International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 02 | Feb - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Code Craft AI- AI Powered Code Generator

S.S. Harale [1] ,Anjali patil [2] ,Vaishnavi Sagare [3],Adarsha Bodgire [4]

[1] professor, [2,3,4] Artificial intelligence & Machine Learning Students

Department of Artificial Intelligence & Machine learning Engineering,

Shri Ambabai Talim Sanstha Sanjay Bhokare Group Of Institutes, Miraj 416410, India

ABSTRACT

The rapid advancement of Artificial Intelligence has significantly transformed software development by automating code generation and improving developer productivity. Code Craft AI is a local AI-powered code generator designed to generate programming code without relying on cloud-based APIs. The system uses locally hosted large language models to generate source code based on user prompts and provides a download feature to save the generated code in specific file formats such as .py, .html, .css, .js, and .php. The application is developed using Python and Flask, offering a simple web-based interface for users. Since the system operates locally, it ensures data privacy, low latency, and offline usability. CodeCraft AI is especially useful for students, beginners, and developers who want fast, secure, and customizable code generation. The project demonstrates an efficient approach to integrating local AI models with a user-friendly interface and file management system, with future scope for multi-language support and advanced code optimization.

1. INTRODUCTION

Artificial Intelligence has become an essential tool in modern software development, enabling automation of repetitive tasks such as code writing, debugging, and optimization. Traditional AI-based code generators rely heavily on cloud services, which raises concerns regarding data privacy, internet dependency, and usage costs. To overcome these limitations, CodeCraft AI is developed as a local AI-based code generation system.

This project focuses on generating programming code using locally hosted AI models, eliminating the need for external APIs or continuous internet access. The system allows users to input natural language prompts describing their requirements, and the AI model generates the corresponding code. Additionally, the generated code can be downloaded in user-selected file formats, making it practical for real-world development use.

CodeCraft AI is suitable for educational institutions, developers, and offline environments where cloud access is restricted. The system is designed to be lightweight, secure, and easily extendable, showcasing the practical implementation of local AI models in software engineering.

2. LITERATURE REVIEW

1. **Brown et al.,** discuss the capabilities of large language models in automated code generation and highlight their effectiveness in reducing development time and human errors. Their study emphasizes the growing role of AI in assisting programmers.
2. **Li and Chen,** explore local deployment of AI models for software applications. Their research shows that locally hosted models provide better data privacy and lower latency compared to cloud-based systems.
3. **Singh and Sharma,** present an AI-powered developer assistant that generates code snippets based on user input. Their work demonstrates how natural language processing can simplify programming tasks for beginners.
4. **Kumar et al.,** study offline AI systems and conclude that local AI deployment is beneficial for educational and secure environments where internet access is limited.
5. **Patil and Deshmukh,** analyze web-based AI tools using Flask and Python. Their findings show that Flask is an effective framework for building lightweight AI-driven web applications.

3. METHODLOGY

- **Module 1: User Interface Module**

a. **Web-Based Interface :** A responsive web interface is designed using HTML and CSS to ensure smooth interaction. It provides an accessible platform for users to communicate with the system easily.

b. **Prompt Input System :** Users can enter natural language instructions describing the required code logic. This allows even non-technical users to interact with the system effectively.

c. **Programming Language Selection :**The interface allows users to select the desired programming language or file type .This ensures that the generated code matches the user's development requirements.

d. **Generate Code Button** : The button submits the user prompt securely to the backend for AI processing. It initiates the code generation workflow within the system.

e. **Display Generated Code :** Generated code is displayed in a formatted and readable manner on the webpage. This helps users easily review and understand the output.

f. **User-Friendly Design :** The interface follows a simple and intuitive design approach. It is suitable for students, beginners, and non-expert users.

g. **Real-Time Interaction :** The system generates and displays code instantly after prompt submission. This ensures a smooth and interactive user experience.

h. **Input Validation :** The system validates user input to prevent empty or incorrect submissions. This reduces unnecessary processing and improves system reliability.

i. **Minimal Learning Curve :** The interface requires minimal effort to learn and operate. Users can generate code without prior technical training.

j. **Primary Interaction Layer :**This module acts as the main communication bridge between the user and AI engine. All user requests originate and are managed through this layer.

