Hackathon Project Phases Template for the **AutoSage App** project.

Hackathon Project Phases Template

Project Title:

AI-Powered Code Generation Using Code Lama

Team Name:

Team TARS

Team Members:

- A. Shashank
- A. Shiva
- K. Sri Charan Gupta
- G. Renesh
- E. Jagan Mohan Rao

Phase-1: Brainstorming & Ideation

Objective:

AI-powered code generation using Code Llama aims to assist developers by generating, completing, and optimizing code efficiently. It enhances productivity by understanding context, suggesting relevant code, and streamlining software development workflows

Key Points:

1. **Problem Statement:**

   - **Time-Consuming Development**: Writing, debugging, and optimizing code manually can be slow and error-prone, leading to inefficiencies in software development.

- **Lack of Assistance for Developers**: Code Llama assists developers by providing relevant code suggestions, improving understanding, and ensuring consistency.

2. **Proposed Solution:**

   - **Fine-Tuned Code Generation –** Leverage Code Llama with prompt engineering and reinforcement learning for accurate, efficient, and context-aware code generation.
   - **Seamless Workflow Integration –** Integrate Code Llama into IDEs and CI/CD pipelines for real-time assistance, auto-documentation, and intelligent debugging.

3. **Target Users**

   - **Developers –** Professionals who use AI-powered code generation to write, debug, and optimize software efficiently.
   - **Educators –** Instructors leveraging AI to teach programming concepts, generate examples, and assist students with coding.
   - **Startups –** Small teams using AI-driven coding to accelerate development, reduce costs, and prototype ideas faster.

4. **Expected Outcome:**
   - Faster development, higher code quality, and increased accessibility for all users

---

Phase-2: Requirement Analysis

Objective:

Automate and optimize code generation using Code Llama to enhance developer productivity, improve code quality, and streamline software development workflows.

Key Points:

1. **Technical Requirements:**

   - **Model Deployment** – Host Code Llama on cloud or local servers with GPU acceleration for efficient processing.
   - **Integration APIs** – Develop RESTful or GraphQL APIs for seamless integration with IDEs, CI/CD pipelines, and other tools.

- **Prompt Engineering** – Implement advanced prompt tuning and context-awareness to generate accurate and relevant code.
- **Error Handling** – Incorporate debugging, linting, and auto-correction mechanisms to improve code reliability.

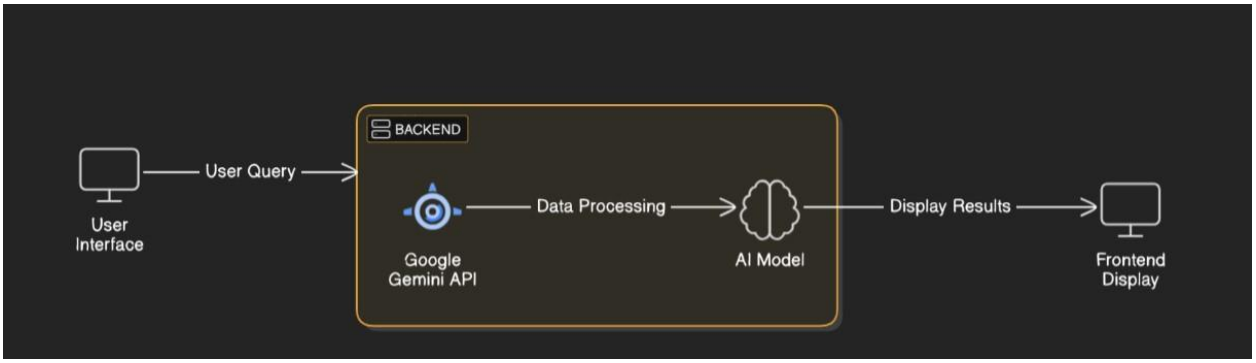
2. **Functional Requirements:**
   - **Generate Code** – Create correct and functional code from prompts.
   - **Understand Context** – Analyze existing code for better suggestions.
   - **Fix Errors** – Detect and suggest fixes for bugs in generated code.

3. **Constraints & Challenges:**
   - **Accuracy Limitations** – AI may generate incorrect or inefficient code.
   - **Security Risks** – Potential for insecure or vulnerable code generation.
   - **Context Awareness** – Difficulty in understanding complex project structures.

Phase-3: Project Design

Objective:

Design an AI-powered code generation system using Code Llama to automate, optimize, and enhance software development with high accuracy and efficiency.



