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

**Hackathon Project Phases Template**

**Project Title:**

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

**Team Name:**

GenieBytes

**Team Members:**

* K. Greeshma
* D. Varsha Reddy
* Ch. Varshitha

**Phase-1: Brainstorming & Ideation**

**Objective:**

CodeGenie: AI-Powered Code Generation using CodeLlama aims to enhance the software development process by utilizing CodeLlama, an advanced AI model. The primary objective is to assist developers by automatically generating high-quality code snippets, offering real-time code completion, and providing debugging support. By understanding natural language descriptions, CodeGenie translates them into functional code, speeding up development and reducing human error. It also aids in code optimization and refactoring, making it easier for developers to implement complex features. Overall, CodeGenie streamlines the coding workflow, boosting productivity and ensuring more efficient software development.

**Key Points:**

1. **Problem Statement:**
   * This innovative system simplifies the development process by providing accurate and efficient code snippets, comprehensive code explanations, and debugging support.
2. **Proposed Solution:**
   * **Natural Language Input**: Developers can describe the functionality they need in simple language, and CodeGenie converts those descriptions into code, reducing the need for manual coding.
   * **Code Snippet Generation**: The system provides code snippets, solutions to common programming problems, and code completions to help developers write and implement code faster.
   * **Error Detection and Debugging**: The AI model assists in identifying bugs, errors, and issues in the code, offering suggestions for fixes, and even automating the debugging process.
3. **Target Users:**
   * **Software Developers**: Developers across various experience levels (from beginners to advanced) can benefit from CodeGenie’s ability to assist in code generation, optimization, and debugging.
   * **Startups and Small Teams**: Small teams with limited resources can use CodeGenie to rapidly generate and deploy code, improving efficiency and reducing the need for extensive in-house expertise.
   * **Educators and Students**: Coding instructors and students can utilize CodeGenie to understand best practices, learn coding patterns, and generate code examples for learning purposes.
4. **Expected Outcome:**
   * CodeGenie allows developers to generate code effortlessly. For instance, a programmer can input a brief description of a desired function, and CodeGenie will provide the complete code snippet, including necessary libraries and comments. This feature enables developers to save time, reduce errors, and focus on more complex tasks by automating repetitive coding tasks.

**Phase-2: Requirement Analysis**

**Objective:**

Define the technical and functional requirements for the AI-Powered Code Generation

**Key Points:**

1. **Technical Requirements:**
   * Programming Language: **Python**
   * Backend API: **Flask**
   * Frontend : **Streamlit Web Framework**
   * Database: **Not required initially (API-based queries)**
2. **Functional Requirements:**
   * **Natural Language Input** : The system should allow users to input code requirements in natural language, such as asking the AI to "Generate a Python function to sort a list."
   * **Code Generation :**  The system must be able to generate code in various programming languages based on user input.
   * **Error Detection and Debugging:** The system must help users find and fix errors in their code.
   * **Collaboration and Sharing:** Users should be able to share generated code or projects with others.
3. **Constraints & Challenges:**

**Model Accuracy and Quality of Generated Code**:

**Challenge**: While CodeLlama can generate code based on natural language descriptions, the generated code may not always be optimal or contextually accurate. There is a risk of code that is syntactically correct but inefficient or prone to errors.

**Constraint**: The AI model’s ability to understand complex or ambiguous descriptions in natural language may lead to inaccurate or suboptimal code generation.

