No-Code App Development Mobile Platform with AI Integration

D. Rahmathulla (21981A4414), G. Pavani (21981A4420), Pathan Raheem (21981A4449), B. Sri Harsha Vardhan (21981A4466)

Under the estimated Guidance of Dr. K. V. Satyanarayana (Professor). Department of Computer Science and Engineering, Raghu Engineering College, Visakhapatnam, Andhra Pradesh - 531162

Abstract: The rapid growth of mobile applications has created a demand for accessible and efficient app development tools, especially for non-technical users and small businesses. However, traditional app development requires significant technical expertise, time, and resources, making it inaccessible to many. To address this gap, we present CodeLoomer, a no-code mobile application that empowers users to build custom mobile apps directly from their smartphones without any coding knowledge.

CodeLoomer leverages a drag-and-drop interface and AI-powered code generation to simplify the app development process. Users can select from a library of pre-built components, customize their app's design and functionality, and deploy it to app stores with just one click. The platform uses advanced machine learning models to generate efficient and clean code for user-created components, ensuring high-quality results with minimal effort.

Keywords: No-code platform, Drag-and-drop interface, Backend automation, Machine learning.

I. INTRODUCTION

In today's digital age, mobile applications have become an integral part of our daily lives, transforming how we communicate, work, and access services. From small businesses to educational institutions, the demand for custom mobile apps is growing rapidly. However, the process of developing an app has traditionally been complex, time-consuming, and expensive, requiring specialized technical skills and resources. This has created a significant barrier for non-technical users and small organizations who lack the expertise or budget to hire developers or invest in sophisticated tools.

To address this challenge, we introduce CodeLoomer, a no-code mobile application that empowers users to create custom mobile apps directly from their smartphones. CodeLoomer is designed to democratize app development, making it accessible to anyone with an idea, regardless of their technical background. By combining an intuitive drag-and-drop interface with AI-powered assistance, CodeLoomer simplifies the entire app development process, enabling users to design, build, and deploy apps with ease.

At the core of CodeLoomer's innovation is its use of transformer-based pretrained models, specifically DeepSeek Coder 1.3B and Gemma 2B, which work together to generate high-quality, context-aware code for user-created components. These advanced models ensure that the generated code is not only efficient but also tailored to the user's specific requirements, making app development faster, smarter, and more intuitive.

CodeLoomer is not just a tool; it is a solution for individuals and organizations looking to leverage mobile technology to solve real-world problems. Whether it's a school managing student assignments, a small business streamlining employee communication, or an event organizer creating a seamless experience for attendees, CodeLoomer provides the flexibility and power to bring these ideas to life. With its mobile-first approach, users can build apps on the go, without the need for a desktop or laptop, making it a truly innovative and accessible platform.

This document explores the design, functionality, and impact of CodeLoomer, highlighting its unique features, use cases, and the technology behind its development. By bridging the gap between ideas and execution, CodeLoomer is poised to

revolutionize the way mobile apps are created, empowering users to innovate and thrive in the digital era.

II. LITERATURE SURVEY

The rise of low-code development platforms (LCDPs) has transformed software development, enabling individuals without programming expertise to create functional applications. This shift is driven by the demand for faster and more accessible app development tools, particularly for small businesses and non-technical users. The paper titled "Supporting the understanding and comparison of low-code development platforms" by Sahay et al. provides a comprehensive analysis of LCDPs, serving as a foundational reference for understanding their architecture, features, and challenges.

LCDPs are defined as cloud-based platforms that allow users to develop applications through visual interfaces and drag-and-drop tools, minimizing the need for traditional coding. These platforms address the shortage of skilled developers by enabling citizen developers—individuals without a programming background—to contribute to software development. At the same time, professional developers can focus on complex tasks like optimizing business logic and ensuring system scalability. LCDPs leverage model-driven engineering (MDE) principles, using visual models and abstractions to automate code generation, deployment, and maintenance processes.

Architecturally, LCDPs are structured into four main layers: the Application Layer, where users interact with visual tools to design applications; the Service Integration Layer, which connects the application to external services via APIs; the Data Integration Layer, which manages data from heterogeneous sources; and the Deployment Layer, which handles the deployment of applications to cloud or on-premise environments. This layered architecture allows users to focus on designing applications without worrying about low-level technical details like infrastructure setup or data integrity.

The core components of LCDPs include the Application Modeler, a visual tool for designing applications using drag-and-drop components; Backend Services, which handle code generation, database management, and API integration; and Collaboration Tools, which support team-based development using agile methodologies. The typical development process involves data modeling, user interface design, business logic specification, integration with external services, and deployment.

The paper analyzes eight leading LCDPs, including OutSystems, Mendix, Zoho Creator, and Microsoft PowerApps, identifying key features such as graphical user interfaces, interoperability with external services, security mechanisms, reusability through pre-built templates, and scalability for large-scale applications. While these platforms share many features, they also have unique strengths tailored to specific use cases like business process automation and customer relationship management.

