**PROJECT DOCUMENTATION**

**TITLE: Smart SDLC – AI-Enhanced Software Development Lifecycle**

**TEAM MEMBERS:**

* Aafiya kounine.S
* Harini. P
* Keerthi. M
* Keerthiga. P

**PROJECT OVERVIEW**

**Purpose:** The purpose of *SmartSDLC* is to revolutionize the traditional software development lifecycle by integrating artificial intelligence to optimize every phase—from requirements gathering to deployment and maintenance. By leveraging AI-driven automation, predictive analytics, and intelligent decision-making, SmartSDLC aims to reduce development time, minimize errors, enhance code quality, and improve collaboration across teams. Its ultimate goal is to deliver reliable, scalable, and cost-effective software solutions faster and smarter.

**features:**

* AI-powered requirement analysis
* Smart project planning & risk prediction
* Automated code generation & review
* Intelligent testing & bug detection
* Code quality & security checks
* Predictive maintenance & monitoring
* Optimized CI/CD pipelines
* Smart team collaboration & documentation

**conversational interface:**

**Key Points**:

* Natural, context-aware AI assistant
* Works across all SDLC phase
* Supports text, voice & code queries
* Learns from project history

**Functionality**:

* Clarify requirements & generate user stories
* Suggest designs & patterns
* Generate/review code
* Create & run test cases
* Optimize CI/CD & deploy builds

**Policy Summarization:** SmartSDLC establishes a structured framework that governs the responsible and efficient integration of AI within the software development lifecycle. The policy ensures transparency, accountability, and ethical use of AI across all phases—requirements, design, development, testing, deployment, and maintenance. It emphasizes secure handling of project data, adherence to coding and security standards, and compliance with organizational and industry regulations. The system prioritizes automation for productivity while maintaining human oversight to prevent bias or errors in decision-making. Collaboration and communication are streamlined through an AI-driven conversational interface, while continuous monitoring and learning policies guarantee that SmartSDLC adapts and improves over time.

**Resources Forecasting:**

**Key Points**

* **AI-Driven** : required manpower, tools, infrastructure, and budget across SDLC phases.
* **Dynamic Allocation**: Adjusts resources in real time based on project progress and risks.
* **Cost Optimization**: Prevents under- or over-utilization of resources, reducing waste.
* **Scalability**: Adapts to small, medium, or enterprise-level projects.
* **Risk Awareness:** Highlights potential bottlenecks and suggests proactive solutions.

**Functionality**

* Manpower Planning
* Infrastructure Forecasting
* Budget Forecasting
* Time Forecasting
* Risk-Based Resource Adjustment

**Eco-Tip Generator**

**Key points**:

An AI-powered feature that provides sustainability-focused suggestions during software development to reduce environmental impact and promote green computing practices.

**Functionality**:

* Energy-Efficient Coding Tips – Recommends optimized algorithms and code practices to lower CPU/GPU usage.
* Green Infrastructure Suggestions – Suggests use of renewable-energy-based cloud providers or efficient server configurations.
* Paperless Documentation – Encourages digital-first practices and AI-generated documentation to minimize printing.
* Resource Optimization – Provides tips on minimizing storage waste, reducing redundant builds, and optimizing CI/CD pipelines.
* Eco-Friendly Workflows – Suggests remote collaboration and reduced travel practices for distributed teams.
* Carbon Footprint Insights – Tracks estimated energy consumption of builds/tests and offers reduction strategies.

**Citizen Feedback Loop**

### **Key Points**

* **User-Centric Approach**: Involves citizens/end-users in shaping software through continuous feedback.
* **AI-Driven Analysis**: Collects, categorizes, and prioritizes feedback automatically.
* **Transparency & Trust**: Ensures citizens see their input reflected in system improvements.
* **Agility**: Rapidly integrates user suggestions into iterative development cycles.
* **Inclusivity**: Provides multilingual, accessible channels for diverse citizen groups.

