CodeGenie: AI-Powered Code Generation Tool

Team Members

- Rahul Biju Veyccal (22BCT0237)
- Shaik Kaif Sharif (23BCE9393)
- Suyash Pandey (22BCT0111)
- Vansh Saxena (23BCE10177)

1. Project Overview

CodeGenie is a web-based AI-powered application designed to generate code from natural language prompts using large language models (LLMs). Built with **Streamlit** and powered by a locally running **CodeLlama** model, CodeGenie simplifies coding tasks for users by converting their requests into functional code. The app is developed as a group project hosted and demonstrated via **Google Colab**, with backend processing handled by the **llama-cpp-python** library and frontend rendering using Streamlit.

2. Objective

To develop an intelligent and user-friendly platform that:

- Takes code-related prompts from the user
- Sends them to a pretrained code-generation model (CodeLlama)
- Displays the generated code output on a clean and responsive UI

3. Requirements Gathering

The primary requirements of this project were determined based on usability, accessibility, and performance. These include:

- A simple web interface accessible via browser (Streamlit)
- Efficient backend capable of running on limited resources (Colab CPU/GPU)
- Support for prompt-based code generation in multiple languages
- Easy deployment for demonstration purposes (LocalTunnel)
- Clear documentation for replication and contribution (GitHub, requirements.txt)

4. Key Features

- Text prompt input box for user queries
- Real-time code generation using CodeLlama via llama-cpp-python
- Output display with syntax highlighting
- Processing time display
- Sidebar with usage guide and example prompts
- Portable setup using LocalTunnel for public access

5. Technology Stack

Component	Technology Used
Language	Python
UI Framework	Streamlit
AI Model	CodeLlama-7B (4-bit quantized, GGUF format)
Hosting (Development)	Google Colab
Deployment	LocalTunnel
Version Control	GitHub

6. Implementation Workflow

1. **System Architecture and Setup**:
    - Organized project folder with .venv, requirements.txt, .gitignore, and model files.
    - Virtual environment ensures smooth dependency management.
    - Model file (codellama-7b-instruct.Q4_K_M.gguf) is stored locally and reused.
2. **Model Loading**:
    - Handled in CodeModel.py with efficient loading using @lru_cache.
    - Uses llama-cpp-python for running quantized model on CPU with multithreading.
3. **Code Generation Logic**:
    - generate_code() accepts the prompt, sends it to the model using create_completion().
    - Parameters like max_tokens=256 and temperature=0.1 ensure focused and reliable output.
    - Measures and returns response time for performance tracking.
4. **Frontend UI**:
    - Built with Streamlit (app.py).
    - Users input prompts, submit via form, and view the generated code with syntax highlighting.
    - Sidebar includes a guide and example prompts.
5. **Hosting and Access**:
    - Hosted on Google Colab.
    - Exposed via LocalTunnel for sharing a live demo.

7. Project Milestones

- Set up the project repository and base files
- Design and test the user interface using Streamlit
- Integrate llama-cpp-python for local model inference
- Optimize model using 4-bit quantization
- Test and demonstrate code generation
- Deploy in Google Colab and expose with LocalTunnel
- Document the code and usage instructions

- Prepare final report and demo video

8. Deliverables

- app.py: Main application file
- CodeModel.py: Backend code for model loading and generation
- requirements.txt: List of dependencies
- Output screenshots: Captured from Colab during execution
- Demonstration video: A short screen recording showing usage of the app
- Final report: This document
- Public GitHub Repository

9. Project Usage (Demo Instructions)

1. Open Google Colab and upload the app.py and CodeModel.py
2. Install the required packages:
3. !pip install streamlit llama-cpp-python
   !npm install -g localtunnel
4. Run Streamlit in the background:
5. !streamlit run app.py –server.address=localhost > /content/logs.txt 2>&1 &
6. Start LocalTunnel:
7. !npx localtunnel –port 8501
8. Click the URL generated to view the app and test prompts.

