Project Title:

CodeGenie:AI-Powered Code Generation Using CodeLlama

Team Name:

Red Sea

Team Members:

- Avinash Yadav Pasham
- Shriyans Gouti
- Sai Naga Kireeti
- Ujwal Kumar

Phase-1: Brainstorming & Ideation

Objective

Develop an AI-powered **code generation tool** that helps developers generate, refine, and execute code across multiple programming languages using **CodeLlama**.

Key Points

Problem Statement

- Developers often struggle with **writing boilerplate code, debugging errors, and optimizing algorithms**.
- Existing AI code generators lack **real-time execution, multi-language support, and an interactive UI**.
- Beginners and professionals need **an efficient tool to generate, test, and learn code faster**.

Proposed Solution

An **AI-driven web app** built with **Streamlit** that:
✅ Generates **high-quality code** based on user prompts using **CodeLlama**.
✅ Supports multiple programming languages (**Python, Java, JavaScript, C++**).
✅ Provides **real-time execution** (for supported languages like Python & JavaScript).
✅ Includes **a well-designed UI with easy-to-use feature.**

Target Users

🧑‍💻 Developers & Programmers – Need quick **code snippets & debugging help**.
🎓 Students & Beginners – Want to **learn coding faster**.
🚀 Hackathon Teams – Need **rapid code prototyping**.

Expected Outcome

⚡ A **functional AI-powered code generator** that helps users create, refine, and execute code efficiently.

Phase-2: Requirement Analysis

Technical Requirements

- **Frontend**: Streamlit (for interactive UI)
- **Backend**: Hugging Face API (CodeLlama Model)
- **Programming Language**: Python
- **Execution Support**: Local execution for Python, JavaScript via sandboxing

Functional Requirements

✅ Accepts user **prompts** to generate relevant code snippets.
✅ Allows **real-time code execution** for Python & JavaScript.
✅ Offers **download & copy options** for easy usage.
✅ Supports **syntax highlighting & AI-powered debugging**.

Challenges & Constraints

🔴 API rate limits & response time optimization.
□ Handling multiple languages efficiently.
□ Ensuring security while allowing **code execution**.

Phase-3: Project Design

System Architecture

1 **User** enters a programming query or task.
2 **CodeLlama API** processes the input and generates a code snippet.
3 The **Streamlit frontend** displays the generated code.
4 **Optional Execution**: Python & JavaScript code can be tested in real-time.
5 Users can **copy, edit, or download** the generated code.

Phase-4: Project Planning (Agile Methodology)

Sprint Planning

Sprint	Task	Priority	Duration	Assigned to	Outcome
Sprint 1	**Set up API & Frontend UI**	● High	6 hrs	Sai Naga Kireeti	API connection & basic UI working
Sprint 2	**Implement Code Generation**	● High	4 hrs	Shriyans	AI generates relevant code snippets
Sprint 3	**Add Multi-Language Support**	□ Medium	5 hrs	Avinash	Code generation supports Python, JS, Java, C++
Sprint 4	**Add Execution & Debugging Features**	● High	6 hrs	Ujwal Kumar	Users can run code inside the app
Sprint 5	**UI Enhancements & Testing**	□ Medium	3 hrs	Sai Naga kireeti& Ujwal	Responsive UI & better user experience
Sprint 6	**Final Deployment & Demo**	□ Low	2 hrs	Avinash & Shriyans	Deploy on Streamlit & GitHub

Sprint Planning with Priorities for CodeGenie

Sprint 1 – Setup & Integration (Day 1)

● **High Priority –** Set up the **Streamlit environment &** install dependencies.
● **High Priority –** Integrate **Hugging Face API using Mistral-7B-Instruct-v0.3 model.**
□ **Medium Priority –** Build a basic UI with input fields and a "Generate Code" button.
🕒 **Duration: 6 hrs | Assigned to: Sai Naga Kireeti | Outcome: API connection & basic UI working**

Sprint 2 – Implement Code Generation (Day 1)

● **High Priority –** Implement code generation functionality using API requests.
● **High Priority –** Debug API responses and handle errors (e.g., invalid responses, timeouts).
🕒 **Duration: 4 hrs | Assigned to: Shriyans | Outcome: AI generates relevant code snippets**

Sprint 3 – Multi-Language Support (Day 1)

□ **Medium Priority** – Add support for multiple programming languages (Python, JavaScript, Java, C++).
🕒 **Duration: 5 hrs | Assigned to: Avinash | Outcome: Code generation supports Python, JS, Java, C++**

Sprint 4 – Execution & Debugging Features (Day 1)

● **High Priority** – Implement code execution functionality for Python & JavaScript.
● **High Priority –** Improve debugging features for better usability.
🕒 **Duration: 6 hrs | Assigned to: Ujwal Kumar | Outcome: Users can run code inside the app**

Sprint 5 – UI Enhancements & Testing (Day 2)

▯ **Medium Priority –** Test API responses and optimize generated code formatting.
▯ **Medium Priority** – Improve copy-to-clipboard functionality with better UI feedback.
🕒 **Duration: 3 hrs | Assigned to: Sai Naga Kireeti & Ujwal | Outcome: Responsive UI & better user experience**

Sprint 6 – Final Deployment & Demo (Day 2)

▯ **Low Priority –** Optimize performance and API response time.
▯ **Low Priority –** Deploy the project on Streamlit Cloud & GitHub submission.
🕒 **Duration: 2 hrs | Assigned to: Avinash & Shriyans | Outcome: Deploy on Streamlit & GitHub**

Phase-5: Project Development

Technology Stack

- **Frontend**: Streamlit (Python-based UI framework)
- **Backend**: Hugging Face API (CodeLlama Model)
- **Programming Language**: Python
- **Code Execution**: Local execution for Python & JavaScript

Development Process

✅ **API integration**: Connect CodeLlama to generate responses.
✅ **UI development**: Create input fields, buttons, and result display.
✅ **Multi-language support**: Implement language selection.
✅ **Execution feature**: Allow users to test Python & JavaScript code.

Challenges & Fixes

🛠 **Slow API Response** → Implement caching for repeated queries.
🔐 **Security Issues** → Restrict execution to safe sandboxed environments.
📉 **Unoptimized Code Generation** → Fine-tune prompts & output filtering.

Phase-6: Functional & Performance Testing

Test Case ID	Category	Scenario	Expected Outcome	Status
TC-001	Functional	User enters "Generate a Python function"	Function is generated	✅ Passed
TC-002	Functional	User requests JavaScript code	JavaScript code is generated	✅ Passed
TC-003	Performance	API response time under 1s	AI should return results quickly	⚠ Needs Optimization
TC-004	Security	User tries to run malicious code	Execution is blocked	✅ Secured
TC-005	UI Testing	Mobile & Desktop responsiveness	Works across all devices	❌ Needs Fixing
TC-006	Deployment	Hosted on Streamlit & GitHub	App is accessible	🚀 Deployed

---

Final Submission Requirements

📄 Project Report
🎥 Demo Video (3-5 Minutes)
🔗 GitHub Repository
📊 Presentation Slides