**Functionality**

1. **Feedback Collection**
   * Chatbots, surveys, and in-app feedback forms.
   * Voice/text input for accessibility.
2. **Sentiment & Priority Analysis**
   * AI detects satisfaction levels, pain points, and urgent issues.
   * Categorizes feedback by features, bugs, or usability.
3. **Integration with SDLC**
   * Maps citizen feedback directly to requirements and bug tracking tools.
   * Auto-generates user stories or issue tickets.
4. **Feedback-to-Action Tracking**
   * Citizens can track status of their suggestions (e.g., “under review,” “in development”).
   * Builds accountability and trust.
5. **Continuous Improvement**
   * AI identifies recurring patterns in feedback.
   * Suggests long-term design or policy improvements.

**KPI Forecasting**

**Key Points**

* AI-Driven Predictions: Uses historical data and current trends to forecast project KPIs.
* Real-Time Monitoring: Tracks progress and adjusts KPI predictions dynamically.
* Goal Alignment: Ensures KPIs reflect business objectives and SDLC efficiency.
* Proactive Decision-Making: Identifies risks before targets are missed

**Functionality**

1. Performance Metrics Forecasting

2. Time & Cost Forecasting

3. Quality & Risk Forecasting

4. Team Efficiency Forecasting

5. Business Impact Forecasting

**Anomaly Detection** :

### **Key Points**

* **AI-Powered Monitoring**: Continuously scans development, testing, and deployment processes.
* **Early Warning System**: Flags unusual behaviors before they cause failures.
* **Multi-Phase Coverage**: Detects anomalies in code, performance, security, and resources.
* **Adaptive Learning**: Improves detection accuracy with historical project data.
* **Risk Mitigation**: Helps teams act quickly to prevent downtime or defects.

### **Functionality**

1. **Code Anomaly Detection**
   * Identifies unusual coding patterns, vulnerabilities, or non-standard practices.
   * Flags sudden spikes in bug introduction rate.
2. **Testing & QA Anomalies**
   * Detects irregular test failures or inconsistent results.
   * Alerts if coverage drops below expected thresholds.
3. **Performance Anomalies**
   * Monitors system response times, memory usage, and CPU load.
   * Detects sudden slowdowns or unexpected spikes.
4. **Deployment Anomalies**
   * Flags unusual build failures or deployment rollbacks.
   * Detects unexpected downtime during releases.
5. **Resource & Cost Anomalies**
   * Identifies abnormal cloud usage or infrastructure costs.
   * Predicts unusual workload surges.
6. **User Behavior Anomalies**
   * Tracks end-user activity for security breaches or unusual patterns.
   * Helps prevent fraud or malicious use.

**Multimodal Input Support**:

### **Key Points**

* **Multiple Input Channels**: Supports text, voice, code, diagrams, and images.
* **Accessibility & Inclusivity**: Enables users with different preferences or abilities to interact naturally.
* **Context-Aware Processing**: Understands and integrates inputs across SDLC phases.
* **Seamless Collaboration**: Allows multiple team members to contribute in their preferred format.
* **Enhanced Productivity**: Reduces friction and accelerates decision-making.

### 

### 

### **Functionality**

1. **Text Input**
   * Chat, notes, requirements, and queries for AI analysis.
2. **Voice Input**
   * Dictate requirements, bug reports, or commands to the AI assistant.
3. **Code Snippets**
   * Submit code for instant review, optimization, or anomaly detection.
4. **Diagrams & Images**
   * Upload architecture diagrams, UI mockups, or logs for AI interpretation.
5. **Cross-Modal Integration**
   * Combines different input types for a holistic understanding of tasks.
   * Example: Voice description + diagram → AI generates user story or design suggestion.
6. **Real-Time Feedback**
   * Provides instant suggestions, error detection, or insights based on input t**type**

**Streamlit and Gradio UI**

### **Key Points**

* **Dashboard + Visualization (Streamlit)**
  + Project KPIs, resource forecasts, anomaly detection charts.
  + Phase-wise tabs: Requirements, Design, Development, Testing, Deployment, Maintenance.
  + Real-time updates on code quality, performance metrics, and user feedback.
* **Conversational & Multimodal AI (Gradio)**
  + Chatbot for queries, suggestions, and AI guidance.
  + Supports text, voice, code snippets, diagrams, and images.
  + Quick action buttons: Generate test cases, review code, deploy builds, suggest eco-tips.
* **Integrated Workflow**
  + Streamlit visualizations update based on Gradio interactions.
  + AI suggestions in Gradio feed directly into dashboards (e.g., resource allocation, KPI updates, anomaly alerts).
  + Users can seamlessly switch between analyzing metrics and interacting with the AI assistant.

