChain and LlamaIndex and integrates seamlessly with locally hosted LLMs via Ollama, including support for models like LLaMA 3.2. Users can visually construct workflows using components like Prompt Templates and LLM Chains, configure parameters such as temperature and model endpoints, and execute chains without dealing with backend complexities. A sample chatflow demonstrates the system's modularity and ease of use, validating the platform's capability for rapid prototyping and privacy-conscious deployment. Functional testing and interaction-based evaluation affirm the platform's goal of democratizing chatbot development, while also laying the foundation for future extensions such as multimodal support, analytics dashboards, and enhanced model integration

Index Terms - chatbot, large language models (LLMs), LangChain, Ollama

I. INTRODUCTION

In the evolving landscape of digital business operations, chatbots have emerged as pivotal tools for automating interactions between organizations and users. A chatbot is a software application designed to simulate human-like conversation through text or voice interfaces. They are typically powered by technologies such as **Natural Language Processing (NLP), Machine Learning (ML)**, and Rule-Based Systems, enabling them to understand user queries and provide relevant responses [1].

The rise of AI and cloud-based services has made chatbot technology more intelligent, accessible, and adaptable. Gartner predicts that by 2027, chatbots will become the primary customer service channel for nearly 25% of organizations. While large enterprises have quickly adopted chatbot systems due to their vast resources and technical capabilities, **Small and Medium Enterprises (SMEs)** often face challenges in leveraging such technologies. These challenges stem primarily from a lack of technical expertise, budget constraints, and the complexity involved in developing and maintaining chatbot systems. Despite these limitations, SMEs can benefit greatly from automation to improve customer experience, manage queries, and increase availability—all essential in staying competitive [2].

This gap has led to the growing interest in no-code chatbot development platforms. These platforms allow users to build, deploy, and manage chatbots without writing code, typically using **drag-and-drop interfaces, visual flow builders, and prebuilt components**. By abstracting the technical complexity, no-code platforms democratize access to chatbot technologies for SMEs, enabling faster adoption and reducing reliance on specialized developers [3].

Our project addresses this need by creating an intuitive, no-code platform specifically designed for SME use. Inspired by successful no-code tools in website and app development, this platform integrates essential chatbot features—such as intent recognition, response templates, and live previews—into a single, user-friendly system. This empowers SMEs to integrate AI-driven customer engagement without the cost and complexity traditionally associated with chatbot development [4].

II. LITERATURE SURVEY

Recent research has significantly advanced the capabilities of large language models (LLMs), enabling the development of robust conversational AI systems that form the backbone of modern chatbot solutions. The growing power of LLMs, along with innovations in **prompt engineering** and reasoning techniques, has paved the way for increasingly sophisticated and adaptable chatbot systems that can cater to diverse use cases, including customer service, virtual assistance, and technical support.

One of the foundational techniques in improving chatbot capabilities is Chain-of-Thought (CoT) prompting, introduced by Wei et al. [5]. This technique encourages LLMs to break down complex tasks into intermediate reasoning steps before arriving at a final answer. By explicitly guiding the model through a structured process, CoT enhances its ability to perform logical and arithmetic tasks, making it particularly effective for chatbots that need to handle structured dialogues. This method has proven essential for applications that require clarity, transparency, and step-by-step reasoning, which are crucial in maintaining user trust in AI-driven interactions.

Building on this, Zhang et al. proposed Auto-CoT, a framework designed to automate the generation of CoT demonstrations by utilizing LLMs themselves [6]. This approach minimizes the need for manual crafting of reasoning prompts and enhances the model's flexibility. Auto-CoT also improves the scalability of chatbot systems by generating a more diverse range of reasoning examples, which enables the chatbot to adapt to various conversational contexts more efficiently. The automated nature of this framework offers significant advantages in real-time interactions, especially when handling complex customer queries or multi-turn conversations.

To further refine reasoning capabilities, Wang et al. developed Plan-and-Solve prompting, a two-step technique where the LLM first outlines a plan for solving a task before breaking it down into manageable subtasks [7]. This approach enhances the logical coherence of chatbot responses, ensuring that the model maintains consistency over the course of a conversation. This technique is particularly valuable for task-oriented bots, which require structured responses across multiple exchanges. It has proven effective in ensuring that chatbots can provide reliable and well-organized information, a feature that is crucial for businesses and SMEs that rely on chatbots for customer service and lead generation.