Despite their advantages, LCDPs face challenges such as interoperability issues due to a lack of standardization, extensibility limitations caused by proprietary constraints, steep learning curves that hinder adoption by non-technical users, and scalability concerns when handling large-scale data and computations. The paper concludes by emphasizing the need for standardization to improve interoperability and reusability, as well as the development of repositories for sharing reusable artifacts and generic mechanisms for integrating different platforms.

In summary, the literature survey highlights the growing importance of LCDPs in democratizing app development and provides valuable insights into their architecture, features, and challenges. These findings serve as a foundation for understanding the potential and limitations of low-code platforms in modern software development.

Features	Out Systems	Mendix	App Craft
No - Code Support	Yes	Yes	Yes
Intelligent Assistance	No	No	Yes
Backend Automation	Limited	Limited	Fully Automated
Database Integration	Yes	Yes	Yes

III. PROPOSED METHODOLOGY

The development of CodeLoomer follows a structured and iterative methodology to ensure the creation of a user-friendly, scalable, and efficient no-code platform for mobile app development. The process begins with requirement analysis, where the needs of non-technical users and small businesses are identified through surveys, interviews, and an analysis of existing low-code platforms. This phase helps define the functional and non-functional requirements, such as ease of use, scalability, and integration with external services. Based on these requirements, the system design phase focuses on creating an intuitive drag-and-drop interface for the mobile app using React Native and the website using Next.js with the shadcn library. The backend, built with Go and the Fiber framework, is designed to handle data processing, and integration with transformer-based AI models (DeepSeek Coder 1.3B and Gemma 2B) for code generation. The database, implemented using PostgreSQL on Railway, ensures secure and scalable data storage.
The development phase involves building the frontend, backend, and AI integration components, followed by rigorous testing and validation to ensure the platform is bug-free and meets user requirements. Once tested, the platform is deployed, with the mobile app published on the Google Play Store and Apple App Store, and the website hosted on a reliable cloud platform. Firebase is used for user authentication, while Azure handles the deployment and management of the AI models. Post-deployment, the platform enters a maintenance and updates phase, where continuous improvements are made based on user feedback and technological advancements. Finally, the platform's success is evaluated through user satisfaction surveys, usage data analysis, and performance metrics, ensuring that CodeLoomer meets its goals of empowering users to build custom mobile applications with ease.

Code generation using AI

The FastAPI-based chatbot leverages transformer-based AI models, specifically Gemma 2B and DeepSeek-Coder 1.3B, to provide intelligent and context-aware responses. These models are deployed using Ollama, a lightweight framework for running AI models in Docker containers, and are integrated into the FastAPI application to enable seamless interaction with users.

Key features of the AI models: The Gemma 2B model is a lightweight and efficient language model designed for general-purpose text generation. It is capable of understanding and generating human-like text, making it ideal for conversational applications. In the chatbot, Gemma 2B is used to provide natural and contextually relevant responses to user queries. On the other hand, the DeepSeek-Coder 1.3B model is a specialized AI model trained for code generation and understanding. It excels at generating code snippets, debugging, and providing programming-related assistance. In the chatbot, DeepSeek-Coder 1.3B is used to assist users with coding tasks, such as generating code, explaining algorithms, or fixing errors.

How the AI models work: The chatbot interacts with the AI models through the Ollama API, which runs the models in a Docker container. When a user sends a prompt, the FastAPI backend sends the request to the appropriate AI model (Gemma 2B or DeepSeek-Coder 1.3B) via the Ollama API. The AI model processes the prompt and generates a response, which is streamed back to the FastAPI application in real-time. The response is then returned to the user in a structured JSON format, ensuring a smooth and interactive experience.

Frontend Development

The frontend of the platform is divided into two main components: the mobile app for app development and the website for user metrics and analytics. Each component is designed with a specific purpose, ensuring a seamless and efficient user experience.

Mobile app for app development: The mobile app is the core component of the platform, enabling users to build mobile applications directly from their smartphones. It is developed using React Native, a cross-platform framework that ensures compatibility with both Android and iOS devices. The app features a drag-and-drop interface, allowing users to design their apps visually by selecting and arranging components such as buttons, forms, and lists. This intuitive interface eliminates the need for

coding knowledge, making app development accessible to non-technical users.

The mobile app communicates with the backend via API requests, sending user inputs such as drag-and-drop actions or component configurations. Once the backend processes these inputs and generates the corresponding code using the AI models, the app integrates the output into the project. This seamless interaction ensures that users can build apps efficiently without worrying about the underlying technical details. The mobile app is designed with a focus on user experience, providing a simple and intuitive interface that empowers users to bring their app ideas to life.

Website for user metrics and chatbot: The website serves as a complementary component to the mobile app, providing users with detailed insights into their app development activities. Built using Next.js and the shadcn library, the website offers a fast and responsive interface for accessing user metrics and analytics. Unlike the mobile app, the website does not support app development; instead, it focuses on displaying user data in the form of graphs, charts, and numerical representations.