### **Functionality Examples**

1. Developer pastes code in Gradio → AI detects anomalies → Streamlit dashboard updates code quality and bug trends.
2. Project manager uploads requirement document → AI analyzes ambiguities → KPI and progress charts adjust in Streamlit.
3. Tester asks AI via voice input → “Generate regression tests for payment module” → Suggestions appear in Gradio and test coverage charts in Streamlit.
4. Eco-tip generator suggests energy-efficient coding → Updates reflected in sustainability dashboard.

**ARCHITECTURE**

**FRONTEND (STREAM LIT)**

The **Streamlit frontend** of SmartSDLC serves as the interactive dashboard and visualization layer, providing real-time insights and monitoring across all phases of the software development lifecycle. It features project overview panels, phase-specific tabs, and anomaly alerts, along with interactive widgets like buttons, sliders, file uploads, and filters. Streamlit communicates with backend AI modules via APIs to fetch processed data for requirement analysis, code quality checks, KPI forecasting, and eco-tips, updating visualizations and dashboards in real time. While it does not store data itself, it integrates seamlessly with databases, CI/CD tools, and cloud infrastructure to display insights, enabling teams to make informed, timely decisions and efficiently manage the development process.

**BACKEND (FAST API)**

The FastAPI backend of SmartSDLC acts as the core processing layer, providing high-performance RESTful endpoints for all system functionalities. It handles requests from the Streamlit dashboard and Gradio conversational interface, managing AI-driven tasks such as requirement analysis, code quality checks, anomaly detection, KPI and resource forecasting, and eco-tip generation. The backend integrates machine learning models, business logic, and multimodal input processing to deliver intelligent recommendations and predictions. It securely manages data through relational and NoSQL databases, with optional caching for faster responses, while enforcing authentication and role-based access control. This architecture ensures seamless communication with the frontend, real-time analytics, and scalable, efficient operation across the software development lifecycle.

### 

### **LLM Integration (IBM Watson Granite)**

* Serves as the **natural language understanding and generation engine**.
* Handles:
  + Requirement analysis and ambiguity detection.
  + Conversational interface queries (via Gradio/Streamlit chat).
  + User story generation, code suggestions, and eco-tips.
* Interacts with **FastAPI backend** through REST APIs for processing and returning insights.

### **Vector Search (Pinecone)**

* Provides **semantic search and retrieval** for project documentation, code snippets, and historical feedback.
* Supports:
  + Efficient lookup of similar requirements or past solutions.
  + Retrieval of relevant patterns for anomaly detection and KPI forecasting.
* Integrated with the LLM to enhance **contextual understanding** during queries.

### **ML Modules**

* **Forecasting Modules**
  + Predict KPIs, resource needs, timelines, and costs using historical project data.
  + Integrated with dashboard visualizations (Streamlit).
* **Anomaly Detection Modules**
  + Monitor code, testing, performance, deployment, and user behavior.
  + Flag irregularities and generate early warnings.
* Communicates with FastAPI backend to provide **real-time predictions** to the frontend.