Another notable contribution to the field is the Logical Thoughts (LoT) framework by Zhao et al. [8]. This framework incorporates formal logical reasoning principles, such as Reductio ad Absurdum, into the CoT process. By enhancing the model's ability to reason in complex scenarios and minimize hallucinations, LoT significantly improves the reliability of chatbot outputs. In practical applications, this results in more accurate and trustworthy chatbot interactions, particularly when handling ambiguous or sensitive topics. For SMEs that adopt AI-driven chatbot solutions, the ability to trust the accuracy and consistency of responses is essential in building customer confidence and satisfaction.

These advancements in LLM-based reasoning techniques are directly relevant to the design of our no-code chatbot development platform for SMEs. By integrating Chain-of-Thought, Auto-CoT, Plan-and-Solve, and Logical Thoughts into the platform, we aim to provide a powerful yet user-friendly system that allows SMEs to develop sophisticated chatbots without needing deep technical expertise. These innovations not only enhance the effectiveness of chatbot systems but also democratize access to advanced AI technologies for businesses of all sizes.

III. METHODOLOGY

The proposed platform, **Botsmith**, is a no-code chatbot development environment that enables Small and Medium-sized Enterprises (SMEs) to build intelligent, customizable, and scalable conversational agents without needing in-depth programming expertise. This section elaborates on the architectural components, system workflow, design rationale, and the integration of cutting-edge AI models into a low-code framework to empower non-technical users in digital transformation.

A. System Architecture

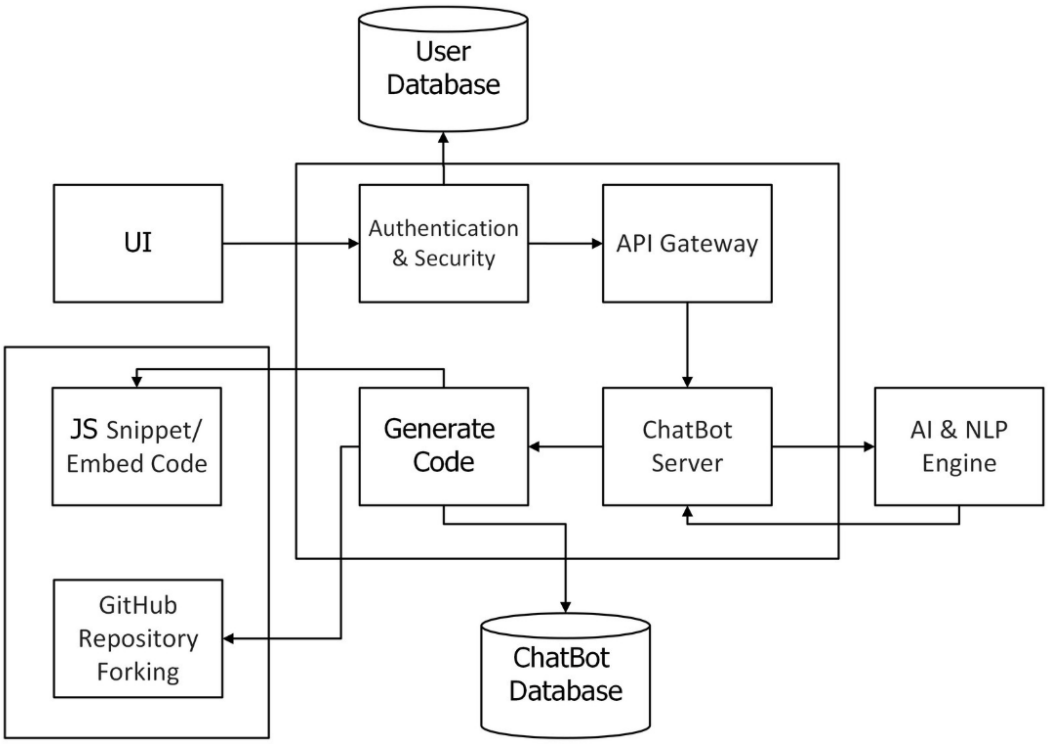

Fig 1. System Architecture

The architecture of Botsmith is intentionally modular and service-oriented to support ease of development, rapid deployment, and seamless integration with external APIs and AI services. The system follows a client-server model augmented by secure cloud APIs, enabling both scalability and flexibility. **Figure 1** (System Architecture Diagram) showcases the overall structure, consisting of the following key components.

