SmartSDLC – AI-Enhanced Software Development Lifecycle

1. Introduction

Projecttitle: **SmartSDLC – AI-Enhanced Software Development Lifecycle**

- **Teamleader:** MANIMARAN A
- **Teammember:** MOHAMED ITHRIS A
- **Teammember:** MANOJ K
- **Teammember:** NITHESH K

2.Projectoverview

1. Requirements Gathering & Analysis

AI Enhancements:

- **Natural Language Processing (NLP)** for parsing client documents, emails, or voice into user stories or requirements.
- **AI chatbots/assistants** for stakeholder interviews and clarification.
- **Machine learning** to suggest missing or implied requirements based on similar past projects.

Benefits:

- Reduced ambiguity
- Faster turnaround
- Automatic traceability links

---

2. Planning

AI Enhancements:

- **Effort estimation models** trained on historical project data.
- **Risk prediction** via AI models (project delays, budget overruns).
- **Sprint planning automation** using prioritization algorithms.

Benefits:

- Data-driven decisions
- Better resource allocation
- More accurate timelines

---

3. Design

AI Enhancements:

- **AI-assisted architecture modeling tools**
- **Pattern recommendation engines** based on functional/non-functional requirements.
- **Code scaffolding AI** to propose design skeletons or templates.

Benefits:

- Standardized designs
- Faster prototyping
- Reduction in design errors

---

4. Implementation / Coding

AI Enhancements:

- **AI pair programming tools** (e.g., GitHub Copilot, CodeWhisperer)
- **Code completion and refactoring suggestions**
- **Secure code generation** using AI vulnerability models

Benefits:

- Increased developer productivity
- Reduced coding errors
- Higher code quality

---

5. Testing

AI Enhancements:

- **Test case generation** from requirements or code
- **Automated bug detection** using static analysis and AI
- **Predictive QA analytics** for identifying high-risk areas in code

Benefits:

- Improved test coverage
- Faster feedback cycles
- Higher software reliability

---

6. Deployment

AI Enhancements:

- **CI/CD pipeline optimization** using ML to identify bottlenecks
- **AI-based deployment strategies** (e.g., canary analysis)
- **Environment config recommendations** using past deployment data

Benefits:

- Lower downtime
- Smoother releases
- Automated rollbacks

---

7. Maintenance & Monitoring

AI Enhancements:

- **Anomaly detection in logs** (AIOps)
- **Predictive maintenance** using historical error patterns
- **ChatOps** integration for faster issue resolution

Benefits:

- Proactive issue detection
- Reduced MTTR (mean time to resolution)
- Continual learning and improvement

---

8. Documentation & Knowledge Management

AI Enhancements:

- **Auto-generation of technical documentation**
- **Semantic search** across project artifacts
- **Chat-based project assistants** trained on codebase and wiki

Benefits:

- Updated, accurate documentation
- Faster onboarding
- Improved collaboration

Optional View: AI Models/Tools by Phase

Phase	Example AI Tools/Models
Requirements	OpenAI (NLP), ChatGPT, Azure Cognitive Services
Planning	Microsoft Azure ML, TensorFlow models
Design	Mermaid.js with AI assistance, Sketch2Code
Coding	GitHub Copilot, CodeWhisperer
Testing	Test.AI, Diffblue Cover
Deployment	Harness, Spinnaker with AI plugins
Maintenance	Dynatrace, Splunk + ML, DataDog AI
Documentation	Mintlify, Docusaurus + NLP

Continuous Feedback Loop

Each phase should integrate feedback from the next to form a **closed-loop system**, enabling:

- Adaptive learning
- Continuous process improvement
- Enhanced project agility

Final Thoughts

SmartSDLC isn't just about adding AI to existing tools; it's about **re-imagining how software is built**, with AI playing a collaborative, adaptive, and decision-support role throughout the lifecycle.

Would you like a **visual diagram**, **implementation roadmap**, or a **demo template** of a SmartSDLC system?