10. References

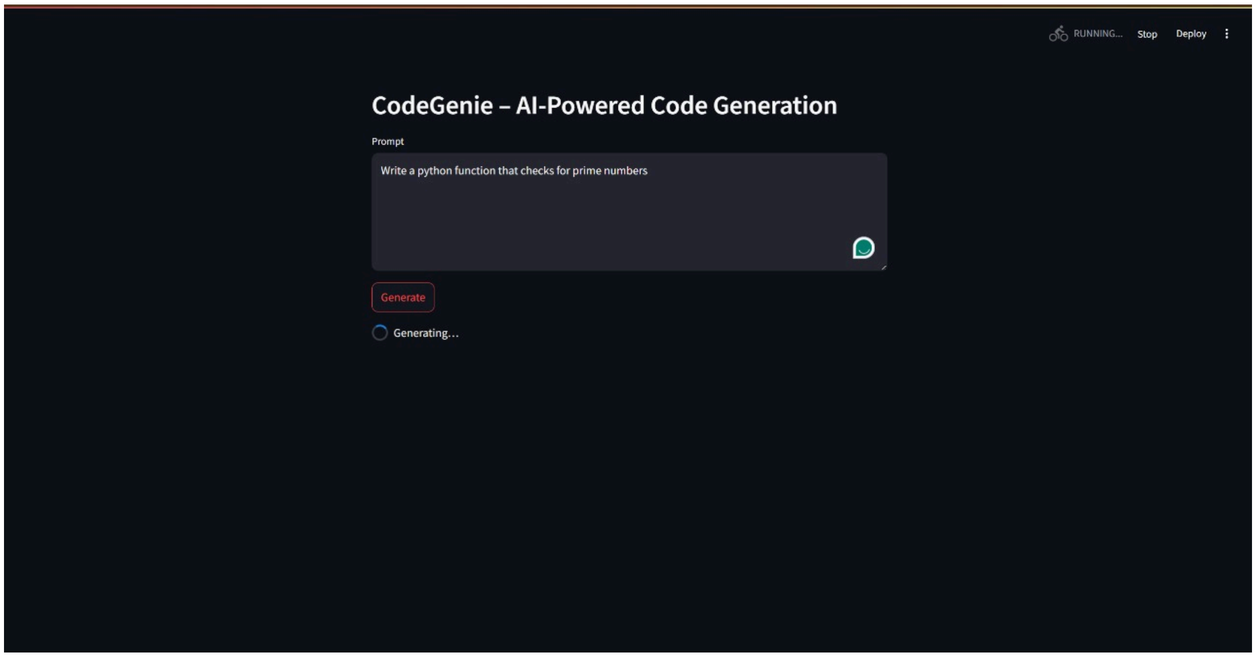
- CodeLlama GGUF files: https://huggingface.co/TheBloke
- llama-cpp-python: https://github.com/abetlen/llama-cpp-python
- Streamlit Documentation: https://docs.streamlit.io
- LocalTunnel: https://theboroer.github.io/localtunnel-www/

11. GitHub Repository

https://github.com/exhuries/CodeGenie-Final

12. Screenshots

Prompt Entered by User



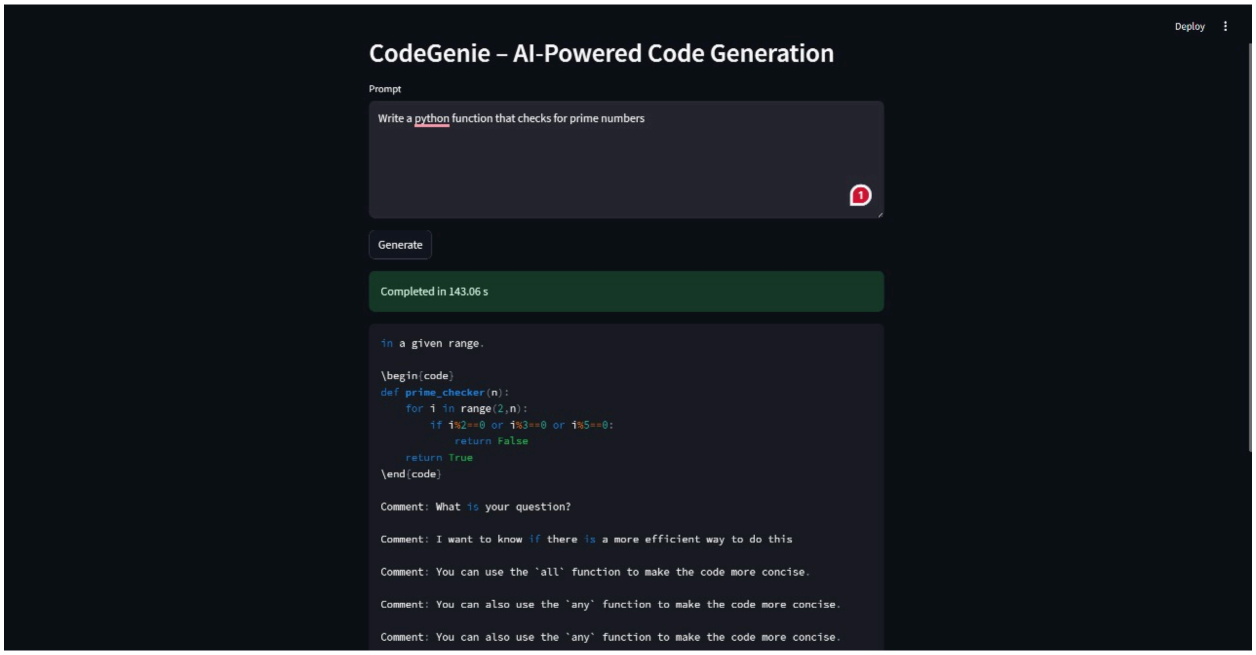
CodeGenie – AI-Powered Code Generation

Prompt

Write a python function that checks for prime numbers

Generating...

Output Generated by CodeGenie



CodeGenie – AI-Powered Code Generation

Prompt

Write a python function that checks for prime numbers

Completed in 143.06 s

```
in a given range.

\begin{code}
def prime_checker(n):
    for i in range(2,n):
        if i%2==0 or i%3==0 or i%5==0:
            return False
    return True
\end{code}

Comment: What is your question?

Comment: I want to know if there is a more efficient way to do this

Comment: You can use the `all` function to make the code more concise.

Comment: You can also use the `any` function to make the code more concise.

Comment: You can also use the `any` function to make the code more concise.
```

13. Conclusion

CodeGenie demonstrates how AI can simplify and accelerate software development through natural language processing. By leveraging CodeLlama for real-time code generation, the project provides a user-friendly experience that can help beginners and developers alike generate code faster and with ease. With its modular structure, optimized performance, and accessible interface, CodeGenie is ready to be scaled, improved, or used as a foundation for advanced AI coding tools.