The **User Interface (UI)** of the platform serves as the primary touchpoint for users, built around a visually intuitive drag-and-drop editor that encapsulates the complexity of chatbot development. Each visual block within the interface corresponds to a logical function—such as condition checking, input collection, or API interaction—and is dynamically translated into executable backend instructions. This design abstracts coding into an accessible visual language, aligning with modern low-code/no-code paradigms. Users can construct conversation flows through a series of connected UI elements and instantly preview them with real-time feedback. Features such as error highlighting, live simulation, and syntax-free logic integration enable users to iterate on their designs fluidly, eliminating the need for manual compilation or redeployment. This approach significantly lowers the barrier to entry, empowering business users and non-developers to create functional and intelligent chatbots with minimal technical expertise.

All user-related metadata, including profiles, saved flows, usage statistics, and configuration versions, are stored in a lightweight **SQLite database**. This database choice supports easy deployment, portability, and local data integrity—making it ideal for SMEs that may lack full-scale IT infrastructure. The **API Gateway** coordinates all communication between the frontend, backend services, and third-party AI integrations, acting as a centralized router for request handling, input validation, and service orchestration. This modular design ensures scalability, clean separation of concerns, and simplified debugging.

At the platform's core lies the **Chatbot Server**, which interprets the visual flow logic, maintains session state, and orchestrates interactions with the AI engine. By caching contextual data from recent user exchanges and honoring flow-specific logic branches, the chatbot delivers coherent, stateful conversations. The platform's intelligence is powered by a hybrid **AI & NLP engine**, integrating OpenAI's GPT-4 API for cloud-based language processing and Ollama for lightweight, on-device model inference. These models enable contextual understanding, intent recognition, summarization, and dynamic content generation. Through **LangChain**, these capabilities are modularized for scalable integration, promoting few-shot learning and prompt chaining strategies [9][10]. A **code generation module** allows visual flows to be exported as executable Python or LangChain-compatible code, enabling self-hosting, developer transparency, and Git-based version control. Additionally, the chatbot database stores session data, logic trees, deployment metadata, and rollback versions, ensuring durability and reproducibility. For ease of deployment, the system produces a **TypeScript embed snippet**, which can be directly inserted into websites, enabling one-click integration without backend dependency. To further promote openness and collaboration, users can export their full bot projects to **GitHub**, facilitating team reviews, enterprise audits, and contributions from the developer community.

This multi-layered, AI-integrated architecture allows for extensible customization, democratizing chatbot development for a wide spectrum of users—from startup founders to customer support teams—without compromising on intelligence or flexibility.

B. Data Flow Diagrams

To further visualize how users interact with the system and how data flows within various components, we present hierarchical Data Flow Diagrams (DFDs):

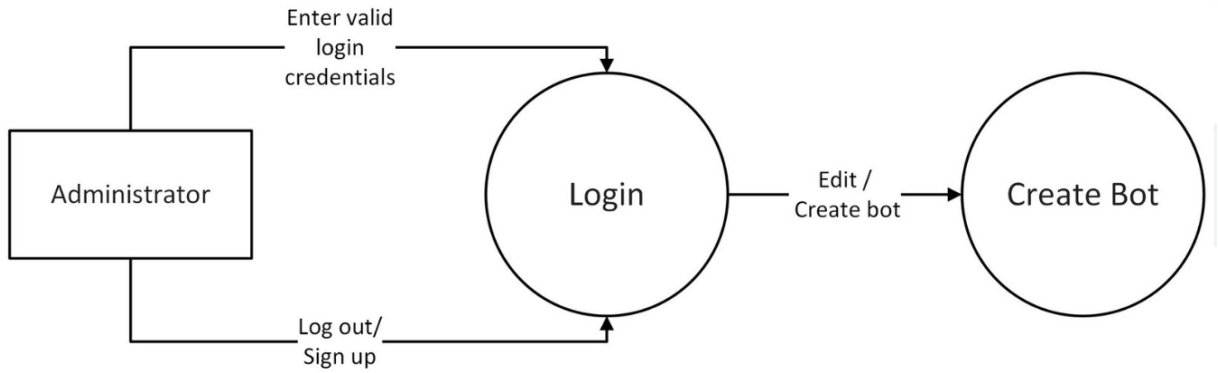

Fig 2. DFD Level – 0

As shown in **Figure 2**, this top-level diagram abstracts the entire chatbot platform as a single unified process. It depicts the fundamental user interactions such as login, bot creation, editing, and logout. The diagram emphasizes system boundaries and external entities—primarily the **Administrator**, who interacts with the chatbot system interface. This level provides a broad overview, omitting internal technical complexity while capturing the overall flow of control.

