**Hackathon Project Phases Template** for the Ai-powered code Generatation using CodeLlama.

# Hackathon Project Phases Template

**Project Title:**

AI-Powered code Generartion using Code llama

**Team Name:**

Skill Titans

**Team Members:**

* Member 1: Shruthi siripuram
* Member 2: Boga Sai
* Member 3: Geedi Navitha
* Member 4: Lavanya Tamarapalli
* Member5: Chavva Vishwateja

## Phase-1: Brainstorming & Ideation

When brainstorming and ideating AI generation using **Code Llama**, the process can involve several steps, where we identify potential use cases, applications, and how to leverage the power of Llama's capabilities for creative, productive, and technical advancements. Here are some ideas for brainstorming:

**1. Automating Code Generation**

* **Use Case**: Code Llama could be used to generate code for specific tasks such as building web apps, APIs, or automating simple tasks like data processing.
* **Example**: You could prompt Code Llama to create the code for a RESTful API with CRUD operations in Python (Flask or FastAPI) or generate frontend code for a React application.

**2. Code Refactoring**

* **Use Case**: Code Llama can help refactor existing code for better readability, efficiency, and scalability.
* **Example**: Prompt it to optimize a large block of code by breaking it down into smaller, modular functions or improving its performance by suggesting alternative algorithms.

**3. Language Translation**

* **Use Case**: Code Llama can assist in translating code from one programming language to another.
* **Example**: You could use Llama to convert Python code into JavaScript or C++, enabling cross-platform compatibility.

**4. Code Explanation & Documentation**

* **Use Case**: Code Llama can generate detailed explanations or docstrings for code.
* **Example**: Automatically generate human-readable comments or documentation for complex code sections, helping both developers and non-developers understand the code.

**5. Code Testing & Debugging**

* **Use Case**: Create unit tests and identify potential bugs in code.
* **Example**: After writing a function or method, prompt Code Llama to generate test cases using frameworks like PyTest or Mocha for testing, and ask it to identify edge cases.

## Phase-2: Requirement Analysis

**Objective**: Requirement analysis is the process of identifying and documenting the requirements for a software system. For AI generation using Code Llama, a thorough requirement analysis will help in identifying what needs to be accomplished, the specific challenges to address, and how to structure the development process. Below is a breakdown of the key steps in the requirement analysis for an AI generation system based on Code Llama.

**1. Project Objective**

The first step in requirement analysis is to clearly define the project objectives. This involves understanding what the goal of using Code Llama for AI generation is and the value it will provide.

* Objective: Leverage Code Llama to automate software development tasks, assist in code generation, refactoring, testing, and optimization for improved productivity, quality, and efficiency.
* Value Proposition: The AI will help reduce manual effort in code writing, ensure consistency in coding styles, improve code quality through testing and refactoring, and assist with learning and code documentation.

**2. Stakeholders**

Identify and analyze the stakeholders involved in the project. These could be:

* End Users: Developers, data scientists, and engineers who will use the generated code.
* Project Managers: Those who oversee the development process and ensure that the solution aligns with business goals.
* QA Engineers: Individuals responsible for testing the quality and functionality of the generated code.
* Clients/Business Analysts: Define the requirements from the business side, including features and outcomes.
* Technical Architects: Individuals who focus on designing the high-level structure of the system.

## Phase-3: Project Design

**Objective**: The project design phase involves translating the requirements into a comprehensive architecture that details how the system will be built. This design serves as a blueprint for the development and deployment of the AI-powered code generation system using Code Llama

**1. System Architecture**

The system architecture defines the high-level structure of the application and how its components will interact with each other. It can be broken down into the following core components:

**A. Core Components**

**AI Model (Code Llama)**

* + **Role**: The AI model (such as Code Llama) will serve as the heart of the system, responsible for generating code, refactoring code, testing, debugging, and providing recommendations

* + **Serving Method**: The model will be exposed through a REST API or gRPC for interaction with other system components.

## Phase-4: Project Planning (Agile Methodologies)

**Objective:**

**Steps to Create an AI-Powered Agile Project Planning Generator:**

1. **Define Objectives & Scope of the Generator:** Determine what aspects of Agile planning the AI will automate. Common areas include:
   * Backlog creation and prioritization
   * Sprint planning (task assignment and estimation)
   * User story generation based on high-level features
   * Sprint retrospectives (based on historical sprint data)
   * Estimation of velocity and workload
2. **AI Model Integration:** The AI-powered generator would leverage a machine learning model like Code Llama, GPT, or other specialized AI to:
   * Parse project requirements.
   * Analyze historical data (if available) on past sprints.
   * Automatically generate user stories, tasks, and even suggest prioritization.
3. **Input Data:** The generator would need input from users, such as:
   * Product vision or project goals.
   * High-level user stories or features.
   * Constraints (e.g., team size, sprint duration).
   * Historical sprint data for better decision-making.
4. **Output:** The output would be a detailed Agile plan that includes:
   * A **Product Backlog** with prioritized user stories and tasks.
   * A **Sprint Backlog** with tasks assigned to specific team members.
   * Suggested **Sprint Planning** (based on the available capacity, sprint goals, and priority).
   * Estimated **Velocity** and task estimation.

**Example Code for AI-Powered Agile Project Planning Generator:**

Below is a simplified example of how the generator could work using Python, integrating an AI model (such as OpenAI Codex or Code Llama). This code would require an AI API integration for generating tasks or user stories based on input.

python

Copy

## Phase-5: Project Development

**Objective:**

. Project development refers to the process of planning, creating, testing, and delivering a project or product. It typically involves multiple phases, from initial concept to final delivery, and requires careful management, teamwork, and the application of various methodologies and tools..

**Key Phases of Project Development:**

1. **Project Initiation:**
   * **Goal Setting**: Clearly define the project's purpose and goals. This is often outlined in the **Project Charter**, which establishes the scope, objectives, stakeholders, and high-level requirements.
   * **Feasibility Study**: Assess whether the project is viable, considering factors like budget, resources, timeline, and technology.
   * **Team Formation**: Assign key roles, such as project managers, developers, designers, and testers.
2. **Project Planning:**
   * **Define Requirements**: Collaborate with stakeholders to gather detailed requirements for the project. This can include functionality, features, and non-functional requirements (e.g., performance, security).
   * **Create Project Plan**: Develop a roadmap, including deliverables, timelines, and milestones. In Agile methodologies, this could be a product roadmap and sprint plan.
   * **Risk Management**: Identify potential risks (e.g., budget overruns, technical challenges) and create a risk management plan.
   * **Resource Allocation**: Ensure the right resources (personnel, equipment, budget) are allocated to the project.
   * **Budgeting**: Estimate costs for different stages and ensure the project stays within budget.
3. **Design & Architecture:**
   * **System Design**: Create architectural diagrams, workflows, and wireframes. For software projects, this includes deciding on the tech stack (e.g., programming languages, frameworks).
   * **Prototyping**: Develop early prototypes or mockups of the product to visualize the end result and gain feedback from stakeholders.
   * **User Interface (UI)/User Experience (UX) Design**: If applicable, design how the product will look and how users will interact with it. This can involve wireframes, design systems, or user journeys.

## Phase-6: Functional & Performance Testing

**Objective:**

Ensure that the AutoSage App works as expected.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 | Tester 2 |
| 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 | Develop er |
| 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**