**SETUP INSTRUCTIONS**

## 

## **1. Prerequisites**

**System Requirements:**

* OS: Windows 10/11, macOS, or Linux
* RAM: ≥ 8 GB (16 GB recommended for ML modules)
* Disk: ≥ 20 GB free space
* Python: ≥ 3.10
* Node.js (if using additional frontend integrations)

**Software & Tools:**

* **Python packages:** FastAPI, Uvicorn, Streamlit, Gradio, Pydantic, Pandas, NumPy, Matplotlib, Scikit-learn, TensorFlow/PyTorch (for ML modules)
* **Database:** PostgreSQL or MySQL (relational), MongoDB (NoSQL), Pinecone account for vector search
* **AI Services:** IBM Watson Granite account for LLM integration
* **Version Control:** Git
* Optional: Docker for containerized deployment

## 

## **2. Installation Process**

### **Step 1: Clone the Repository**

git clone https://github.com/your-org/SmartSDLC.git

cd SmartSDLC

### **Step 2: Set Up Python Environment**

python -m venv venv

source venv/bin/activate # macOS/Linux

venv\Scripts\activate # Windows

### **Step 3: Install Required Packages**

pip install -r requirements.txt

**Requirements may include:**

* fastapi, uvicorn, streamlit, gradio, pydantic
* numpy, pandas, matplotlib, scikit-learn, tensorflow/torch
* requests (for API calls to IBM Watson Granite)
* pinecone-client (for vector search integration)

### **Step 4: Configure Environment Variables**

Create a .env file with:

IBM\_WATSON\_API\_KEY=your\_ibm\_watson\_api\_key

IBM\_WATSON\_URL=your\_ibm\_watson\_url

PINECONE\_API\_KEY=your\_pinecone\_api\_key

DATABASE\_URL=postgresql://user:password@localhost:5432/smart\_sdlc

### **Step 5: Initialize Databases**

* For PostgreSQL/MySQL: Run provided SQL scripts to create tables.
* For MongoDB: Create required collections (e.g., logs, feedback).
* For Pinecone: Initialize index for embeddings.

### **Step 6: Start Backend (FastAPI)**

uvicorn backend.main:app --reload

### **Step 7: Start Streamlit Frontend**

streamlit run frontend/streamlit\_app.py

### **Step 8: Start Gradio Interface (if separate)**

python frontend/gradio\_app.py

### **Step 9: Access UI**

* **Streamlit Dashboard:**<http://localhost:8501>
* **Gradio Chat Interface:**<http://localhost:7860>

**FOLDER STRUCTURE**

The SmartSDLC project is organized modularly to ensure scalability, maintainability, and AI integration across the software development lifecycle. The backend/ directory contains the FastAPI application with subfolders such as backend/routes/ for API endpoints, backend/models/ for Pydantic schemas, backend/services/ for business logic and AI/ML integrations including IBM Watson Granite (backend/services/lml\_integration.py), Pinecone (backend/services/vector

\_search.py), backend/services/forecasting.py for KPI/resource predictions, and backend/services/anomaly.py for anomaly detection. Database connections are in backend/database/ and helper functions in backend/utils/. The frontend/ folder contains frontend/streamlit\_app.py for dashboards and frontend/gradio\_app.py for conversational AI, with frontend/components/ for reusable UI widgets and frontend/assets/ for images, diagrams, and CSS. Machine learning models are stored in ml\_models/ with subfolders for anomaly\_detection\_model/, kpi\_forecasting\_model/, and embeddings/ for Pinecone vectors. Supporting scripts are in scripts/ such as scripts/data\_preprocessing.py and scripts/db\_init.py, while tests are under tests/backend\_tests/ and tests/frontend\_tests/. Core files include requirements.txt, Dockerfile, .env, and README.md. This structure provides clear separation of concerns, modularity, and seamless integration of AI-enhanced SDLC features.

**RUNNING THE APPLICATION**

To run the SmartSDLC application, start by setting up the backend with FastAPI. Activate the Python virtual environment using source venv/bin/activate on macOS/Linux or venv\Scripts\activate on Windows, and install the required dependencies with pip install -r requirements.txt. Configure the .env file with necessary API keys for IBM Watson Granite, Pinecone, and database connection URLs. Launch the FastAPI server using uvicorn backend.main:app --reload, which will start the backend at http://localhost:8000 and provide interactive API documentation at http://localhost:8000/docs. Once the backend is running, start the frontend using Streamlit by executing streamlit run frontend/streamlit\_app.py from the project root. This will open the dashboard in a browser at http://localhost:8501, displaying project KPIs, resource forecasts, anomaly alerts, and AI-generated recommendations. If using the Gradio conversational interface, run python frontend/gradio\_app.py to access a chat-based AI assistant at http://localhost:7860, which supports text, voice, code, and diagram inputs. Ensure the backend is active before starting the frontend so that all AI insights and real-time data can be fetched seamlessly, providing a fully interactive and intelligent SDLC management environment.