3.Architecture

Frontend(Streamlit):

The frontend is built with Stream lit, offering an interactive web UI with multiple pages including dashboards, file uploads, chat interface, feedback forms,andreportviewers.Navigationishandledthroughasidebarusingthe stream lit-option-menu library. Each page is modularized for scalability.

Backend(FastAPI):

FastAPIserves as thebackendRESTframeworkthatpowers APIendpoints for document processing, chat interactions, eco tip generation, report creation, andvectorembedding.Itisoptimizedforasynchronousperformanceandeasy Swagger integration.

LLMIntegration(IBMWatsonxGranite):

Granite LLM models from IBM Watsonx are used for natural language understandingandgeneration.Promptsarecarefullydesignedtogenerate summaries, sustainability tips, and reports.

VectorSearch(Pinecone):

UploadedpolicydocumentsareembeddedusingSentenceTransformersand storedinPinecone.Semanticsearchisimplementedusingcosinesimilarityto allow users to search documents using natural language queries.

MLModules(ForecastingandAnomalyDetection):

LightweightMLmodelsareusedforforecastingandanomalydetectionusing Scikit-learn.Time-seriesdataisparsed,modeled,andvisualizedusingpandas and matplotlib.

2. SetupInstructions

Prerequisites:

- Python3.9orlater
- pipandvirtualenvironmenttools
- APIkeysforIBMWatsonxandPinecone
- Internetaccesstoaccesscloudservices

InstallationProcess:

- Clonetherepository
- Installdependenciesfromrequirements.txt
- Createa.envfileandconfigurecredentials
- RunthebackendserverusingFastAPI
- LaunchthefrontendviaStreamlit
- Uploaddataandinteractwiththemodules

3. FolderStructure

app/–ContainsallFastAPIbackendlogicincludingrouters,models,and integration modules.

app/api/–SubdirectoryformodularAPIrouteslikechat,feedback,report,and document vectorization.

ui/–ContainsfrontendcomponentsforStreamlitpages,cardlayouts,and form UIs.

smart_dashboard.py–EntryscriptforlaunchingthemainStreamlit dashboard.

granite_llm.py–HandlesallcommunicationwithIBMWatsonxGranitemodel including summarization and chat.

document_embedder.py–Convertsdocumentstoembeddingsandstoresin Pinecone.

kpi_file_forecaster.py–Forecastsfutureenergy/watertrendsusingregression.

anomaly_file_checker.py – Flags unusual values in uploaded KPI data.

report_generator.py– ConstructsAI-generatedsustainabilityreports.

4. RunningtheApplication

Tostarttheproject:

- LaunchtheFastAPIservertoexposebackendendpoints.
- RuntheStreamlitdashboardtoaccessthewebinterface.
- Navigatethroughpagesviathesidebar.
- UploaddocumentsorCSVs,interactwiththechatassistant,andview outputs like reports, summaries, and predictions.

- Allinteractionsarereal-timeandusebackendAPIstodynamically update the frontend.

Frontend(Streamlit):

The frontend is built with Stream lit, offering an interactive web UI with multiple pages including dashboards, file uploads, chat interface, feedback forms,andreportviewers.Navigationishandledthroughasidebarusingthe stream lit-option-menu library. Each page is modularized for scalability.

Backend(FastAPI):

FastAPIserves as thebackendRESTframeworkthatpowers APIendpoints for document processing, chat interactions, eco tip generation, report creation, andvectorembedding.Itisoptimizedforasynchronousperformanceandeasy Swagger integration.

5.APIDocumentation

BackendAPIsavailableinclude:

POST/chat/ask–AcceptsauserqueryandrespondswithanAI-generated message

POST/upload-doc–UploadsandembedsdocumentsinPinecone

GET/search-docs–Returnssemanticallysimilarpoliciestotheinputquery

GET/get-eco-tips–Providessustainabilitytipsforselectedtopicslikeenergy, water, or waste

POST/submit-feedback–StorescitizenfeedbackforlaterrevieworanalyticsEachendpointistestedanddocumentedinSwaggerUIforquickinspection and trial during development.

