SmartSDLC – AI-Enhanced Software Development Lifecycle

Generative AI with IBM

Project Documentation

1.Introduction

- **Project title:**

SmartSDLC – AI-Enhanced Software Development Lifecycle

- **Team members:**

  - ❖ DAFNE.B
  - ❖ JOYCY.G
  - ❖ AARTHI.L
  - ❖ DHILSHA STEPI.M
  - ❖ BADHMA PRIYA.S

2.Project Overview

Purpose:

The purpose of SmartSDLC is to modernize the traditional Software Development Lifecycle by embedding Artificial Intelligence into each phase—planning, analysis, design, coding, testing, deployment, and maintenance. AI-driven insights, automation, and predictive analytics reduce errors, accelerate delivery, and ensure higher software quality. The system acts as a digital project partner—assisting developers, testers, and managers with intelligent recommendations, workflow automation, and real-time monitoring.

Features:

- Conversational AI Assistant
  - Key Point: Natural language project interaction

    - Functionality: Allows stakeholders to ask queries about project status, deadlines, bugs, and requirements in plain language.

- Automated Requirement Analysis
    - Key Point: NLP-driven requirement gathering
    - Functionality: Extracts, analyzes, and validates requirements from documents or user input.
- AI-driven Code Review
    - Key Point: Intelligent quality checks
    - Functionality: Reviews code for bugs, security flaws, and best practices.
- Test Case Generation
    - Key Point: Automated testing
    - Functionality: Creates and executes test cases based on project requirements and code changes.
- Effort & Risk Prediction
    - Key Point: Project planning support
    - Functionality: Predicts project effort, cost, timeline risks using historical and real-time project data.
- Continuous Monitoring & Feedback
    - Key Point: Post-deployment intelligence
    - Functionality: Tracks performance, user feedback, and suggests improvements.

3. Architecture

- **Frontend (Streamlit/Gradio):** Interactive dashboards, project timelines, bug reports, and AI chat assistant.

- **Backend (FastAPI):** Handles lifecycle workflows, data storage, and AI model orchestration.
- **LLM Integration (Watsonx / OpenAI):** Provides NLP support for requirement analysis, code review, and documentation.
- **Vector Search (Pinecone / FAISS):** Stores project documents and allows semantic search.
- **ML Modules:** Predictive models for project risk analysis, testing automation, and defect detection.

4. Setup Instructions

(Similar structure: Python, APIs, environment setup, etc.)

5. Folder Structure

- app/ – Backend logic (lifecycle APIs, code analysis, testing modules)
- ui/ – Frontend dashboards for monitoring and interaction
- ai_models/ – AI modules for NLP, code review, and testing
- project_forecaster.py – Predicts risks, effort, and deadlines
- bug_detector.py – Flags coding issues and vulnerabilities
- report_generator.py – Generates AI-driven project reports

6. Running the Application

- Start backend server with FastAPI
- Run dashboard with Streamlit
- Upload project documents/code for AI review
- Interact with the lifecycle assistant

7. API Documentation

Examples:

- POST /analyze-requirements – Extracts requirements from docs
- POST /review-code – Returns AI-based code review
- GET /generate-tests – Auto-creates test cases
- POST /predict-risk – Forecasts project risks

8. Authentication

Role-based access: Admin, Developer, Tester, Manager.

