SmartSDLC – AI-Enhanced Software Development Lifecycle

Project Title:

SmartSDLC Pro – AI Development Suite

Team Name:

LTVIP2025TMID31066

Team Members:

1. Akula Chaitanya Eswar (Team Leader)

1. Akula Chaitanya Eswar (Team Leader)

Phase-1: Brainstorming & Ideation

Objective:

Automate and streamline the Software Development Life Cycle (SDLC) using AI.

Key Points:

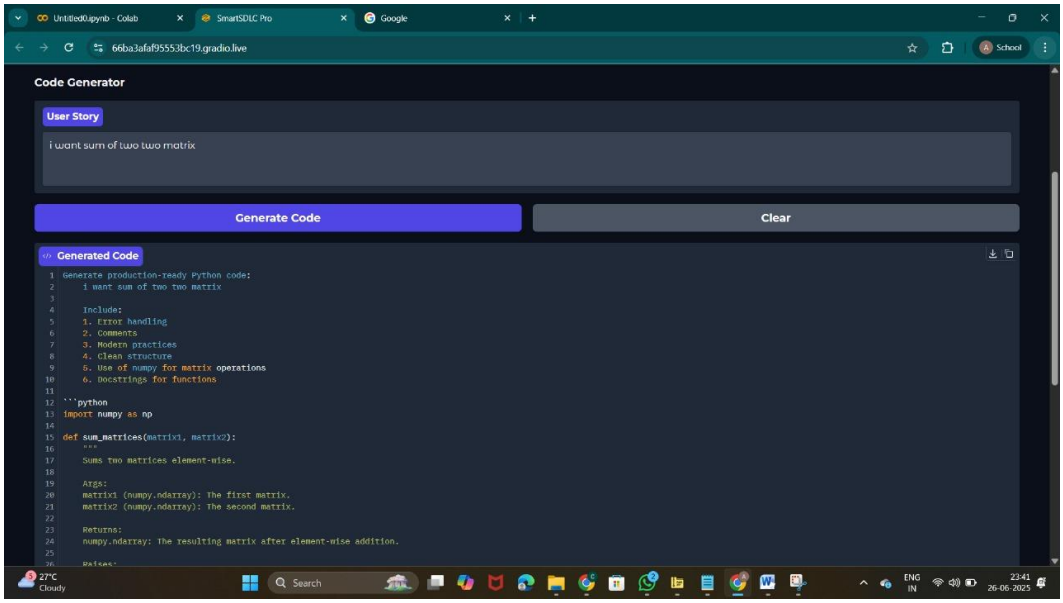
1. Problem Statement:
Software development is time-consuming, fragmented, and prone to errors due to manual tasks like requirement analysis, coding, debugging, and documentation. Developers lack a unified AI-powered tool that supports multi-language workflows seamlessly.

2. Proposed Solution:
An AI-enhanced development suite, SmartSDLC Pro, integrates IBM's Granite 3.3-2B model and Gradio-based UI to automate SDLC tasks like requirements analysis, code generation, debugging, testing, documentation, and architectural consultation.

3. Target Users:
- Software Developers
- QA Engineers
- Project Managers
- Software Development Teams in startups and enterprises

4. Expected Outcome:
- Faster software development
- Reduced errors

- Improved code quality
- Centralized AI-driven SDLC automation



Phase-2: Requirement Analysis

Objective:

Identify and define both technical and functional requirements.

Key Points:

1. Technical Requirements:
- Python
- Gradio, PyTorch, Transformers, PyPDF2
- IBM Granite 3.3-2B Instruct
- GPU (CUDA) or CPU fallback
- JSON handling, Tempfile, OS modules

2. Functional Requirements:
- PDF-based requirement extraction
- Multi-language code generation
- Automated debugging
- Test case generator
- Documentation tool
- AI-based architectural consulting
- Feedback collection module

3. Constraints & Challenges:
- GPU dependency

- Prompt quality affects output
- Model requires significant compute resources
- PDF parsing accuracy depends on formatting

Phase-3: Project Design

Objective:

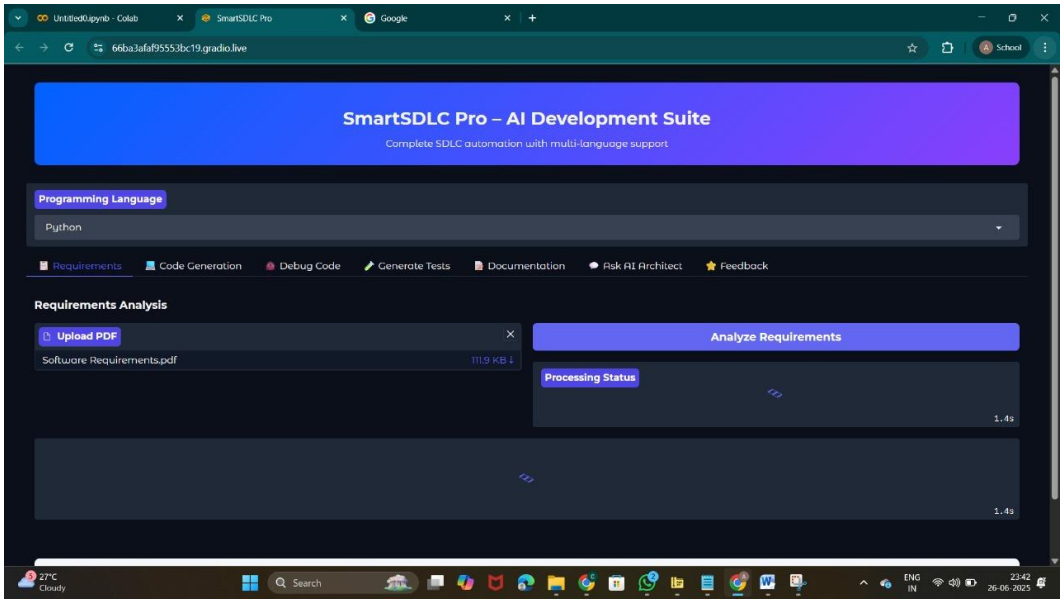
Design system architecture, user flow, and interface.

