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

# **Hackathon Project Phases Template**

## **Project Title:**

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

## **Team Name:**

CodeGen

## **Team Members:**

* Kandagaddala Venkata Sai Geetesh
* Varshith Nomula
* Varshith Potnuru

## **Phase-1: Brainstorming & Ideation**

### **Objective:**

Develop an **AI-powered code generation tool** using **CodeLlama** to help developers generate, debug, and optimize code snippets effortlessly.

### **Key Points:**

1. **Problem Statement:**
   * Developers often struggle with writing efficient code, debugging issues, and following best practices.
   * Searching for accurate code snippets across multiple sources is time-consuming.
2. **Proposed Solution:**
   * **CodeGenie**: An AI-powered application using **CodeLlama** to generate, explain, and optimize code snippets across multiple languages.
   * The app will take **natural language prompts** and provide **complete, well-structured code**, including necessary imports and comments.
3. **Target Users:**
   * **Beginner programmers** needing quick code generation and learning resources.
   * **Software developers** looking for optimized solutions.
   * **Students & researchers** exploring new coding patterns.
   * **Tech teams** improving productivity through AI-assisted development.
4. **Expected Outcome:**
   * A functional AI-powered code generation tool that provides **accurate, efficient, and customizable code snippets** in real-time.
   * **Faster development cycles** by reducing time spent on writing repetitive code.
   * **Better code quality** through AI-driven best practices and optimizations.

## **Phase-2: Requirement Analysis**

### **Objective:**

Define the technical and functional requirements for **CodeGenie: AI-Powered Code Generation using CodeLlama**.

