# SmartSDLC – AI-Enhanced Software Development Lifecycle

## 1. Introduction

**Project Title**: **SmartSDLC** – AI-Enhanced Software Development Lifecycle

**Team Members**:

Afra Juwairiya

Dharshini S

Varshini D V

Varshini J

Rizwana G

## 2. Project Overview

Purpose:

The goal of SmartSDLC is to streamline the Software Development Lifecycle (SDLC) using AI technologies such as IBM Watsonx Granite LLM, FastAPI, LangChain, and Streamlit.

The platform automates requirement classification, code generation, debugging, and documentation, reducing manual effort and improving productivity.

**Key Features:**

Requirement Upload & Classification – Upload unstructured requirements in PDF, classify them into SDLC phases, and convert into structured user stories.

**AI Code Generator** – Convert natural language prompts into production-ready code.

**Bug Fixer** – Identify and fix syntax & logic issues in code automatically.

**AI-Powered Chatbot** – Real-time project assistance.

**Modular & Secure Backend** – Built using FastAPI, scalable and API-documented with Swagger UI.

## 3. SDLC Phases with Examples

**Phase 1:** Requirement Analysis

Users upload PDFs with unstructured requirements

AI processes requirements and transforms them into structured user stories.

🔹 **Example Requirement (Input):**

"The application should allow users to reset their password securely."

🔹 **Structured User Story (AI Output):**

*As a user, I want to reset my password so that I can regain access to my account securely.*

**Phase 2**: System Design

**Backend**: FastAPI for API management, LangChain for workflow integration.

**Frontend**: Streamlit dashboard for interactive visualizations.

**Authentication**: JWT-based role authentication (future enhancement).

## Example Design Output:

Entity Diagram for User Request → AI Model → Response Generator

- **API Endpoints:**
  - `POST /analyze_requirements` – Process uploaded requirements
  - `POST /generate_code` – Convert natural language to code
  - `POST /fix_bug` – Debug provided code

**Phase 3:** Development

AI-powered code generator creates functional code from natural prompts.

**Example Prompt:**

“Generate a Python function to validate an email address using regex.”

🔹 **AI-Generated Output:**

```
import re
```

```
def validate_email(email):
    pattern = r'^[\w\.-]+@[\w\.-]+\.\w+$'
    return bool(re.match(pattern, email))
```

**Phase 4**: Testing

Generated code is automatically tested; errors are corrected by the AI.

🔹 **Test Case Example:** Input: `validate_email("test@gmail.com")` Output: `True`

🔹 **Bug Fixer Example:**

➕ Input Code (Buggy):

```
def multiply(a, b):
    return a + b   # Wrong operator
```

🟩AI-Fixed Code:

```
def multiply(a, b):
    return a * b
```

Phase 5: Deployment & Maintenance**

- Backend deployed with **FastAPI + Uvicorn**
- Frontend hosted with **Streamlit**
- APIs documented via **Swagger UI**

🔹 **Future Enhancements:**

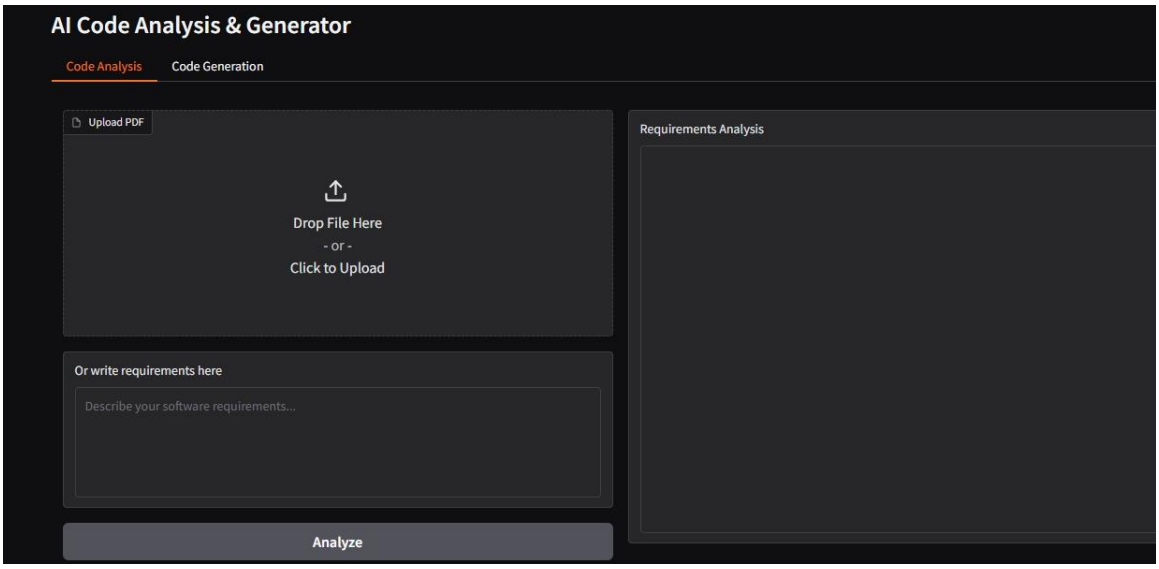
- Real-time database integration (user data, logs)
- Role-based access control (admin, developer, tester)
- Multi-language support