- **Module 2: Local AI Processing Module**

a. **Local AI Model (Qwen 2.5 Coder 7B) :** The system uses Qwen 2.5 Coder 7B deployed locally through Ollama. This model is responsible for intelligent and accurate code generation.

b. **Prompt Processing :** User input is analyzed and structured before being sent to the AI model. This improves the relevance and quality of the generated code.

c. **Code Generation Logic :** The AI model generates code based on contextual understanding of the prompt. It ensures logical correctness and programming accuracy.

d. **Offline Execution :** Since the AI model runs locally, internet connectivity is not required. This enables uninterrupted usage in offline environments.

e. **Data Privacy Protection :** All prompts and generated outputs remain on the local system. This ensures complete data security and user privacy.

f. **Low Latency Response :** Local processing significantly reduces response time. Users receive code outputs faster compared to cloud-based systems.

g. **Model Optimization :** The system efficiently manages hardware resources during AI execution. This ensures stable performance and reduced system load.

h. **Error Handling :** The module detects and manages invalid or incomplete prompts. This prevents system crashes and improves reliability.

i. **Scalable AI Architecture :** The AI model can be upgraded or replaced as technology evolves. This ensures long-term scalability of the system.

j. **Core Intelligence Module :** This module functions as the brain of the CodeCraft AI system. All intelligent decision-making and code generation occur here.

- **Module 3: Backend Control Module (Flask & Python)**

a. **Flask Framework Usage :** Flask is used to develop a lightweight and efficient backend system. It manages routing and communication between system components.

b. **Request Handling** : User requests are securely received and processed through Flask routes. This ensures safe and controlled data flow.

c. **AI Model Integration :** The backend integrates with Ollama to access the local AI model. This enables seamless communication between Flask and the AI engine.

d. **Prompt Forwarding :** Processed prompts are forwarded to the AI model for execution. This ensures accurate input delivery to the AI system.

e. **Response Handling :** Generated code is received from the AI model and formatted properly. It is then prepared for display on the frontend.

f. **Session Management :** The backend maintains session continuity during user interaction. This ensures a smooth and consistent user experience.

g. **Error Logging :** Backend errors are logged systematically for debugging purposes. This helps in system maintenance and performance monitoring.

h. **Lightweight Architecture :** Flask provides minimal overhead and fast response times. This improves the overall efficiency of the system.

i. **Secure Execution Flow :** Security measures are implemented to prevent unauthorized access. This ensures safe execution of backend operations.

j. **Application Backbone :** This module controls and coordinates the entire system workflow. It acts as the backbone of the application architecture.

- **Module 4: File Generation & Download Module**

a. **File Format Selection :** Users can choose different file formats such as .py, .html, or .js. This allows flexibility in code usage and deployment.

b. **Dynamic File Creation :** The system dynamically creates files containing the generated code. This ensures quick and accurate file generation.

c. **Download Feature :** Generated files can be downloaded directly through the browser. This improves user convenience and accessibility.

d. **Correct File Encoding :** Files are saved using proper encoding standards. This prevents syntax errors and compatibility issues.

e. **Automatic File Naming :** The system assigns meaningful file names automatically. This improves file organization and usability.

f. **Secure File Handling :** Temporary files are used to reduce security risks. This ensures safe handling of generated code files.

g. **Multi-Language Support :** The module supports multiple programming languages. This enables a unified platform for diverse coding needs.

h. **Developer Convenience :** The module eliminates manual copying and pasting of code. This improves developer productivity and efficiency.

i. **Improved Workflow :** The code generation and download process is streamlined. This accelerates development and testing activities.

j. **Output Delivery Module :** This module delivers the final usable code to the user. It completes the code generation pipeline.