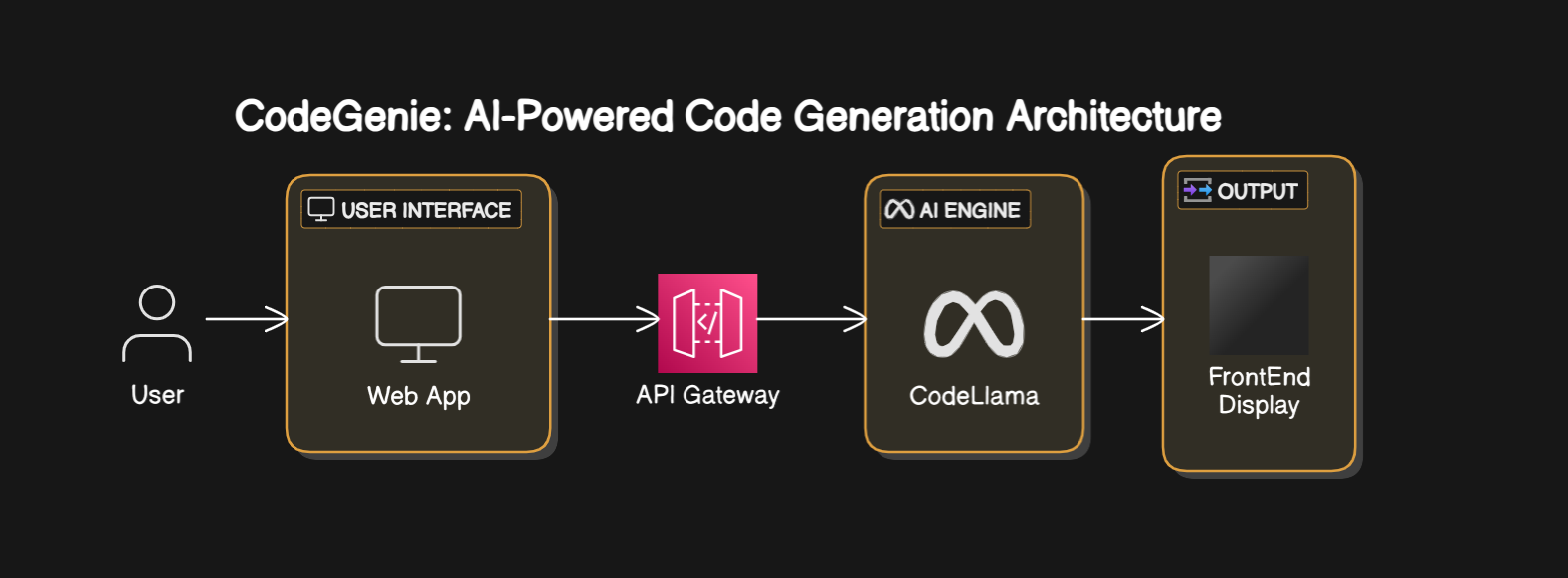
### **Key Points:**

1. **Technical Requirements:**
   * **Programming Language: typescript**
   * **Backend: CodeLlama Model (Optimized for code generation)**
   * **Frontend: typescript**
   * **Database: Not required initially (direct LLM-based code generation)**
2. **Functional Requirements:**
   * **Accept user input** for function descriptions, programming languages, and frameworks.
   * **Generate accurate and efficient code** based on user queries.
   * **Support multiple programming languages**, including Python, Java, C++, JavaScript, and C#.
   * **Provide AI-driven debugging and optimization suggestions** for generated code.
   * **Enable framework-specific code generation**, such as Flask, Django, and React.
   * **Display generated code in a scrollable output section** for better readability.
   * **Allow users to copy or save generated code** for future use.
3. **Constraints & Challenges:**
   * **Ensuring fast response times** while generating code using CodeLlama.
   * **Handling large language model resource requirements** on different hardware setups.
   * **Optimizing UI performance** for smooth interaction.
   * **Managing potential inaccuracies** in code generation and debugging suggestions.

## **Phase-3: Project Design**

### **Objective:**

Develop the architecture and user flow of the application.



### **Key Points:**

1. **System Architecture:**
   * User enters program-related query via UI.
   * Query is processed using Codellama.
   * AI model fetches and processes the data.
   * The frontend displays the generated code.
2. **User Flow:**
   * Step 1: User enters a query (e.g., "code for factorial of a number").
   * Step 2: The backend **calls the Codellama open source** to retrieve data.
   * Step 3: The app processes the data and **displays results** in an easy-to-read format.
3. **UI/UX Considerations:**
   * **Minimalist, user-friendly interface** for seamless navigation..
   * **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 | Varshith Nomula | Codellama setup | API connection established & working |
| Sprint 1 | Frontend UI Development | 🟡 Medium | 2 hours (Day 1) | End of Day 1 | K.V.Sai Geetesh | API response format finalized | Basic UI with input fields |
| Sprint 2 | Vehicle Search & Comparison | 🔴 High | 3 hours (Day 2) | Mid-Day 2 | Varshith Nomula | API response, UI elements ready | Search functionality with filters |
| Sprint 2 | Error Handling & Debugging | 🔴 High | 1.5 hours (Day 2) | Mid-Day 2 | K.V.Sai Geetesh | API logs, UI inputs | Improved API stability |
| Sprint 3 | Testing & UI Enhancements | 🟡 Medium | 1.5 hours (Day 2) | Mid-Day 2 | Varshith Potnuru | 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**.  
 **(🟡 Medium Priority)** Build a **basic UI with input fields**.

### **Sprint 2 – Core Features & Debugging (Day 2)**

**(🔴 High Priority)** Implement **search 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:**

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

### **Key Points:**

1. **Technology Stack Used:**
   * **Frontend:** typescript
   * **Backend:** typescript (Codellama open source model)
   * **Programming Language:** typescript
2. **Development Process:**
   * Implement **API key authentication** and **Codellama integration**.
   * Optimize **search queries for performance and relevance**.
3. **Challenges & Fixes:**
   * **Challenge:** Delayed API response times.  
      **Fix:** Implement **caching** to store frequently queried results.
   * **Challenge:** Limited API calls per minute.  
      **Fix:** Optimize queries to fetch **only necessary data**.

## **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 "write a program to find factorial of a number" | Relevant factorial of a number code should be displayed. | ✅ Passed | Varshith Nomula |
| TC-002 | Functional Testing | Query "write a program to implement quick sort algorithm" | Relevant quicksort code should be displayed. | ✅ Passed | K.V.Sai Geetesh |
| TC-003 | Performance Testing | API response time under 500ms | API should return results quickly. | ⚠ Needs Optimization | Varshith Potnuru |
| TC-004 | Bug Fixes & Improvements | Fixed incorrect API responses. | Data accuracy should be improved. | ✅ Fixed | K.V.Sai Geetesh |
| TC-005 | Final Validation | Ensure UI is responsive across devices. | UI should work on mobile & desktop. | ❌ Failed - UI broken on mobile | Varshith Potnuru |
| TC-006 | Deployment Testing | Host the application using typescript | App should be accessible online. | 🚀 Deployed | Varshith Nomula |

## **Final Submission**

1. **Project Report Based on the templates**
2. **Demo Video (3-5 Minutes)**
3. **GitHub/Code Repository Link**
4. **Presentation**