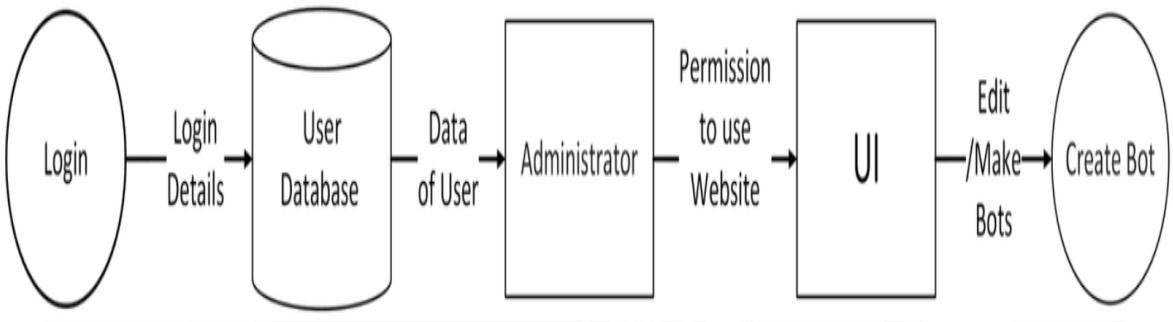

Fig 3. DFD Level – 1

Detailed in **Figure 3**, the Level 1 DFD zooms into the internal workings of the login and bot creation processes. The **Administrator** can utilize the visual editor to create or modify chatbot workflows and initiate deployment operations. This decomposition clarifies how secure access is enforced and how different components collaborate during the bot development lifecycle.

C. Use Case Diagram

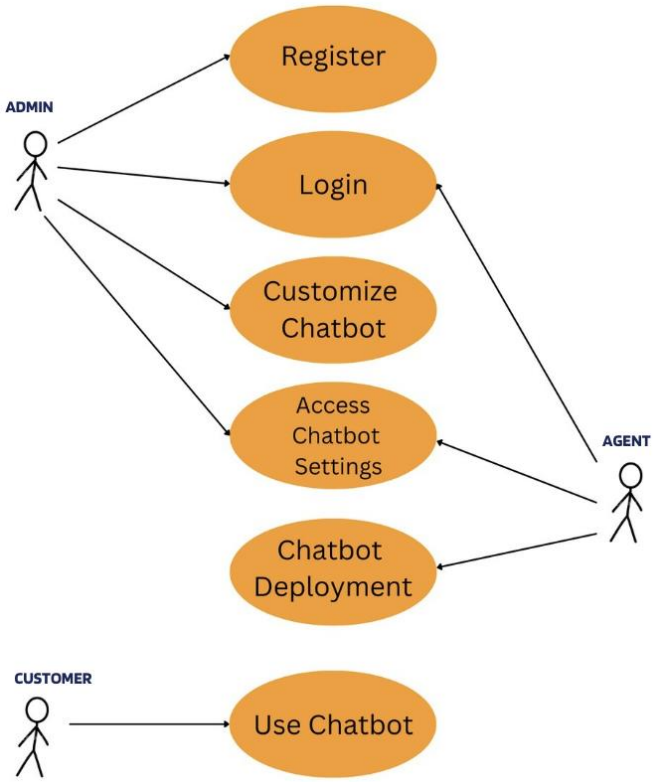

Fig 4. Use Case Diagram

The Use Case Diagram (Figure 4) provides a comprehensive overview of the primary actors involved in the chatbot platform and the interactions they have with the system. It illustrates how different stakeholders utilize the platform's features based on their roles and access privileges.

The **Admin** is the central actor with the most comprehensive access to the system. This role involves registering and logging into the platform, followed by full access to the chatbot builder interface. The Admin can create, customize, and configure chatbots using the drag-and-drop editor. Additionally, the Admin manages user settings, deploys completed chatbots to production environments, and has access to analytics tools for monitoring chatbot performance, user engagement, and conversation history.