- **Module 5: Frontend Technologies Module (HTML & CSS)**

a. **HTML Structure Design :** HTML is used to create the structural layout of the application. It defines the organization of all frontend components.

b. **CSS Styling :** CSS enhances the visual appearance and layout responsiveness. It improves overall user interface aesthetics.

c. **Form Elements :** Input fields, buttons, and dropdowns are implemented using forms. These elements enable effective user interaction.

d. **Responsive Layout :** The interface adapts smoothly to different screen sizes. This ensures usability across multiple devices.

e. **User Interaction Handling :** UI elements respond accurately to user actions. This improves system usability and feedback.

f. **Clean Code Presentation :** Generated code is displayed in a clear and readable format. This enhances user understanding and readability.

g. **Minimalistic Design Approach :** A clean and clutter-free design is followed. This improves focus and user experience.

h. **Browser Compatibility :** The frontend works consistently across modern browsers. This ensures accessibility for all users.

i. **Lightweight Frontend :** The frontend is optimized for fast loading speeds. This ensures smooth and efficient performance.

j. **Visual Output Layer :** This module acts as the presentation layer of the system. It visually represents system outputs to the user.

4. RESOURCES REQUIRED

Hardware Components

Computer / Laptop : The primary device used for development, model execution, and running the CodeCraft AI application.

Processor (CPU) : A multi-core CPU (e.g., Intel i5/i7 or AMD Ryzen) that handles the computation required for running the AI model and backend processes.

RAM (8GB minimum, recommended 16GB) : Random Access Memory stores temporary data and ensures smooth running, especially when loading and running the Qwen model locally.

Storage (SSD Recommended) : Solid State Drive provides faster loading and data access, which helps when storing the AI model and generated files.

Internet (Optional for Setup) : While not required for offline code generation, internet is useful for downloading tools, packages, and the AI model during initial setup.

Display Monitor : Used to visualize the interface, code editor, and output during development and testing.

Power Supply : Constant and stable power ensures uninterrupted development and running of local AI models.

software Requirements

Python : A high-level programming language used to write the backend logic for CodeCraft AI, including Flask routes and model integration.

Flask : A lightweight Python web framework used to build the web server, handle requests, and manage interaction between the frontend and the AI model.

Ollama : A local AI model deployment tool that allows running large language models like Qwen 2.5 Coder 7B on the local machine without cloud access.

Qwen 2.5 Coder 7B Model : An instruction-tuned AI code generation model used to generate programming code based on user prompts.

HTML (HyperText Markup Language) : Used to structure the web pages and UI components such as input fields, buttons, and display areas for generated code.

CSS (Cascading Style Sheets) : Used to style the web interface, making the UI visually appealing and user-friendly.

Web Browser (e.g., Chrome, Firefox) : Used to access and interact with the CodeCraft AI interface via the Flask-hosted web application.

Text Editor / VS Code : An integrated development environment used to write and debug Python, HTML, and CSS code.

Terminal / Command Prompt : Used to run Python scripts, start the Flask server, and execute Ollama commands to run the AI model.

5. CONCLUSION

CodeCraft AI successfully demonstrates the practical implementation of a local AI-based code generation system that operates without dependency on cloud services. By integrating Python and Flask with Ollama-hosted Qwen 2.5 Coder 7B, the project provides an efficient platform capable of generating accurate programming code from natural language prompts. The inclusion of a file download feature further enhances usability by allowing users to directly save generated code in required file formats.

Running the AI model locally ensures data privacy, offline functionality, reduced latency, and cost efficiency, making the system suitable for educational institutions and secure development environments. The web-based interface developed using HTML and CSS offers simplicity and ease of use, especially for students and beginners. Overall, the project highlights how local large language models can be effectively applied to real-world software development tasks. With future enhancements such as multi-language support, intelligent debugging, and IDE integration, CodeCraft AI has strong potential to evolve into a complete AI-assisted development tool.

REFERENCES

[1] Qwen Team, *Qwen 2.5 Coder: Large Language Model for Code Generation*, Alibaba AI Research, 2024.

[2] Ollama Documentation, *Running Large Language Models Locally*, Ollama Open-Source Platform, 2024.

[3] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, O'Reilly Media, 2018.

[4] T. Brown *et al.*, "Language models as code generators," *Journal of Artificial Intelligence Research*, vol. 68, pp. 321–345, 2020.

[5] World Wide Web Consortium (W3C), *HTML5 Specification*, 2023.

[6] Mozilla Developer Network, *CSS Styling and Web Design Documentation*, MDN Web Docs, 2023.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[8] R. Singh and P. Sharma, "AI-assisted software development tools," *International Journal of Computer Applications*, vol. 185, no. 12, pp. 15–20, 2022.