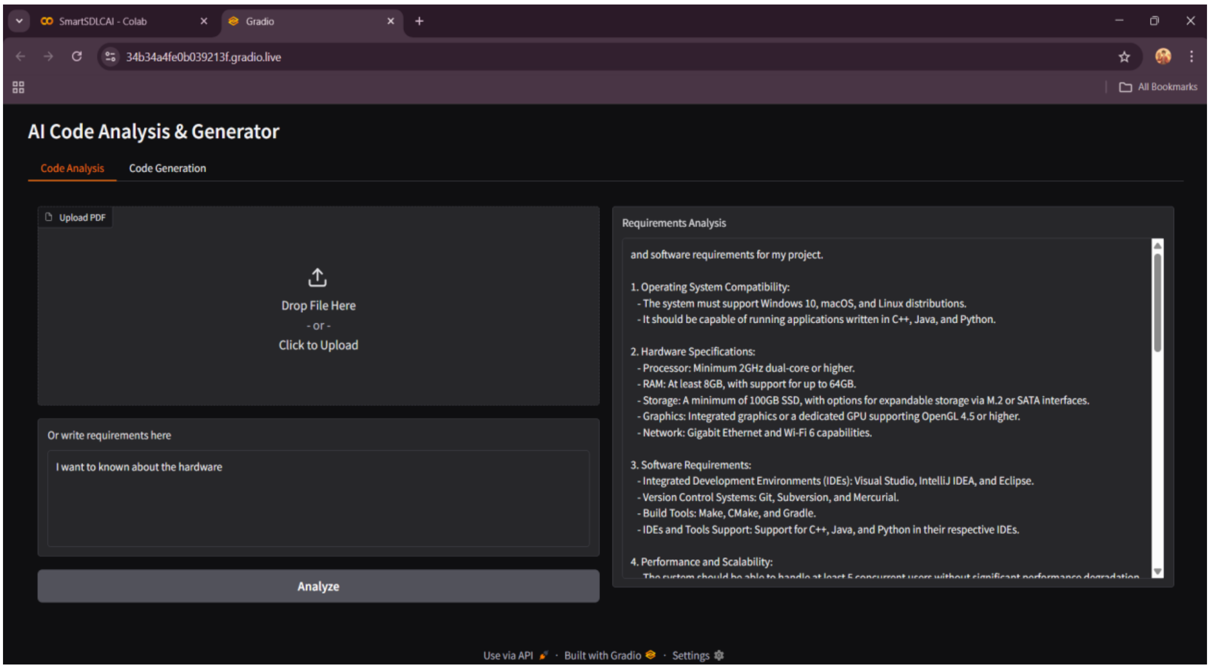
9. User Interface

- Sidebar navigation (Requirements, Code Review, Testing, Reports)
- Real-time dashboards for progress & bug tracking
- AI-powered chat assistant for project queries

10. Testing

- Unit testing for AI models
- API testing with Postman/Swagger
- Automated regression testing for updates