The website also includes a chatbot feature, which assists users with queries related to the platform, provides guidance on using the mobile app, and offers support for troubleshooting issues. The chatbot is integrated into the website interface, ensuring that users can access help whenever needed.

Backend Architecture

The backend of the platform serves as the central hub that connects the frontend (user interface) with the AI-powered code generation system, ensuring smooth communication and efficient app development. Built using Go and the Fiber framework, the backend is designed for high performance, scalability, and reliability. It handles user inputs, processes data, interacts with AI models, and manages database operations, making it a critical component of the platform.

When a user interacts with the frontend—such as dragging and dropping a component or describing a feature—the input is sent to the backend via an API request. The backend processes this input, validates it, and prepares it for the AI models. The processed input is then sent to the AI models hosted on Azure, which include DeepSeek Coder 1.3B and Gemma 2B. The DeepSeek Coder model generates the initial code structure based on the input, while the Gemma 2B model refines the code, optimizing it for performance and ensuring it adheres to the platform's standards. Once the code is generated, the backend receives it and prepares it for delivery to the frontend.

The backend also plays a crucial role in data storage and management. It uses PostgreSQL, hosted on Railway, to store user data, app designs, and generated code. This ensures that all information is securely stored and easily accessible for future use.

To ensure seamless operations, the backend is equipped with several key components. The API Gateway acts as the entry point for all frontend requests, routing them to the appropriate backend services. The AI Service Connector handles communication between the backend and the AI models hosted on Azure, ensuring efficient and secure data transfer. The Database Manager manages interactions with the PostgreSQL database, ensuring data integrity and efficient storage. The Authentication Service integrates with Firebase to handle user authentication and authorization, while the Code Generator orchestrates the code generation process, ensuring that the AI models work together seamlessly.

Azure plays a vital role in backend operations by providing the infrastructure to host and run the AI models. It ensures scalability through its auto-scaling capabilities, allowing the backend to handle increased workloads during peak usage. Azure also provides robust security features, such as encryption and access control, to protect sensitive data and ensure secure communication between the backend and the AI models. Additionally, Azure's monitoring tools enable real-time tracking of AI model performance and backend operations, ensuring that the platform runs smoothly and efficiently.

Workflow example:

- A user drags and drops a button component in the frontend.
- The frontend sends the input to the backend via an API request.
- The backend processes the input and sends it to the AI models hosted on Azure.
- The AI models generate the code for the button and send it back to the backend.
- The backend stores the code in PostgreSQL.

Deployment Process:

- Backend: Hosted on Railway
- Frontend: Deployed via Vercel

Documentation:

- User Manual: Step-by-step guide for app creation
- Developer Guide: API documentation

IV. CONCLUSION

The AI-powered no-code mobile app development platform successfully bridges the gap between technical complexity and user-friendly app creation, empowering non-technical users to build functional applications through its intuitive drag-and-drop interface and AI-generated code capabilities. By combining transformer-based models (DeepSeek Coder 1.3B and Gemma 2B) with a robust React Native/Go/PostgreSQL architecture, the platform delivers a scalable solution that maintains performance while ensuring accessibility. The implementation of rigorous testing protocols and security measures guarantees reliability, making professional grade app development available to entrepreneurs, educators, and small businesses without requiring coding expertise. This innovation not only democratizes mobile app development but also sets a foundation for future advancements in AI-assisted no-code technologies.

V. REFERENCES

[1] Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). Supporting the Understanding and Comparison of Low-Code Development Platforms. *Journal of Software Engineering.*

[2] Huang, S.-Y., Yeh, C.-H., Wang, F., & Huan, C.-H. (2020). Android Black-box Coverage Analyzer of Mobile App Without Source Code. *Journal of Software Testing.*

[3] Ollama Documentation. Retrieved from [https://ollama.com/library]

[4] React Native Documentation. Retrieved from[https://archive.reactnative.dev/docs/getting-started]

[5] PostgreSQL Documentation. Retrieved from [https://www.postgresql.org/]

[6] Website url :https://www.codeloomer.in/

[7] App development: https://github.com/harshavardhanyadav2004/app-react-native-mobile

VI. Copyright Transfer and Declaration Form

It is mandatory on the part of the corresponding author to furnish the “Copyright Transfer and Declaration Form” signed at the time of submission of the manuscript for publication. We must have this form before your paper can be published in the Journal. The form can be downloaded from the Journal Website i.e www.ijcst.com. Author should send the scanned copy (Duly signed by the Corresponding author) of the “Copyright Transfer and Declaration form

VII. Author Profiles

D. Rahmathulla includes no-code development platforms and generative AI.

G. Pavani specializes in backend development and database management.

P. Raheem focuses on frontend development and user experience design.

B. Sri Harsha Vardhan researches machine learning and AI-driven application development.

Signatures:

D. Rahmathulla | G. Pavani | Pathan Raheem | B. SriHarshaVardhan

Supervisor: Dr. K. V. Satyanarayan