**PROJECT DOCUMENTATION**

**Project Title: CodeGenie: AI-Powered Code Generation using CodeLlama**

**Overview**

CodeGenie is an intelligent code-generation tool that allows users to enter natural language prompts and receive fully functional, AI-generated code in return. It is powered by CodeLlama, a specialized Large Language Model designed for programming tasks. The project is designed to assist both developers and learners by automating coding tasks and offering high-quality code suggestions.

**Objectives**

Build a web-based application that takes user prompts as input and generates code output.

Integrate Meta's CodeLlama model to perform the actual code generation.

Use Streamlit to provide a simple and interactive user interface.

Enable public access to the tool using Localtunnel.

**Prerequisites**

Before using or extending this project, you should be familiar with:

* Large Language Models (LLMs) and prompt engineering - LLM Prompting Guide
* CodeLlama model and how it handles code generation - Meta's CodeLlama
* Streamlit for building Python-based web interfaces - Streamlit Tutorial

**Technologies Used**

* Transformers - Load and use CodeLlama model
* Torch - Model inference with GPU support
* Streamlit - Web application frontend
* Localtunnel - Publicly expose the local server

**Project Modules**

1. CodeModel.py

Loads the CodeLlama-7b-hf model and tokenizer.

Defines generate\_code(prompt) to generate code using a text-generation pipeline.

Returns the generated code and time taken.

2. app.py

Collects user input via a Streamlit interface.

Submits the input to the backend model.

Displays code output, processing time, and example use cases in a sidebar.

**Deployment**

* Local Execution

!streamlit run app.py --server.address=localhost &>/content/logs.txt &

* Public Deployment via Localtunnel

!npx localtunnel --port 8501

This exposes your app to a public URL like https://something.loca.lt, making it accessible to others even if you're running it on a local/Colab environment.

**Use Case Scenarios**

1. Automated Code Writing

Developers can describe a function in plain English and receive complete Python code, including comments and necessary libraries.

2. Educational Tool

Students input assignments or coding questions and receive structured, readable solutions with comments. Great for understanding coding logic and syntax.

3. Rapid Prototyping

Freelancers or startup devs can prototype ideas by quickly generating template code for APIs, data processing, or basic UI logic.

**Limitations & Future Improvements**

Currently dependent on local tunnel, which may be unstable for long use.

Code output is not syntax-verified automatically — future versions can include a linter.

Could add language selection, theme customization, or download option for generated code.

**Author**

Sakhi Telang – telangsakhi.sanjay2023@vitstudent.ac.in

Solo Developer