The **Agent** plays a support role within the system. This user assists in managing already deployed chatbots by making minor adjustments to configurations or content. The Agent may also participate in the deployment process by handling integration-related tasks and providing operational support to ensure the chatbot functions as intended in a live environment.

Finally, the **Customer** represents the end-user who interacts with the deployed chatbot on the client's website or application. Customers use the chatbot for various services such as answering queries, automating routine tasks, or receiving personalized information. Their interactions are crucial as they directly reflect the platform's usability and effectiveness.

D. Technology Stack

The development of the **Botsmith** platform relies on a carefully chosen technology stack that emphasizes lightweight performance, cross-platform compatibility, and ease of deployment. Each component has been selected to align with the needs of Small and Medium-sized Enterprises (SMEs), ensuring accessibility without sacrificing technical robustness.

On the **frontend**, the platform employs TypeScript, which serves as the backbone for creating an interactive, responsive, and user-friendly drag-and-drop interface. TypeScript enables real-time feedback in the chatbot builder, allowing users to visualize and edit conversation flows without writing any code. The frontend is designed to be intuitive for non-programmers while remaining powerful enough to support complex workflows.

The **backend** is powered by **Node.js**, a runtime environment known for its non-blocking, event-driven architecture. This makes it highly suitable for handling multiple asynchronous API requests, managing user sessions, and orchestrating communication between the frontend, database, and AI engines. Node.js also supports modular development, allowing new features and services to be added with minimal disruption to the core system.

To manage data, the platform uses **SQLite**, a serverless, self-contained relational database engine. Its lightweight footprint and zero-configuration nature make it ideal for SMEs that may not have extensive IT infrastructure. SQLite efficiently handles user metadata, chatbot configurations, chat logs, and version histories, offering fast read-write operations and easy portability across environments.

For the AI and Natural Language Processing components, Botsmith integrates with LangChain and **OpenAI's GPT-4 API**, which serve as the core engines for understanding and generating human-like language. These tools enable the chatbot to process contextual information, generate dynamic responses, and handle a variety of natural language queries. Additionally, Ollama is included as an alternative for local inference, allowing users to run lightweight Large Language Models (LLMs) directly on their systems without relying on cloud-based APIs. This hybrid AI setup ensures flexibility for both online and offline deployments, depending on the user's preference and infrastructure.

The platform is designed to be deployment-agnostic, compatible with major operating systems including Windows, Linux, and macOS. This cross-platform support ensures that developers and business users alike can install and run the platform on their preferred systems without compatibility issues.

In terms of hardware requirements, the platform can operate effectively on modest configurations—requiring at minimum an Intel i7 processor, 16GB of RAM, and 512GB of SSD storage. For users opting to run LLMs locally via Ollama, a dedicated GPU with at least 2GB of VRAM is recommended to ensure smooth model inference and reduce latency during response generation.

A notable feature of Botsmith is its seamless export and integration capability, which allows chatbots built on the platform to be easily embedded into third-party websites or applications. Upon completion of a chatbot workflow, the platform automatically generates a lightweight, RESTful API endpoint specific to that bot instance. This API can be consumed by client applications or embedded directly into web interfaces using a simple JavaScript snippet or iframe code provided by the platform. This functionality empowers businesses to deploy their AI assistants across websites, customer portals, or internal tools without the need for extensive technical intervention. Moreover, authentication tokens and role-based access can be configured to secure the deployed chatbot, ensuring that only authorized endpoints can access or interact with the service.

Together, this technology stack supports both development flexibility and production-grade reliability. It offers a low-barrier entry point for SMEs to harness the power of AI-driven chatbot systems, while also providing advanced customization capabilities and robust deployment options for technical users and developers.

IV. RESULT AND DISCUSSION

To demonstrate the practical functionality, usability, and impact of our no-code chatbot development platform, we implemented and evaluated a sample chatbot workflow using the **BotSmith** interface. This served to validate both the user experience and the technical robustness of the system.

