SmartSDLC - AI-Enhanced Software Development Lifecycle

SmartSDLC is an intelligent AI-powered platform that automates the various phases of the Software Development Lifecycle (SDLC) using IBM Watsonx, LangChain, Streamlit, and FastAPI. It empowers users to accelerate software development through intelligent requirement classification, code generation, test automation, bug fixing, and much more.

Team ID: LTVIP2025TMID60749
Team Size: 1
Team Leader: Gummalla Dharanesh Kumar

Team Members

This project was developed as part of a collaborative academic/innovation initiative. I done this project Alone

Problem Statement

Traditional software development is time-consuming, error-prone, and manually intensive. SmartSDLC addresses this by using generative AI to streamline and automate critical SDLC tasks.

Features & Functionalities

Feature	Description
Requirement Analysis	Extracts SDLC phases from uploaded PDF requirements and generates user stories
Code Generator	Generates production-ready code from natural language tasks
Test Case Generation	Produces test cases for generated or existing code
Bug Fixer	Detects and resolves bugs in code snippets
Code Summarizer	Summarizes code into readable documentation
AI Chatbot Assistant	Floating chatbot to answer SDLC questions using LangChain + Watsonx
Feedback Collector	Captures user feedback for continuous improvement
GitHub Integration	Auto pushes AI-generated code, opens issues, or syncs docs

SmartSDLC is a full-stack, AI-powered platform that revolutionizes the traditional Software Development Lifecycle (SDLC) by automating key stages using advanced Natural Language Processing (NLP) and IBM Watsonx AI technologies.

Features

Requirement Upload and Classification

- Upload PDF documents containing raw, unstructured requirements
- Automatically classify sentences into SDLC phases (Requirements, Design, Development, Testing, Deployment)
- Generate structured user stories from requirements

AI Code Generator

- Generate production-ready code from natural language descriptions
- Support for multiple programming languages (Python, JavaScript, Java, C++, C#)
- Clean, commented, and optimized code output

Bug Fixer

- Automatically identify and fix bugs in code
- Support for multiple programming languages
- Detailed explanations of fixes applied

Test Case Generator

- Generate comprehensive unit tests for your code
- Support for popular testing frameworks (unittest, pytest, Jest)
- Cover normal cases, edge cases, and error conditions

Code Summarizer

• Generate human-readable documentation from code
• Extract key functions, parameters, and use cases
• Perfect for code reviews and onboarding

AI Assistant Chatbot

• Real-time conversational support
• SDLC best practices guidance
• Integrated throughout the application

Feedback System

• Collect and analyze user feedback
• Rating system for continuous improvement
• Feature-specific feedback collection

Project Structure

```
SmartSDLC/
 ├── __init__.cpython-313.pyc
 ├── chat_routes.cpython-313.pyc
 ├── ai_routes.cpython-313.pyc
 ├── feedback_routes.cpython-313.pyc
 ├── ai_routes.py
 ├── __init__.py
 ├── feedback_routes.py
 ├── chat_routes.py
 ├── pdf_service.cpython-313.pyc
 ├── watsonx_service.cpython-313.pyc
 ├── __init__.cpython-313.pyc
 ├── pdf_service.py
 ├── watsonx_service.py
 ├── __init__.py
 ├── main.cpython-313.pyc
 ├── feedback_data.json
 ├── main.py
 ├── server.js
 ├── Home.py
 ├── FloatingChatbot.tsx
 ├── Navbar.tsx
 ├── CodeGenerator.tsx
 ├── PDFClassifier.tsx
 ├── BugFixer.tsx
 ├── Home.tsx
 ├── TestGenerator.tsx
 ├── mockApi.ts
 ├── App.tsx
 ├── index.css
 ├── main.tsx
 ├── vite-env.d.ts
 ├── config.json
 ├── .env
 ├── prompt
 ├── .gitignore
 ├── tailwind.config.js
 ├── package-lock.json
 ├── eslint.config.js
 ├── tsconfig.json
 ├── tsconfig.node.json
 ├── README.md
 ├── postcss.config.js
 ├── tsconfig.app.json
 ├── index.html
 ├── run_backend.py
 ├── requirements.txt
 ├── package.json
 ├── vite.config.ts
 └── run_frontend.py
```

Quick Start

Prerequisites

- Python 3.10 or higher
- IBM Watsonx AI account and API key
- Node.js and npm (for React frontend)

Installation & Setup

1. **Clone or create the project structure:**
2. mkdir SmartSDLC
3. cd SmartSDLC
4. **Set up the backend environment:**
5. python -m venv venv
6.
7. # On Windows

8. venv\Scripts\activate
9.
10. # On macOS/Linux
11. source venv/bin/activate
12. **Install Python dependencies:**
13. pip install -r requirements.txt
14. **Install Node.js dependencies:**
15. npm install
16. **Configure environment variables:**
    - Update the .env file with your IBM Watsonx credentials
    - Ensure all required API keys are properly set
17. **Start the application:**
18. # Start backend
19. python run_backend.py
20.
21. # Start frontend (in a new terminal)
22. python run_frontend.py