6.Authentication

eachendpointistestedanddocumentedinSwaggerUIforquickinspection and trial during development.

Thisversionoftheprojectrunsinanopenenvironmentfordemonstration. However, secure deployments can integrate:

- Token-basedauthentication(JWTor API keys)
- OAuth2withIBMCloudcredentials
- Role-basedaccess(admin,citizen,researcher)
- Plannedenhancementsincludeusersessionsandhistorytracking.8. Authentication

7.UserInterface

Theinterfaceisminimalistandfunctional,focusingonaccessibilityfornon- technical users. It includes:

Sidebarwithnavigation

KPIvisualizationswithsummarycards

Tabbedlayoutsforchat,ecotips,andforecasting

Real-time form handling

PDFreportdownloadcapability

Thedesignprioritizesclarity,speed,anduserguidancewithhelptextsand intuitive flows.

8.Testing

Testingwasdoneinmultiplephases:

UnitTesting:Forpromptengineeringfunctionsandutilityscripts API

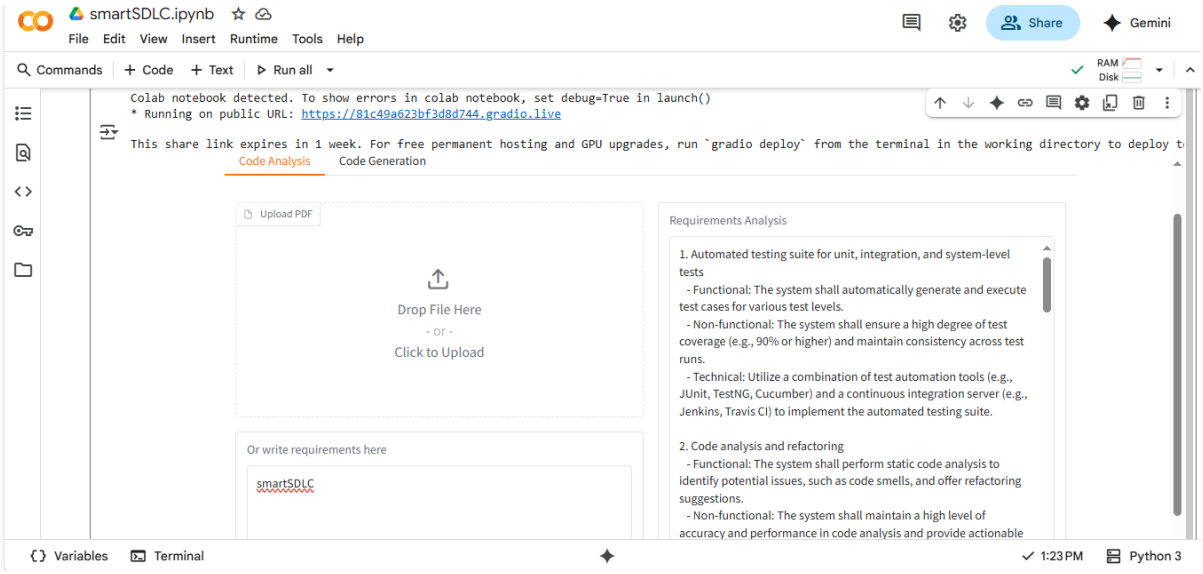
Testing: Via Swagger UI, Postman, and test scripts

ManualTesting:Forfileuploads,chatresponses,andoutputconsistency Edge

Case Handling: Malformed inputs, large files, invalid API keys

Eachfunction was validatedto ensure reliability in both ofline andAPI-connected modes.

9.Screenshots



10.KnownIssues

1. Data Privacy & Security Concerns

- AI tools often require access to codebases, user data, or logs.
- Improper configuration may expose sensitive IP or PII (Personally Identifiable Information).
- Cloud-based AI solutions may conflict with data sovereignty policies.

Mitigation: Use on-premises models or enforce strict access controls and encryption.

11.Futureenhancement

- This section outlines the potential future improvements and expansions planned for the SmartSDLC framework to increase automation, adaptability, and intelligent decision-making throughout the software lifecycle.