**Full Stack Application with LLM Integration**

**Documentation**

# 1. Introduction

* **Project Title:** SmartSDLC – AI-enhanced Software Development Lifecycle
* **Team Members:**
  + 1. Yogesh – Developer
  + 2. Varun – Project Design
  + 3. Yeswanth - Project Planning
  + 4. Vinay – UI design
  + 5. Sowmya – UI design

# 2. Project Overview

* **Purpose:**

This project addresses the challenge faced by software developers in comprehending and maintaining large or unfamiliar codebases. The primary objective is to build an AI-powered assistant that leverages IBM WatsonX's Granite large language models to provide intelligent code summarization, logic explanation, and documentation support. By integrating this functionality into an interactive web-based platform, the system aims to enhance developer productivity, accelerate onboarding processes, and reduce the cognitive effort associated with code understanding and maintenance.

* **Features:**

1. Chat with AI about requirements and coding questions.
2. Generate User Stories or Test Cases.
3. Analyze code snippets and provide review results.
4. Summarize PR or commit messages.
5. Maintain context across interactions.

# 3. Architecture

* **Frontend:** Describe the frontend architecture using React.
* **Backend:** Outline the backend architecture using Node.js and Express.js.
* **Database:** Detail the database schema and interactions with MongoDB.

# 4. Setup Instructions

* **Prerequisites:** List software dependencies (e.g., Node.js, MongoDB).
* **Installation:** Step-by-step guide to clone, install dependencies, and set up the environment variables.

# 5. Folder Structure

* **Client:** Describe the structure of the React frontend.
* **Server:** Explain the organization of the Node.js backend.

# 6. Running the Application

• Provide commands to start the frontend and backend servers locally.

o **Frontend:** npm start in the client directory. o **Backend:** npm start in the server directory.

# 7. API Documentation

* Document all endpoints exposed by the backend.
* Include request methods, parameters, and example responses.

# 8. Authentication

* Explain how authentication and authorization are handled in the project.
* Include details about tokens, sessions, or any other methods used.

1. **User Interface** 
   * Provide screenshots or GIFs showcasing different UI features.
2. **Testing** 
   * Describe the testing strategy and tools used.
3. **Screenshots or Demo**

https://github.com/TYogesh12/MyGenproject

1. **Known Issues**

* The chatbot is limited to predefined SDLC topics and lacks memory/context retention across sessions.
* Current system uses temporary, in-memory data structures with no cloud or local DB for storing history or sessions.
* Classifier model may misclassify some requirement lines due to ambiguity in unstructured text

# 13. Future Enhancements

• Add full login/signup with hashed credentials, JWT-based sessions, password reset, and multi-user support.

* Store classified requirements, generated code, bug fixes, test cases, and user feedback in a database (e.g., PostgreSQL or IBM DB2).
* Enable requirement classification and code generation in languages other than English.
* Add full login/signup with hashed credentials, JWT-based sessions, password reset, and multi-user support.