Alternative Manual Setup

1. **Start the backend server:**
2. uvicorn main:app --host 0.0.0.0 --port 8000 --reload
3. **Start the frontend (Streamlit):**
4. streamlit run Home.py --server.port 8501 --server.address 0.0.0.0
5. **Start the React frontend (if using):**
6. npm run dev

Accessing the Application

Once all services are running: • **Streamlit Frontend:** http://localhost:8501
• **React Frontend:** http://localhost:5173 (if using Vite)
• **Backend API:** http://localhost:8000
• **API Documentation:** http://localhost:8000/docs

Authentication

The application includes a built-in authentication system:

1. **Register:** Create a new account with username, email, and password
2. **Login:** Access the application with your credentials
3. **Secure Sessions:** JWT-based authentication for API calls

Using SmartSDLC

1. Upload & Classify Requirements

• Navigate to the "Upload & Classify" section
• Upload a PDF document with requirements
• View classified requirements and generated user stories

2. Generate Code

• Go to "Code Generator"
• Enter a natural language description of what you want to build
• Select programming language
• Get production-ready code instantly

3. Fix Bugs

• Use "Bug Fixer" to upload buggy code
• Get fixed code with explanations
• Compare original vs. fixed versions

4. Generate Tests

• Input your code in "Test Generator"
• Receive comprehensive unit tests
• Download test files for your project

5. Summarize Code

• Paste code in "Code Summarizer"
• Get human-readable documentation
• Perfect for code reviews and documentation

6. AI Assistant

• Use the floating AI assistant for real-time help
• Ask questions about SDLC, coding, or development
• Get contextual guidance and best practices

Configuration

Variable	**Description**	**Default**
WATSONX_API_KEY	IBM Watsonx API Key	Required
WATSONX_PROJECT_ID	IBM Watsonx Project ID	Required
WATSONX_URL	IBM Watsonx Service URL	https://eu-de.ml.cloud.ibm.com
SECRET_KEY	JWT Secret Key	smartsdlc-secret-key-2024
API_HOST	Backend Host	0.0.0.0
API_PORT	Backend Port	8000
STREAMLIT_HOST	Frontend Host	0.0.0.0
STREAMLIT_PORT	Frontend Port	8501
FASTAPI_BASE_URL	Backend URL for Frontend	http://localhost:8000

IBM Watsonx Setup

1. Create an IBM Cloud account
2. Set up Watsonx AI service
3. Generate API key and project ID
4. Update the .env file with your credentials

Troubleshooting

Common Issues

1. **Backend Connection Error**
   - Ensure FastAPI server is running on port 8000
   - Check if the backend URL in frontend is correct
   - Verify environment variables are set
2. **IBM Watsonx Authentication Error**
   - Verify API key and project ID are correct
   - Check if your IBM Cloud account has Watsonx access
   - Ensure the service URL is correct for your region
3. **File Upload Issues**
   - Only PDF files are supported for requirement analysis
   - Check file size limits (default: 16MB for FastAPI)
   - Ensure PDF is readable and not corrupted
4. **Port Already in Use**

    - Change ports in .env file if 8000 or 8501 are occupied
    - Update FASTAPI_BASE_URL if backend port changes

5. **Node.js/React Issues**
    - Ensure Node.js is installed and up to date
    - Run npm install to install all dependencies
    - Check for TypeScript compilation errors

API Endpoints

Authentication

- **POST** /auth/register - Register new user
- **POST** /auth/login - User login

AI Features

- **POST** /ai/upload-pdf - Upload and classify PDF requirements
- **POST** /ai/generate-code - Generate code from description
- **POST** /ai/fix-bugs - Fix bugs in code
- **POST** /ai/generate-tests - Generate test cases
- **POST** /ai/summarize-code - Summarize and document code

Chat & Feedback

- **POST** /chat/chat - Chat with AI assistant
- **POST** /feedback/submit - Submit feedback
- **GET** /feedback/list - List feedback (admin)

Deployment

Local Development

Use the provided startup scripts (run_backend.py and run_frontend.py) for easy local development.

Production Deployment

For production deployment, consider: • Using a proper database (PostgreSQL, MySQL) instead of in-memory storage
• Setting up proper logging and monitoring
• Using environment-specific configuration
• Implementing rate limiting and security measures
• Using Docker containers for deployment
• Setting up CI/CD pipelines with the existing configuration files

Contributing

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Test thoroughly
5. Submit a pull request

License

This project is licensed under the MIT License - see the LICENSE file for details.

Support

For issues and questions: • Check the troubleshooting section

• Review the API documentation at /docs

• Create an issue in the repository

Future Enhancements

• GitHub integration for automated workflows

• Advanced project management features

• Multi-language support for UI

• Advanced analytics and reporting

• CI/CD pipeline integration

• Docker containerization

• Database persistence

• Advanced user management

• Real-time collaboration features

SmartSDLC - Revolutionizing software development with AI!