**API DOCUMENTATION**

The SmartSDLC API provides a comprehensive set of endpoints to facilitate AI-enhanced software development across all SDLC phases. The backend, built with FastAPI, exposes RESTful APIs that support requirement analysis, code quality review, KPI forecasting, anomaly detection, and eco-tip generation. For requirements analysis, the endpoint /api/requirements/analyze accepts project IDs along with requirement text or uploaded documents, processes them using IBM Watson Granite, and returns ambiguous terms along with improvement suggestions, helping teams clarify unclear or incomplete specifications. The code quality and security check endpoint, /api/code/review, allows developers to submit code snippets, which the system analyzes for vulnerabilities, coding standard violations, and best practice recommendations, returning a structured list of issues with line numbers and severity types.For project management, the /api/kpi/forecast endpoint provides KPI forecasting, delivering predicted metrics such as project velocity, estimated completion time, budget utilization, and predicted bug density. These insights are generated by ML forecasting modules and can be queried via GET requests using the project ID. The /api/anomaly/detect endpoint supports anomaly detection across code, testing, deployment, and system performance. Users submit log data or system metrics, and the ML models identify unusual patterns, spikes, or failures, returning timestamps, anomaly types, and detailed messages, enabling proactive intervention and risk mitigation.

Additionally, the API offers an eco-tip generation endpoint, /api/eco/tip, which provides actionable suggestions for energy-efficient coding practices, cloud resource optimization, and sustainable development strategies. All endpoints are secured with authentication, such as JWT or OAuth2, and the API is designed to handle both synchronous and asynchronous requests to ensure performance and scalability. Integration with the frontend, including the Streamlit dashboard and Gradio conversational interface, allows real-time visualization of results and interaction with AI models.

**ATHENTICATION**

**In SmartSDLC – AI-Enhanced Software Development Lifecycle, authentication is a critical component that ensures secure access to both frontend and backend resources, protecting sensitive project data, code repositories, and AI-driven insights. The system employs a token-based authentication mechanism, typically using JWT (JSON Web Tokens), combined with role-based access control to verify and manage user permissions. When a user logs in via the Streamlit dashboard or Gradio interface, credentials are validated against a secure database that stores hashed passwords using modern cryptographic standards such as bcrypt. Upon successful verification, the backend issues a JWT token containing the user’s identity and role information, which is then used for subsequent API requests to authorize access to different endpoints.Roles can include developers, testers, project managers, and administrators, each with varying levels of permissions. For example, developers may have access to code review and testing endpoints, testers can submit logs for anomaly detection, project managers can view KPI dashboards and resource forecasts, and administrators can manage user accounts and system configurations. Each API request includes the JWT in the header, which the FastAPI backend verifies before processing the request. Invalid, expired, or missing tokens result in access denial, ensuring that only authenticated users can interact with the system.Additionally, SmartSDLC supports OAuth2 integration for enterprise environments, allowing users to authenticate via existing organizational accounts, such as Google Workspace or Microsoft Azure Active Directory, streamlining onboarding and enforcing single sign-on policies. Multi-factor authentication (MFA) can also be optionally enabled to enhance security further.**