**Real-Time Performance and Scalability**:

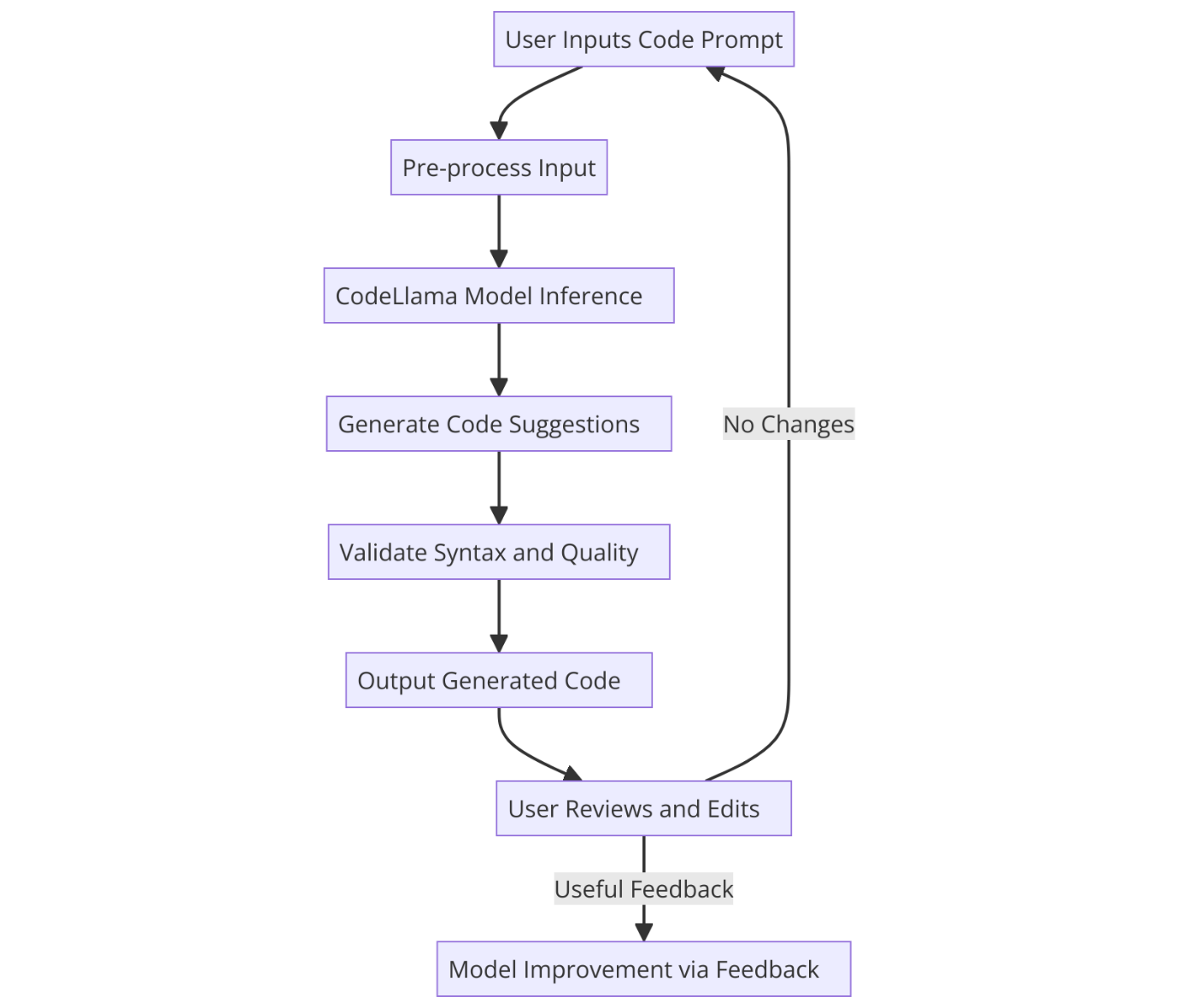
**Challenge**: Real-time code suggestions, error detection, and debugging require low-latency responses, which can be challenging with heavy AI models like CodeLlama, especially in a multi-user environment.

**Constraint**: Ensuring that the AI model responds quickly enough while handling potentially large amounts of concurrent requests could strain resources.

**Phase-3: Project Design**

**Objective:**

Develop the architecture and user flow of the application.



**Key Points:**

1. **System Architecture:**
   * **User submits a code request** via the frontend.
   * **Backend API** validates the input and forwards it to the AI inference layer.
   * **CodeLlama model** processes the request and generates code suggestions.
   * **Generated code is sent** back to the frontend for display.
   * **User reviews and modifies** the AI-generated code.
   * **Optional fine-tuning** based on user feedback.
2. **User Flow:**
   * Login/Register
   * Enter code prompt
   * AI generates code
   * User refines & modifies
   * Run & debug (if applicable)
   * Save or export code
   * Give feedback
   * Logout.
3. **UI/UX Considerations:** 
   * **Fast AI Response Times :** Show loading indicators while AI generates code.
   * **Syntax Highlighting** for multiple programming languages.
   * **Dark & light mode** for better user experience.
   * **Run Code Inline:** If applicable, provide a built-in console for quick execution .
   * **Keyboard Navigation:** Ensure full navigation without a mouse.
   * **Private Code Generation:** Ensure user code is not stored/shared without consent.
   * **Customizable Themes & Fonts** (Light/Dark mode, Font size).
   * **Error Highlighting & Linting** (Red for syntax errors, Yellow for warnings).
   * **Proper Spacing & Typography:** Use **monospace fonts (Fira Code, JetBrains Mono)** for better code readability.
   * **Collapsible Sections** for readability (e.g., function blocks).**Color Contrast:** Ensure colorblind-friendly UI (use WCAG guidelines).

**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 | Greeshma | Google API Key, Python, Streamlit setup | API connection established & working |
| Sprint 1 | Frontend UI Development | 🟡 Medium | 2 hours (Day 1) | End of Day 1 | Varsha | API response format finalized | Basic UI with input fields |
| Sprint 2 | Code Generation and optimization | 🔴 High | 3 hours (Day 2) | Mid-Day 2 | Varshitha | API response, UI elements ready | Search functionality with filters |
| Sprint 2 | Error Handling & Debugging | 🔴 High | 1.5 hours (Day 2) | Mid-Day 2 | Greeshma,  Varshitha | API logs, UI inputs | Improved API stability |
| Sprint 3 | Testing & UI Enhancements | 🟡 Medium | 1.5 hours (Day 2) | Mid-Day 2 | Varsha | 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:**

Implement core features of the AutoSage App.

**Key Points:**

1. **Technology Stack Used:**
   * **Frontend:** Streamlit
   * **Backend:** Flask
   * **Programming Language:** Python
2. **Development Process:**
   *  **Plan & Design** – Define features, choose tech stack (React, Node.js/FastAPI, CodeLlama), and design an intuitive UI/UX.
   * **Develop & Integrate** – Build the frontend, backend, and integrate CodeLlama for real-time AI-driven code generation.
   * **Test & Deploy** – Optimize AI performance, ensure security, and deploy using Docker, Kubernetes, and cloud infrastructure.
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 "Best budget cars under ₹10 lakh" | Relevant budget cars should be displayed. | ✅ Passed |  |
| TC-002 | Functional Testing | Query "Motorcycle maintenance tips for winter" | Seasonal tips should be provided. | ✅ Passed | anwashanwazr |
| 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 |