**Project Design Phase-II**

**Solution Requirements (Functional & Non-functional)**

|  |  |
| --- | --- |
| Date |  |
| Team ID | LTVIP2025TMID59401 |
| Project Name | SmartSDLC – AI-Enhanced Software Development Lifecycle |
| Maximum Marks | 4 Marks |

**Functional Requirements:**

The SmartSDLC system includes multiple intelligent modules that assist developers throughout the SDLC. Each module is designed to perform a specific function while ensuring ease of use and modularity.

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | Requirement Classification | Users upload a software requirement document in PDF format. The system reads and classifies content into relevant SDLC phases (like Requirements, Design, Testing, etc.) using AI-based classification models. |
| FR-2 | Chatbot Assistant | An AI chatbot allows users to interact and ask queries related to software development processes. It provides real-time, context-aware responses for different SDLC phases. |
| FR-3 | Code Generation | Users input a software requirement or description, and the system generates Python code snippets using LLMs trained for software development. |
| FR-4 | Bug Fixer | Allows users to input buggy or incomplete code. The system uses language models to identify bugs, suggest corrections, and return updated code. |
| FR-5 | Code Summarizer | This module accepts source code and generates human-readable explanations, useful for code reviews and documentation. |
| FR-6 | Test Case Generator | Users submit source code or a requirement. The system generates test cases automatically using unit testing standards. |

**Non-functional Requirements:**

Non-functional requirements ensure that the SmartSDLC system performs well, remains secure, and scales effectively. These requirements affect system architecture and deployment strategy.

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | The user interface should be intuitive and accessible, requiring minimal training. |
| NFR-2 | **Security** | All API calls and data transfers must be encrypted. User inputs must be validated and sanitized. |
| NFR-3 | **Reliability** | The application must function consistently with minimal downtime and must recover from failures gracefully. |
| NFR-4 | **Performance** | System operations such as code generation or classification should respond within 5 seconds. |
| NFR-5 | **Availability** | System should run on different environments like local machines, IBM Cloud, or Docker containers |
| NFR-6 | **Scalability** | The backend should be capable of handling multiple concurrent requests using horizontal scaling. |