**USER INTERFACE**

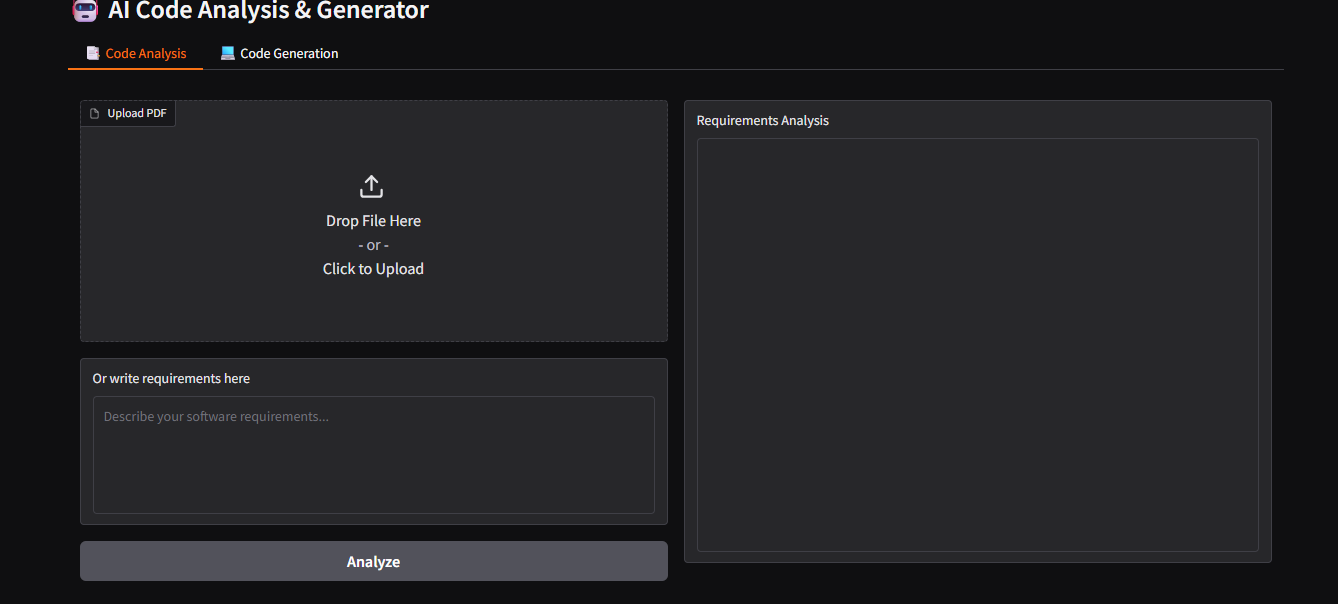
**The user interface (UI) of SmartSDLC – AI-Enhanced Software Development Lifecycle is designed to provide an intuitive, interactive, and efficient environment for managing all phases of software development, while seamlessly integrating AI capabilities. The UI is divided into two complementary components: the Streamlit dashboard and the Gradio conversational interface, enabling both visual analytics and natural language interactions. The Streamlit dashboard serves as the primary visual interface, offering phase-specific tabs for Requirements, Design, Development, Testing, Deployment, and Maintenance. Users can view real-time project KPIs, resource forecasts, anomaly alerts, and system performance metrics through interactive charts, tables, and progress indicators. Widgets such as sliders, buttons, file upload, and filters allow users to manipulate data, submit project artifacts, and explore insights efficiently.The Gradio interface provides a multimodal conversational AI assistant capable of handling text, voice, code snippets, and diagrams. Users can submit requirements, ask questions, request code reviews, or generate test cases directly in natural language. The AI processes these inputs using LLMs like IBM Watson Granite and returns actionable recommendations, which are reflected both in the conversational panel and the Streamlit dashboard. This integration ensures a two-way interaction: user inputs inform backend AI modules, while AI insights update dashboards, charts, and reports in real-time.The UI also emphasizes user accessibility and role-based customization. Developers, testers, project managers, and administrators see only the features relevant to their roles, ensuring clarity and efficiency. Notifications and alerts are highlighted for quick decision-making, and all actions, such as code submission, deployment triggers, or KPI updates, are logged for traceability. Assets such as diagrams, mockups, and uploaded documents are organized and displayed contextually within the interface, supporting collaborative work.**

