Hackathon Project Phases Template

Project Title:

CodeGenie: AI-Powered Code Generation using CodeLlama

Team Name:

- Code Dynamos

Team Members

- Naraboina Jyothi
- Nalla Bhavitha

Phase-1: Brainstorming & Ideation

Objective:

Develop an **AI-powered code generation tool** using **CodeLlama** to help developers generate, explain, and debug code efficiently.

Key Points:

1. **Problem Statement:**
    - Developers spend a lot of time writing repetitive code and fixing errors.
    - Many beginners struggle to understand complex code and need explanations.
    - Existing tools generate code but lack proper debugging and structured learning support.
2. **Proposed Solution:**
    - An AI-powered application that generates optimized code snippets based on user descriptions.
    - Provides step-by-step explanations and debugging suggestions for better learning.
    - Supports multiple programming languages and integrates with IDEs and web applications.

4.Target Users:

- Junior programmers looking for guidance and code examples.
- Senior developers in need of quick prototypes or code optimization.
- Teams and companies aiming for consistent, high-quality code output.
- Data scientists and machine learning engineers working on model deployment or data pipelines.
- Entrepreneurs or startups requiring rapid development of MVPs with minimal resources.
- Software developers and engineers seeking faster code generation

3. **Expected Outcome:**
    - A functional AI-powered coding assistant that enhances development speed and accuracy.
    - Integration with IDEs and web platforms for real-time assistance.
    - A scalable and user-friendly solution for developers at all level

---

Phase-2: Requirement Analysis

Objective:

CodeGenie aims to streamline and accelerate software development by leveraging AI-powered code generation using CodeLlama.

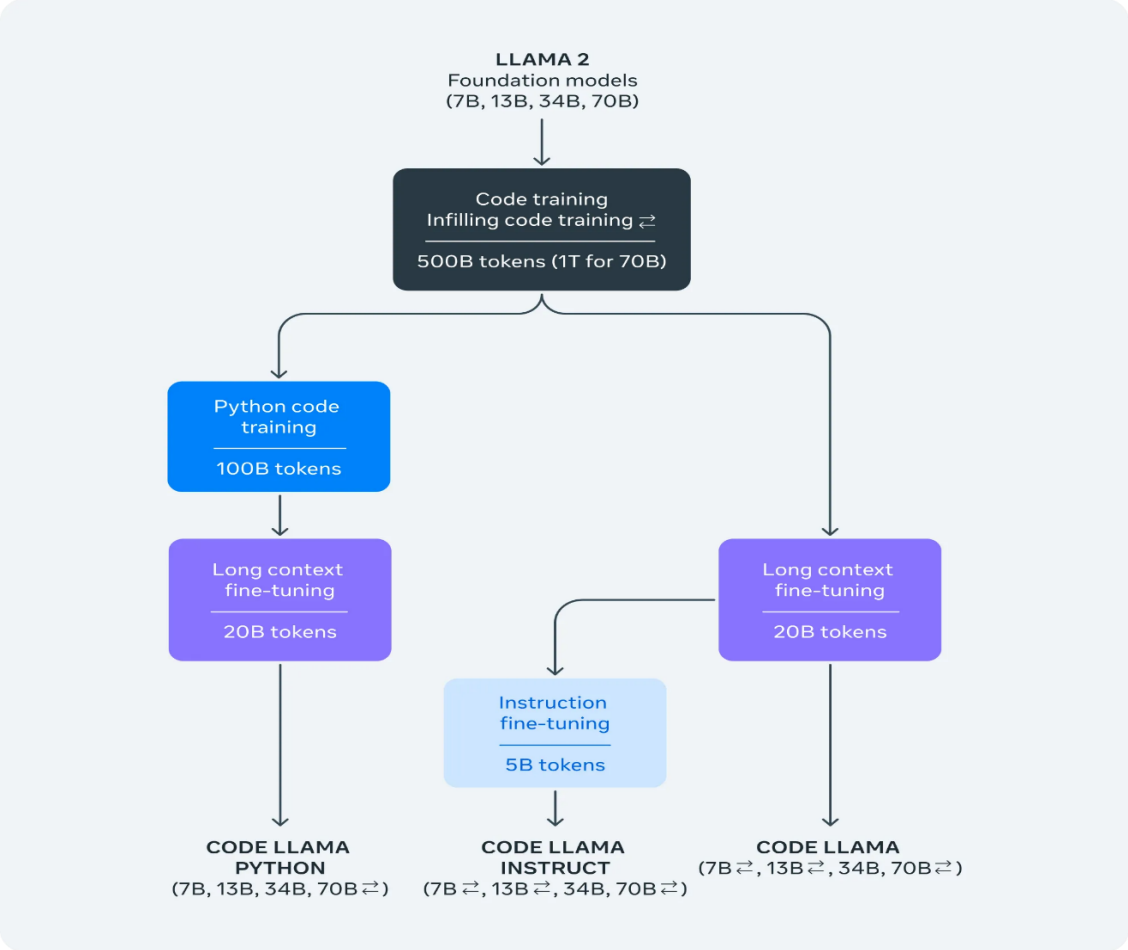
Key Points:

1. **Technical Requirements:**
    - **Programming Language:** Python, JavaScript
    - **Backend:** FastAPI (for AI processing), Node.js (for API handling)
    - **Frontend:** React.js with Tailwind CSS
    - **AI Model:** CodeLlama (via Hugging Face API)
    - **Database:** PostgreSQL / MongoDB (for storing user queries & generated code)
2. **Functional Requirements:**
    - Generate accurate, well-structured code from natural language prompts.
    - Provide step-by-step explanations and debugging support for generated code.
    - Support multiple programming languages (Python, C++, Java, etc.).
    - Integrate with IDEs (VS Code, JetBrains) for seamless usage.
    - Offer an API for external integrations in developer workflows.
3. **Constraints & Challenges:**
    - Ensuring high accuracy in code generation and explanations.
    - Managing API rate limits and response time for real-time interactions.
    - Providing a smooth and interactive UI with React.js.
    - Handling multi-language support efficiently.

Phase-3: Project Design

Objective:

Develop the architecture and user flow of the application.



Key Points:

1. **System Architecture:**
    - User enters a code-related query via the UI.
    - Query is processed using the CodeLlama API.
    - AI model generates and refines the required code.
    - The frontend displays the generated code with explanations and optimizations.

2. **User Flow:**
    - **Step 1:** User enters a prompt (e.g., *"Generate a Python function for sorting an array"*).
    - **Step 2:** The backend calls the CodeLlama API to process the request.
    - **Step 3:** The AI model generates the code and returns it to the frontend.
    - **Step 4:** The app displays the generated code, allowing the user to refine or copy it.
3. **UI/UX Considerations:**
    - Minimalist, developer-friendly interface for smooth interaction.
    - Options to select programming languages (Python, C++, Java, etc.).
    - Syntax highlighting & auto-formatting for readability.
    - Dark & light mode for better user experience.

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	Jyothi	CodeLlama API Key, FastAPISetup	API connection established & working
Sprint 1	Frontend UI Development	🟡 Medium	2 hours (Day 1)	End of Day 1	Bhavitha	API response format finalized	Basic UI with input fields
Sprint 2	Code Generation & Explaination	🔴 High	3 hours (Day 2)	Mid-Day 2	Jyothi,Bhavitha	API response, UI elements ready	Functional code generation module
Sprint 2	Error Handling & Debugging	🔴 High	1.5 hours (Day 2)	Mid-Day 2	Jyothi	API logs, UI inputs	Improved API stability
Sprint 3	Testing & UI Enhancements	🟡 Medium	1.5 hours (Day 2)	Mid-Day 2	Bhavitha	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 CodeLlama API.
🟡 **Medium Priority** – Build a basic UI with input fields.

Sprint 2 – Core Features & Debugging (Day 2)

🔴 **High Priority** – Implement code generation & explanation module.
🔴 **High Priority** – Debug API issues & handle errors in generated code.

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:

Implement core features of **CodeGenie: AI-Powered Code Generation using CodeLlama**.

Key Points:

1. **Technology Stack Used:**
    - **Frontend:** Streamlit
    - **Backend:** CodeLlama API
    - **Programming Language:** Python
2. **Development Process:**
    - Implement API key authentication and integrate CodeLlama API.
    - Develop code generation, explanation, and debugging logic.
    - Optimize query processing for efficiency and accuracy.
3. **Challenges & Fixes:**
    - **Challenge:** Slow response time for complex queries.
      Fix: Implement caching for frequently used code snippets.
    - **Challenge:** Handling ambiguous or incomplete user inputs.
      Fix: Use prompt optimization and context-aware suggestions.
    - **Challenge:** Ensuring code quality and correctness.
      Fix: Implement automated syntax validation before displaying output.

Phase-6: Functional & Performance Testing

Objective:

Ensure that **CodeGenie** functions correctly, performs efficiently, and meets quality standards.

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Generate a Python function for sorting a list.	Correct and optimized code snippet is returned.	✅ Passed	Jyothi
TC-002	Functional Testing	Request explanation for a complex algorithm.	Clear and concise explanation is provided.	✅ Passed	Bhavitha
TC-003	Performance Testing	API response time under 500ms	Results should return results quickly.	⚠ Needs Optimization	Jyothi
TC-004	Bug Fixes & Improvements	Fix incorrect code suggestions for C++.	Code accuracy should improve.	✅ Fixed	Developer
TC-005	Final Validation	Ensure UI works on different screen sizes.	UI should be responsive.	❌ Failed - UI issues	Bhavitha
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**