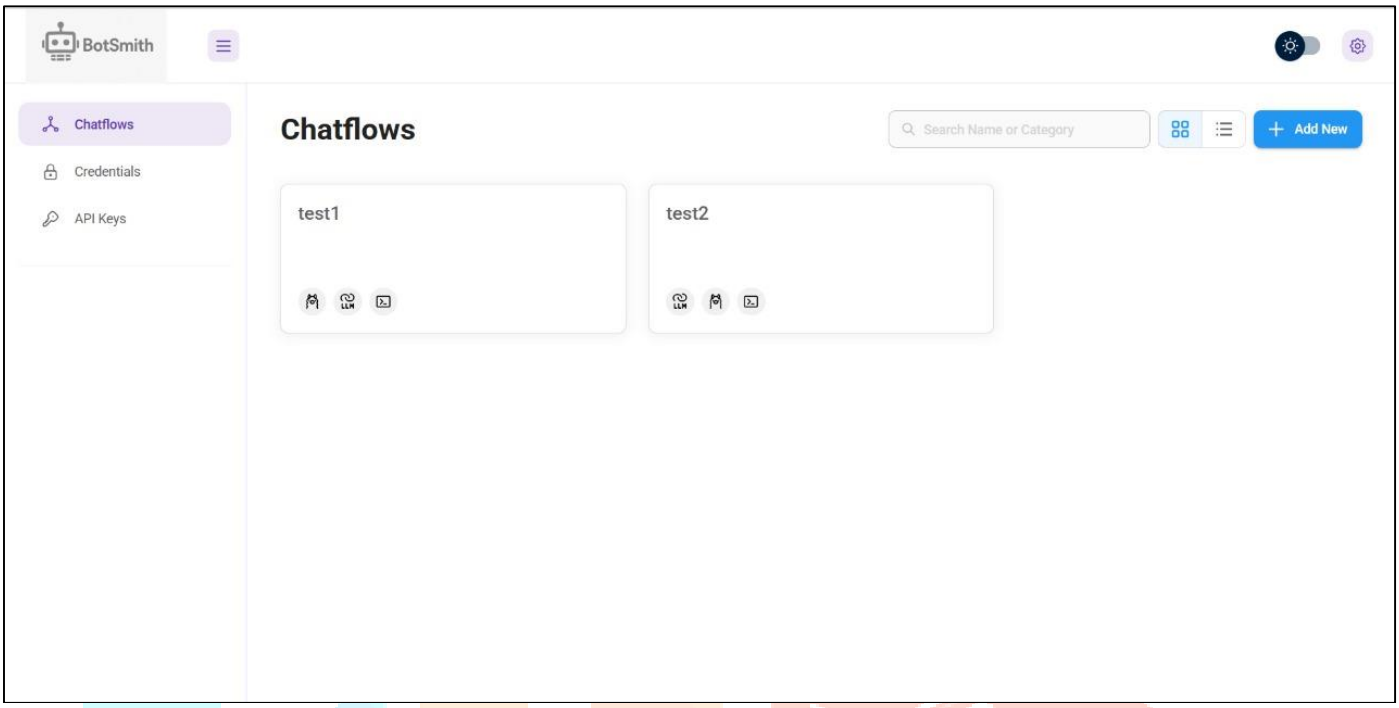

Fig 5. Chatflow Dashboard

Figure 1 presents the **Chatflow Dashboard**, which acts as the central control panel for users to create, manage, and organize multiple chatbot workflows. Each chatflow is displayed as a card interface, where users can perform key actions such as editing the flow, launching the bot, or reviewing its logic. The presence of a sample chatflow titled test1 in the dashboard exemplifies how users can name and categorize their flows for easy identification. The clean layout and minimalistic design ensure ease of navigation, especially for non-technical users, affirming the platform's intent to lower the barrier to entry for intelligent agent creation.