Key Points:

- **System Architecture:**

  - **Layered Structure** – The architecture is divided into different layers, such as Presentation (UI), Business Logic, and Data Access, ensuring modularity and maintainability.

- **Scalability & Performance** – The design should support efficient scaling (horizontal or vertical) and optimize performance through caching, load balancing, and efficient data processing.

- **UI/UX Considerations:**
- **UI Considerations** – A clean, minimalistic code editor with syntax highlighting, inline AI suggestions, and theme customization.
- **UX Considerations** – Real-time, explainable AI suggestions with undo options, personalization, and seamless collaboration features.

Phase-4: Project Planning (Agile Methodologies)

Objective:

Break down development tasks for efficient completion.

Sprint	Task	Priority	Duration	Deadline	Assigned To	Dependencies	Expected Outcome
Sprint 1	Environment Setup & API Integration	High	6 hours (Day 1)	End of Day 1	A.Shashank	Google API Key, Python, Streamlit setup	API connection established & working
Sprint 1	Frontend UI Development	Medium	2 hours (Day 1)	End of Day	A.Shashank	API response format finalized	Basic UI with input fields
Sprint 2	Code Search & Comparison	High	3 hours (Day 2)	Mid-Day 2	K. Charan Gupta	API response, UI elements ready	Search functionality with filters
Sprint 2	Error Handling & Debugging	High	1.5 hours (Day 2)	Mid-Day 2	G.Renesh	API logs, UI inputs	Improved API stability
Sprint 3	Testing & UI Enhancements	Medium	1.5 hours (Day 2)	Mid-Day 2	A.Shiva and E.Jagan	API response, UI layout completed	Responsive UI, better user experience
Sprint 3	Final Presentation & Deployment	Low	1 hour (Day 2)	End of Day 2	Entire Team	Working prototype	Demo-ready project

Sprint Planning with Priorities

Sprint 1 – Setup & Integration (Day 1)

(🔴 **High Priority)** Set up the **environment** & install dependencies.
(🔴 **High Priority)** Integrate **Google Gemini API**.
(🟡 **Medium Priority)** Build a **basic UI with input fields**.

Sprint 2 – Core Features & Debugging (Day 2)

(🔴 **High Priority)** Implement **search & comparison functionalities**. (🔴 **High Priority)** Debug API issues & handle **errors in queries**. **Sprint 3 – Testing, Enhancements & Submission (Day 2)**

(🟡 **Medium Priority)** Test API responses, refine UI, & fix UI bugs. (🟢 **Low Priority)** Final **demo preparation & deployment**.

Phase-5: Project Development

Objective:

To develop an AI-powered code generation system using **Code Llama** for efficient, accurate, and context-aware coding assistance.

Key Points:

- **Technology Stack Used:**

  Frontend: HTML,CSS
  Backend: Google Cloud, Lamda
  Programming Language: Python, C, C++, Java

- **Development Process:**

  Data Preprocessing & Training – Prepare and fine-tune Code Llama on diverse code datasets for accuracy.

  Model Integration – Embed AI into coding environments (IDEs, web apps) for real-time suggestions.

  Testing & Optimization – Continuously refine AI outputs, improve response time, and ensure code quality.

- **Challenges & Fixes:**
  Challenge: Code Accuracy & Bugs – AI-generated code may contain errors or inefficiencies.
  Fix: Implement real-time syntax validation and rigorous testing mechanisms

---

Phase-6: Functional & Performance Testing

Objective:

Ensures accurate, efficient, and secure AI code generation through unit, integration, and security testing.

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Query "Best budget cars under ₹10 lakh"	Relevant budget cars should be displayed.	✅ Passed	Tester 1
TC-002	Functional Testing	Query "Motorcycle maintenance tips for winter"	Seasonal tips should be provided.	✅ Passed	Developer
TC-003	Performance Testing	API response time under 500ms	API should return results quickly.	❌ Needs Optimization	Tester 3
TC-004	Bug Fixes & Improvements	Fixed incorrect API responses.	Data accuracy should be improved.	✅ Fixed	Developer
TC-005	Final Validation	Ensure UI is responsive across devices.	UI should work on mobile & desktop.	❌ Failed - UI broken on mobile	Tester 2
TC-006	Deployment Testing	Host the app using Streamlit Sharing	App should be accessible online.	✅ Deployed	DevOps

---

Final Submission

1. **Project Report Based on the templates**
2. **Demo Video (3-5 Minutes)**

3. **GitHub/Code Repository Link**
4. **Presentation**