**TESTING**

**Testing in SmartSDLC – AI-Enhanced Software Development Lifecycle is designed to ensure the reliability, security, and performance of both the software being developed and the SmartSDLC platform itself. The testing framework is integrated into multiple stages of the SDLC, leveraging AI and automation to optimize quality assurance. During the development phase, unit tests are automatically generated and executed for newly written code using AI-driven test case generators. These tests validate individual functions and modules for correctness, efficiency, and adherence to coding standards. The AI models also perform static and dynamic code analysis, detecting vulnerabilities, logic errors, or performance bottlenecks before integration.**

**In the integration and testing phase, SmartSDLC employs regression and functional testing, where AI predicts potential areas of failure based on historical project data and anomaly patterns. Automated test suites run continuously, covering APIs, workflows, and system interactions to ensure that new changes do not break existing functionality. The platform also supports multimodal testing inputs, including test scripts, logs, and user feedback, allowing AI to identify inconsistencies, gaps, or unexpected behavior. Performance testing monitors CPU, memory, response times, and throughput to detect resource bottlene**

**Screenshot**



**KNOWN ISSUES**

Known Issues in SmartSDLC – AI-Enhanced Software Development Lifecycle highlight current limitations and areas for improvement in the platform, helping teams anticipate challenges during deployment and use. One key issue is the dependency on external AI services, such as IBM Watson Granite for natural language processing and Pinecone for vector search. Network latency or service outages can temporarily disrupt requirement analysis, semantic search, and conversational AI functionalities, affecting real-time responsiveness. Closely related is the learning curve for new users, especially non-technical stakeholders, who may find the multimodal interface—combining Streamlit dashboards and Gradio chat—complex to navigate initially, potentially slowing adoption.

Another limitation lies in scalability under large datasets. While the platform is designed to handle moderate project sizes efficiently, extremely large codebases, logs, or historical data for AI training can increase processing time for anomaly detection, KPI forecasting, and vector-based searches, requiring additional optimization or infrastructure scaling. Similarly, AI model accuracy may vary depending on project domain, coding style, or input quality. LLM-based requirement analysis, code review, or eco-tip suggestions may occasionally produce ambiguous, incomplete, or irrelevant recommendations, necessitating human validation.Integration challenges with existing enterprise tools, CI/CD pipelines, and cloud environments may also arise. Misconfigured API keys, database connections, or version mismatches can cause intermittent errors in data retrieval or processing. Additionally, security and access control limitations exist if role-based permissions are not configured correctly, potentially exposing sensitive project data.Finally, real-time feedback loops such as anomaly detection or citizen feedback processing may generate false positives or overlook subtle issues due to incomplete or noisy input data. These known issues emphasize the importance of human oversight, iterative improvement, and careful infrastructure planning. Despite these limitations, SmartSDLC remains a robust AI-enhanced platform, and continuous updates, model retraining, and user guidance are planned to address these challenges, improving reliability, scalability, and user experience across all phases of the software development lifecycle.

**FUTURE Enhancement**

Future Enhancements for SmartSDLC – AI-Enhanced Software Development Lifecycle focus on expanding AI capabilities, improving scalability, and enhancing user experience. One key area is advanced AI integration, including more sophisticated LLMs and domain-specific models that provide deeper code insights, automated documentation, and intelligent design recommendations. Incorporating reinforcement learning could allow the system to adapt dynamically to team workflows, optimizing task allocation, sprint planning, and resource forecasting over time.

Enhanced collaboration features are planned, such as real-time multi-user editing, shared dashboards, and interactive visualization of project metrics, enabling distributed teams to coordinate seamlessly. The platform may also support extended multimodal inputs, including video tutorials, voice-based code walkthroughs, and augmented reality (AR) diagrams for design and architecture review.

Scalability and performance improvements are anticipated, with cloud-native deployment, container orchestration, and distributed AI processing to handle larger projects and datasets efficiently. Security enhancements, including adaptive access control and real-time threat monitoring, will ensure enterprise-grade compliance.

Additionally, sustainability-focused features will provide more precise eco-tips and energy consumption tracking for development activities. By integrating predictive analytics, continuous learning from historical projects, and more interactive user interfaces, SmartSDLC aims to become an even smarter, more efficient, and user-friendly platform, driving productivity, quality, and sustainability throughout the software development lifecycle.