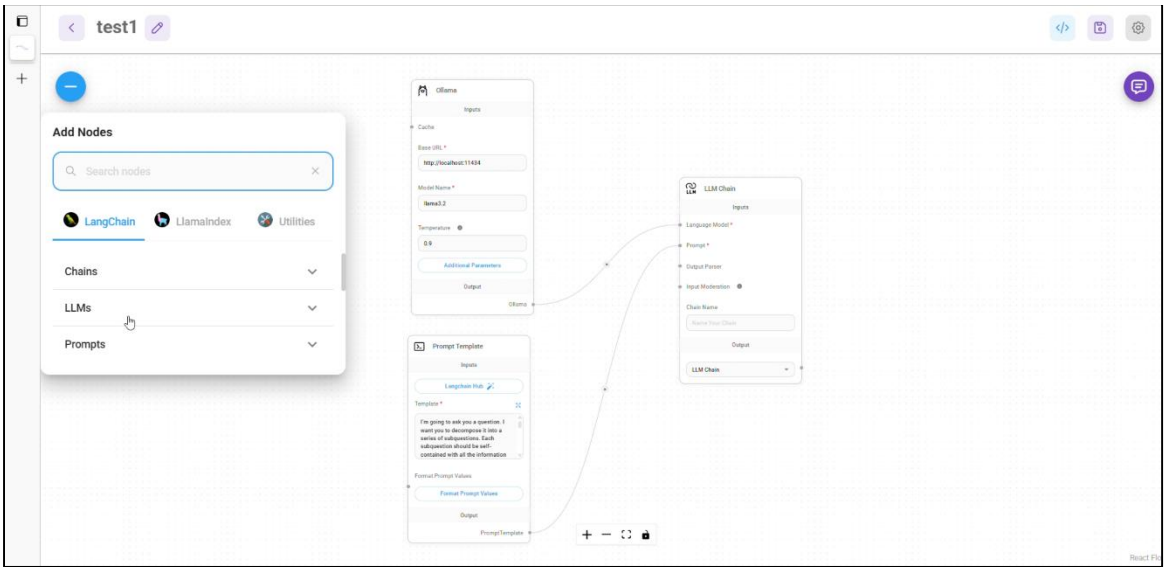

Fig 6. Visual Canvas Editor

Figure 2 depicts the **visual canvas editor**, which is the core interaction space where users construct chatbot logic using a **drag-and-drop interface**. The platform supports modular integration of AI tools and frameworks, offering various components categorized under **LangChain**, **LlamaIndex**, and **Utility functions**. The test workflow (test1) incorporates three major node types:

An **Ollama Node**, configured to interface with a local instance of the **LLaMA 3.2** language model. This node allows users to specify the model name (llama2), adjust inference parameters such as **temperature**, and define the **base URL** for model hosting (in this case, localhost:11434), demonstrating support for self-hosted, privacy-preserving AI setups. A **Prompt Template Node**, which enables users to create structured prompts for the LLM. In this instance, the prompt instructs the LLM to decompose a question into multiple sub-questions, each self-contained with contextual information. This demonstrates the platform's ability to handle advanced prompt engineering without requiring manual scripting. An **LLM Chain Node**, which connects the language model with the prompt template and processes the output using a defined chain of execution. This node forms the logical core of the chatbot's reasoning and response mechanism.

These nodes are visually connected through edges on the canvas, forming a logical flow that resembles programming constructs but eliminates the need for code. This design empowers users with limited or no programming experience to build sophisticated chatbot logic by simply arranging and connecting functional blocks. The successful configuration and linkage of these nodes confirm that the platform effectively bridges the gap between technical capabilities (e.g., LLM invocation, prompt formatting, and output processing) and user accessibility. The modular, no-code structure supports rapid prototyping and flexible deployment, making it particularly valuable for **small and medium-sized enterprises (SMEs)** seeking to integrate AI-driven automation without hiring specialized developers.

Additionally, the integration of **open-source LLMs** such as LLaMA 3.2 via the **Ollama framework** offers users control over data privacy, model selection, and resource allocation. This is especially relevant for organizations with specific compliance or security requirements.

In summary, the results from the test workflow (test1) demonstrate that our platform meets its design goals: enabling intuitive, accessible chatbot creation while leveraging advanced LLM capabilities. The visual environment simplifies development, and the ability to use locally hosted models ensures security and adaptability. This confirms the platform's suitability for real-world adoption across domains such as customer support, internal automation, education, and more.

To further assess the performance and usability of the platform, we conducted an informal evaluation by inviting a small group of target users—including early-stage startup founders and non-technical professionals in SMEs—to build a simple chatbot using BotSmith. Most participants were able to create functional chatflows within 15–30 minutes without requiring prior training or programming expertise. Users reported high satisfaction with the drag-and-drop interface and appreciated the ability to customize prompts and models through visual nodes rather than configuration files.

