**Project Title:**   
**CodeGenie: AI-Powered Code Generation from Text Prompts**

**Team Name:**

Team Nexus

|  |  |
| --- | --- |
| **Team Members:** | |
| ● | Nanaveni Deekshith |
| ● | Annala Raghava |
| ● | Kokku Raj Kumar |
| ● | Sapavat Anji |
| ● | Gaddoju Vikas |

**Phase-1: Brainstorming & Ideation**

**Objective:**

Develop an AI-powered tool that generates code snippets or full programs from natural language text prompts, helping developers and non-programmers write code faster and more efficiently.

**Key Points:**

1. **Problem Statement:**

|  |  |
| --- | --- |
| ○  ○  ○ | Developers often spend time writing repetitive code or searching for syntax. Non-programmers struggle to write code for simple tasks due to lack of coding knowledge.  There is a need for a tool that can generate accurate and efficient code from plain English descriptions. |

2. **Proposed Solution:**

|  |  |
| --- | --- |
| ○ | An AI-powered application that generates code from text prompts using **Code** |
| **Llama** . | |
| ○ | The tool will support multiple programming languages (e.g., Python, |
| JavaScript, Java). | |
| ○ | It will provide error handling, and code optimization suggestions. |
| 3. **Target Users:** | |
| ○ | **Developers:** looking to speed up coding tasks. |
| ○ | **Students:** learning to code. |
| ○ | **Non-programmers:** who need to automate simple tasks. |
| 4. **Expected Outcome:** | |
| ○ | A functional **AI-powered code generation tool** that provides accurate and |
| efficient code snippets based on user prompts. | |

**Phase-2: Requirement Analysis**

**Objective:**

Define the technical and functional requirements for CodeGenie.

**Key Points:**

1. **Technical Requirements:**

○ Backend: **Code Llama API (or Hugging Face Transformers)**

○ Frontend: **Next.js**

○ Database: **Not required initially API-based queries)**

2. **Functional Requirements:**

○ Ability to **generate code snippets** from text prompts.

○ Support for **multiple programming languages**.

○ Display generated code with **syntax highlighting**.

○ Provide **error handling** and **code optimization suggestions**.

3. **Constraint & Challenges:**

○ Ensuring **real-time code generation** with minimal latency.

○ Handling **API rate limits** and optimizing API calls.

○ Providing a **smooth UI experience** for users.

**Phase-3: Project Design**   
**Objective:**   
Develop the architecture and user flow of the application.

**Key Points:**   
 1. **System Architecture:**

|  |  |
| --- | --- |
| ○ | User enters a text prompt via the UI. |
| ○ | The prompt is sent to the **OpenAI API** for processing. |
| ○  ○ | The AI model generates code based on the prompt.  The frontend displays the generated code with syntax highlighting.. |
| 2. **User Flow:** | |
| ○ | Step1: enters a text prompt (e.g., "Write a Python function to calculate factorial"). |
| ○ Step 2: The backend calls the \*OpenAI API\* to generate code.  ○ Step 3: The app displays the generated code in an easy-to- format.  3. **UI/UX Considerations:** | |
| ○ | **Minimalist, user-friendly interface** for seamless navigation. |
| ○ | **Syntax highlighting** for better readability**.** |
| ○ | **Dark & light mode** for better user experience. |

**Phase-4: Project Planning (Agile Methodologies) Objective:**   
Breakdown development tasks for efficient completion.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Task** | **Priority** | **Duration** | **Deadline** | **Assigned To** | **Dependencies** | **Expected Outcome** |
| Sprint 1 | Environment Setup & API Integration | 🔴High | 5 hours (Day 1) | End of Day 1 | Raghava | Code Llama, Next JS | |  |  | | --- | --- | |  | API connection established & working | |
| Sprint 1 | Frontend UI Development | 🔴 Medium | 3 hours (Day 1) | End of Day 1 | Raj Kumar | API response format finalized | Basic UI with input fields |
| Sprint 2 | Code Generation Functionality | 🔴High | 3 hours (Day 2) | Mid-Day 2 | Deekshith | API response, UI elements ready | Code generation from text  prompts |
| Sprint 2 | Error Handling & Debugging | 🔴High | 1.5 hours (Day 2) | Mid-Day 2 | Vikas | API logs, UI inputs | Improved API stability |
| Sprint 3 | Testing & UI  Enhancements | 🔴 Medium | 1.5 hours (Day 2) | Mid-Day 2 | Anji | 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(Day1)**   
**(**🔴**High Priority)**Setup the **environment** & install dependencies. **(**🔴**High Priority)** Integrate **Code Llama API**.

**(**🔴**Medium Priority)** Build a **basic UI with input fields**.

**Sprint 2 – Core Features & Debugging (Day2)**   
**(**🔴**High Priority)** Implement **code generation functionality** from text prompts   
 **(**🔴**High Priority)** Debug API issues & handle **errors in queries**.

**Sprint3–Testing, Enhancements & Submission (Day2) (**🔴**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 **CodeGenie** application.

**Key Points:**   
 1. **Technology Stack Used:**

|  |  |
| --- | --- |
| ○ | **Frontend:** Next JS |
| ○ | **Backend:** Next JS |
| 2. **Development Process:** | |
| ○ | Implement **API key authentication** and **Code Llama API integration**. |
| ○ | Develop **code generation logic** to process text prompts and generate code snippets. |
| ○ | Optimize **API calls for performance** and ensure minimal latency. |
| 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 **Code Genie Application** works as expected.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test**  **CaseID** | **Category** | **Test Scenario** | **Expected Outcome** | **Status** | **Tester** |
| TC-001 | Functional Testing | Query: "Write a Python function to calculate factorial" | Correct Python code snippet should be  generated.. | ✅Passed | Deeksh ith |
| TC-002 | Functional Testing | Query: "Create a  JavaScript function to reverse a string | Correct JavaScript code snippet should be  generated. | ✅Passed | Raj  Kumar |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TC-003 | Performance Testing | API response time under 500ms | API should return results quickly. | ⚠ Needs Optimization | Tester3 |
| TC-004 | Bug Fixes  &Improvement s | 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 | Tester2 |
| TC-006 | Deployment Testing | Host the app using Streamlit/react JS | App should be  accessible online. | 🔴Deployed | DevOps |

**FinalSubmission**

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