**Project Title**

**Project Documentation**

1. **Introduction**

Project Title: SmartSDLC – AI-Enhanced Software Development Lifecycle

Team Members:

Member 1 : Vijiyalakshmi S

Member 2 : Varsha M

Member 3 : Varshitha S

Member 4 : Vishali B

**2.Project Overview**

* **Purpose**:

The purpose of SmartSDLC is to enhance the traditional Software Development Lifecycle by integrating Artificial Intelligence and automation. SmartSDLC provides intelligent support at every stage planning, requirement analysis, design, coding, testing, deployment, and maintenance making the process faster, more reliable, and cost-effective. By leveraging AI, real-time analytics, and predictive models, SmartSDLC assists software teams in decision-making, risk detection, effort estimation, and error reduction. Ultimately, SmartSDLC transforms software development into a smarter, adaptive, and more efficient lifecycle model.

**Features:**

1. **Resource Forecasting**
   * **Key Point:** Predictive analytics
   * **Functionality:** Estimates future resource requirements such as developer effort, testing time, infrastructure needs, and budget using historical project data and AI-driven insights.
2. **AI-Driven Recommendation Engine**
   * **Key Point:** Personalized project guidance
   * **Functionality:** Provides best practices, coding standards, and optimization tips tailored to team performance and project requirements.
3. **Developer Feedback Loop**
   * **Key Point:** Continuous improvement
   * **Functionality:** Collects and analyzes developer/tester feedback on tasks and tools to optimize workflows and reduce bottlenecks.
4. **KPI Forecasting**
   * **Key Point:** Strategic planning support
   * **Functionality:** Projects key performance indicators (e.g., delivery speed, defect density, resource utilization) to help managers track progress and plan ahead
5. **Anomaly Detection**
   * **Key Point:** Early risk detection
   * **Functionality:** Identifies unusual patterns such as sudden code errors, performance degradation, or cost overruns to flag potential risks early.
6. **Multimodal Input Support**
   * **Key Point:** Flexible document handling
   * **Functionality:** Accepts project artifacts like requirement documents (text/PDFs), code repositories, and test reports (CSVs/Logs) for automated analysis and forecasting.
7. **Streamlit or Gradio UI**
   * **Key Point:** User-friendly interface
   * **Functionality:** Provides an intuitive dashboard for developers, testers, and managers to interact with SmartSDLC features, visualize KPIs, and receive AI-driven insights.

## Architecture

**Frontend (Streamlit):**

* Built with Streamlit, offering an interactive web UI with multiple pages including dashboards, file uploads, chat interface, feedback forms, and report viewers.
* Navigation is handled through a sidebar using the streamlit-option-menu library.
* Each page is modularized for scalability.

**Backend (FastAPI):**

* Powers the backend REST framework with endpoints for requirement analysis, bug prediction, KPI forecasting, and reporting.
* Optimized for asynchronous performance with **Swagger UI** support.

**LLM Integration (IBM Watsonx Granite):**

* Granite LLM models are used for natural language understanding and generation.
* Used for summarizing requirement documents, generating test case suggestions, and assisting in project documentation.

**Vector Search (Pinecone):**

* Project documents are embedded using **Sentence Transformers** and stored in Pinecone.
* Enables semantic search with cosine similarity for natural language queries.

**ML Modules (Forecasting and Anomaly Detection):**

* Scikit-learn models handle forecasting of project KPIs and anomaly detection in project performance.
* Data is parsed and visualized with pandas and matplotlib.

## Setup Instructions

**Prerequisites:**

* Python **3.9 or later**
* pip and virtual environment tools
* API keys for **IBM Watsonx** and **Pinecone**
* Internet access for cloud-based AI services

**Installation Steps:**

1. Clone the repository
2. git clone <repo-link>
3. cd smart-sdlc
4. Install dependencies
5. pip install -r requirements.txt
6. Configure credentials in .env
7. WATSONX\_API\_KEY=<your\_ibm\_watsonx\_api\_key>
8. PINECONE\_API\_KEY=<your\_pinecone\_api\_key>
9. Run FastAPI backend
10. uvicorn app.main:app --reload
11. Run Streamlit frontend
12. streamlit run smart\_dashboard.py
13. Open the dashboard in your browser and interact with modules.

**Folder Structure**

App/

├── api/ # API routes (requirements, reports, feedback, anomaly detection)

├── models/ # Data models for project artifacts and metrics

├── services/ # Core AI logic (forecasting, bug detection, summarization)

└── \_\_init\_\_.py

Ui/ # Streamlit frontend components (dashboards, forms, chat)

Smart\_dashboard.py # Entry point for Streamlit frontend

Granite\_llm.py # Handles IBM Watsonx Granite integration

Document\_embedder.py # Embeds project docs into Pinecone

Kpi\_forecaster.py # Forecasts delivery time, cost, defect trends

Anomaly\_checker.py # Detects unusual code/project anomalies

Report\_generator.py # Builds AI-generated project reports

**Running the Application**

* Launch FastAPI server to expose backend endpoints.
* Run Streamlit dashboard to access the UI.
* Navigate via sidebar to different modules.
* Upload project documents (requirements, test results).
* Interact with the AI chat assistant.
* View outputs like summaries, forecasts, anomaly alerts, and reports.

1. **Authentication**

* Token based authentication (JWT/API keys) for securing endpoints.
* OAuth2 with IBM Cloud credentials for IBM watsonx Granite.
* Role based access control(RBAC):
* Admin (project manager) .
* Developer
* Tester
* Stakeholder/viewer

1. **Planned enhancements:**

* User sessions and login/logout.
* History tracking of queries and documents.

**User interface**

* Sidebar navigation (Requirements, Code Review, KPI Dashboard, Reports).
* KPI visualizations with summary cards.
* Tabbed layouts for chat, forecasting, and recommendations.
* Real-time form handling for file uploads.
* PDF report download.
* Help texts and tooltips for guidance.

**Testing**

* Unit Testing: For AI prompts and forecasting modules.
* API Testing: Swagger UI, Postman, test scripts.
* Manual Testing: File uploads, bug predictions, chat responses.
* Edge Cases: Large files, malformed documents, invalid API keys.
* Validated for offline and API-connected modes.

**10. Screenshots**

* SmartSDLC Dashboard (Sidebar navigation)
* KPI Forecasting Dashboard
* Requirement Summarization & AI Chat Assistant
* Bug Prediction Output
* Automated Project Report Download.

**11. Known Issues**

* Limited handling of very large documents.
* Forecasting currently uses basic regression.
* CI/CD integration not fully automated.
* UI performance slows with multiple uploads.

**12.Future Enhancements**

* Advanced deep learning models for forecasting.
* Full CI/CD integration with GitHub/GitLab/Jenkins.
* Enhanced bug prediction for security and performance.
* User session tracking with history.
* Multi-user collaboration dashboards.
* Mobile-friendly UI.