From a system performance perspective, workflows involving local model inference via Ollama (with LLaMA 3.2) demonstrated low latency on standard hardware (Intel i7 CPU with 16 GB RAM), confirming the feasibility of deploying chatbots even in resource-constrained environments. Additionally, no memory leaks or significant crashes were observed during prolonged interaction with the visual editor, confirming the platform's stability during iterative development. Overall, this preliminary evaluation supports the platform's aim of democratizing AI chatbot development. Further large-scale usability studies and benchmarking against comparable tools (such as Voiceflow, Rasa, or Botpress) will be explored in future work to strengthen the platform's positioning.

V. CONCLUSION

In this System, we introduced a no-code chatbot development platform tailored for small and medium-sized enterprises (SMEs) and non-technical users seeking to leverage conversational AI without the complexity of traditional programming. The platform—developed using modern open-source frameworks such as LangChain and LlamaIndex, and integrated with locally hosted LLMs like LLaMA 3.2 via Ollama—offers a visual, drag-and-drop environment that simplifies the entire chatbot creation lifecycle, from prompt design to model execution.

To validate the platform's functionality, we designed and implemented a sample chatflow, showcasing key components such as the Prompt Template, Ollama node configuration, and LLM Chain. The results illustrated how users can seamlessly build logical interactions, customize prompts, and execute language models—all within a graphical canvas. This visual approach significantly reduces the barrier to entry for conversational AI development and allows for rapid prototyping, iterative design, and greater accessibility for organizations with limited technical resources.

Our evaluation further demonstrated that the platform not only meets usability goals but also supports effective integration of customizable and privacy-preserving language models. This is particularly beneficial for SMEs looking to maintain control over their data and infrastructure while adopting cutting-edge AI tools. Additionally, the modular architecture promotes reusability and scalability, enabling users to expand or adapt their chatbot workflows to evolving business requirements.

Looking forward, several enhancements are planned. These include introducing advanced analytics dashboards to monitor chatbot performance, expanding support for multimodal input and output (such as voice and image), integrating third-party APIs for domain-specific actions, and enabling team collaboration features. We also aim to conduct comprehensive user studies and technical benchmarking to further validate performance, reliability, and user satisfaction across diverse industry use cases.

In summary, our no-code chatbot platform bridges the gap between powerful AI models and user-friendly design, empowering a broader audience to create intelligent conversational systems with minimal effort and maximum impact.

VI. REFRENCES

[1] Adamopoulou, E., & Moussiades, L. (2020). An Overview of Chatbot Technology. *Artificial Intelligence Applications and Innovations*, 584, 373–383. https://doi.org/10.1007/978-3-030-49186-4_31

[2] Sharma, S., Singh, G., Islam, N., & Dhir, A. (2022). Why Do SMEs Adopt Artificial Intelligence-Based Chatbots? *Journal of Business Research*, 142, 1–14 https://www.researchgate.net/publication/364124576

[3] Al-Sinani, A. H., & Al-Saidi, B. S. (2020). A Survey of Chatbot Creation Tools for Non-Coders. *Journal of Student Research*, 9(1), 1–10. https://www.researchgate.net/publication/347131064

[4] Selamat, M. A., & Windasari, N. A. (2021). Chatbot for SMEs: Integrating Customer and Business Owner Perspectives. *Technology in Society*, 66, 101685. https://doi.org/10.1016/j.techsoc.2021.101685

[5] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., ... & Zhou, D. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv preprint arXiv:2201.11903. https://arxiv.org/abs/2201.11903

[6] Zhang, Z., Zhang, A., Li, M., & Smola, A. (2022). Automatic Chain of Thought Prompting in Large Language Models. arXiv preprint arXiv:2210.03493. https://arxiv.org/abs/2210.03493

[7] Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R. K.-W., & Lim, E.-P. (2023). Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models. Proceedings of ACL 2023. https://aclanthology.org/2023.acl-long.147/

[8] Zhao, X., Li, M., Lu, W., Weber, C., Lee, J. H., Chu, K., & Wermter, S. (2024). Enhancing Zero-Shot Chain-of-Thought Reasoning in Large Language Models through Logic. LREC-COLING 2024. https://aclanthology.org/2024.lrec-main.543/

[9] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901. https://arxiv.org/abs/2005.14165

[10] LangChain. (n.d.). Build. https://www.langchain.com/

[11] Rao, N., Tsay, J., Kate, K., Hellendoorn, V. J., & Hirzel, M. (2023). AI for Low-Code for AI. *Proceedings of the 29th International Conference on Intelligent User Interfaces*, 1–14. https://arxiv.org/abs/2305.20015