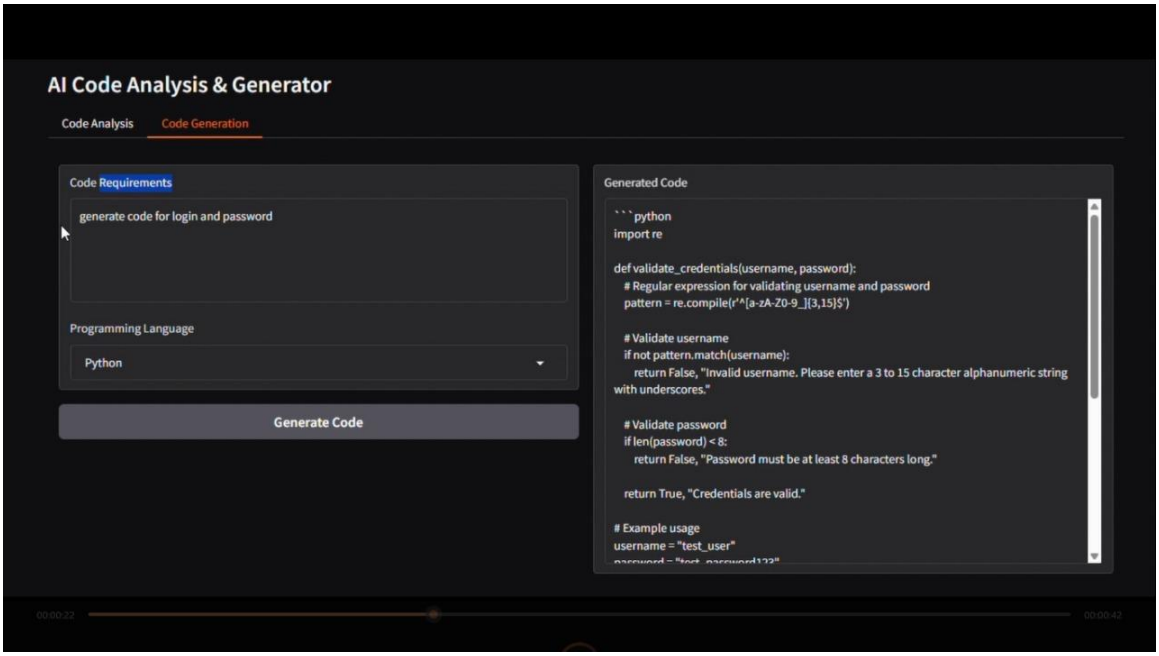
## 4. Example Workflow (End-to-End)

1. Upload requirement document (e.g., "User authentication system").
2. AI classifies text → generates user stories.
3. Select **Code Generator** → enter prompt *"Generate login validation function in Python"*.
4. AI outputs working code.
5. Paste buggy code into **Bug Fixer** → AI returns corrected version.

6. Results shown in **Streamlit dashboard** → exportable to PDF.

# 5. Output




# 6. Conclusion

**SmartSDLC** shows how AI can automate the entire SDLC – from requirement analysis to deployment – making development faster, accurate, and more efficient. This reduces manual effort, improves software quality, and helps teams focus on innovation.