11. Screenshots



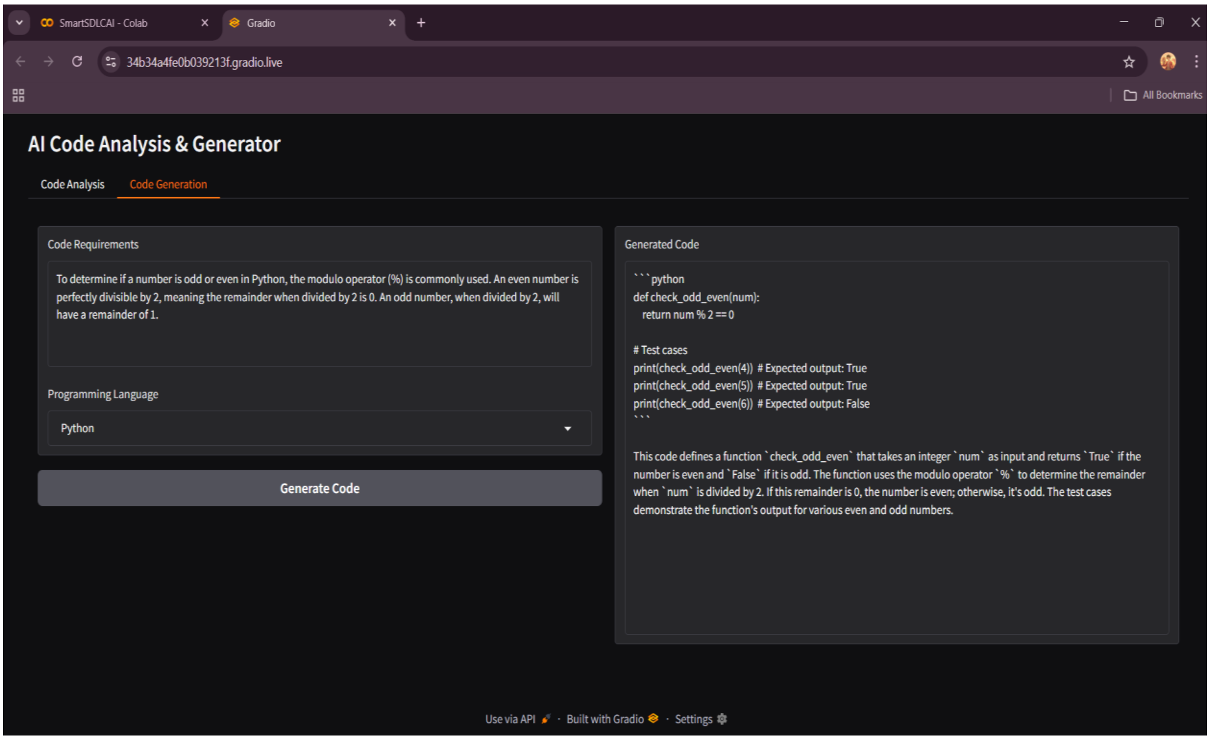
Or write requirements here

I want to known about the hardware

Requirements Analysis

```
and software requirements for my project.

1. Operating System Compatibility:
  - The system must support Windows 10, macOS, and Linux distributions.
  - It should be capable of running applications written in C++, Java, and Python.

2. Hardware Specifications:
  - Processor: Minimum 2GHz dual-core or higher.
  - RAM: At least 8GB, with support for up to 64GB.
  - Storage: A minimum of 100GB SSD, with options for expandable storage via M.2 or SATA interfaces.
  - Graphics: Integrated graphics or a dedicated GPU supporting OpenGL 4.5 or higher.
  - Network: Gigabit Ethernet and Wi-Fi 6 capabilities.

3. Software Requirements:
  - Integrated Development Environments (IDEs): Visual Studio, IntelliJ IDEA, and Eclipse.
  - Version Control Systems: Git, Subversion, and Mercurial.
  - Build Tools: Make, CMake, and Gradle.
  - IDEs and Tools Support: Support for C++, Java, and Python in their respective IDEs.

4. Performance and Scalability:
  - The system should be able to handle at least 5 concurrent users without significant performance degradation
```

We can give any requirements and it will analyze it and give the result.


SmartSDLCAI - Colab
Gradio
34b34a4fe0b039213f.gradio.live
All Bookmarks
AI Code Analysis & Generator
Code Analysis
Code Generation
Generate Code
Use via API · Built with Gradio · Settings

Code Requirements

To determine if a number is odd or even in Python, the modulo operator (%) is commonly used. An even number is perfectly divisible by 2, meaning the remainder when divided by 2 is 0. An odd number, when divided by 2, will have a remainder of 1.

Programming Language: Python

Generated Code

~~~
```python
def check_odd_even(num):
  return num % 2 == 0

# Test cases
print(check_odd_even(4))  # Expected output: True
print(check_odd_even(5))  # Expected output: True
print(check_odd_even(6))  # Expected output: False
```

This code defines a function `check_odd_even` that takes an integer `num` as input and returns `True` if the number is even and `False` if it is odd. The function uses the modulo operator `%` to determine the remainder when `num` is divided by 2. If this remainder is 0, the number is even; otherwise, it's odd. The test cases demonstrate the function's output for various even and odd numbers.
~~~

Now we can generator code we want and we can generate the code in python and it analyze and give the output.

12. Known Issues

- Accuracy depends on dataset quality
- Limited explainability for AI-driven risk predictions

13. Future Enhancements

- Integration with DevOps pipelines (CI/CD)
- Support for multi-language projects
- Enhanced predictive analytics with larger datasets