Key Points:

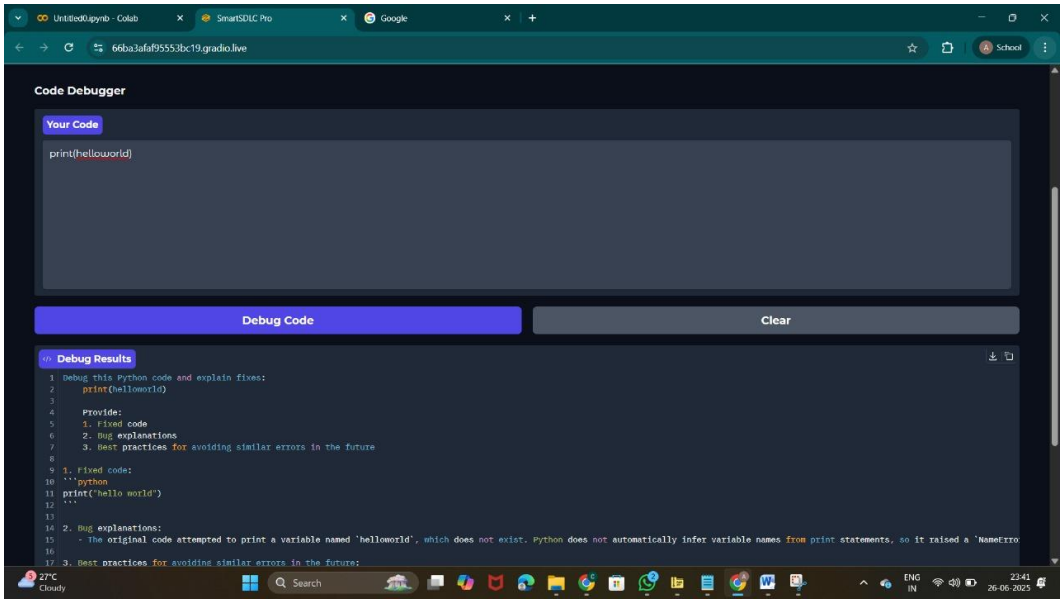
1. System Architecture Diagram:
User Input (PDF/Text) → Tokenizer → IBM Granite LLM → Inference Engine → Response Formatter → Gradio UI (6 Tabs)

2. User Flow:
Upload Requirement → Choose Language → Select Function → AI Processes → Display Output → Review/Edit → AI Consultation → Save/Feedback

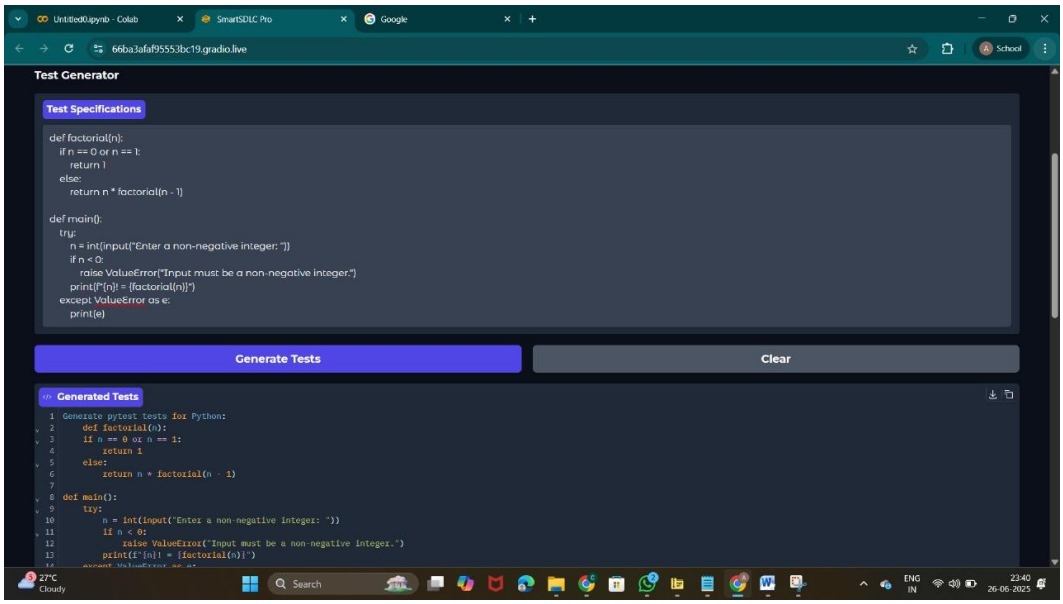
3. UI/UX Considerations:
- Clean Gradio interface
- Tabs for SDLC stages
- Language selector
- Status indicators
- User-friendly input/output editors



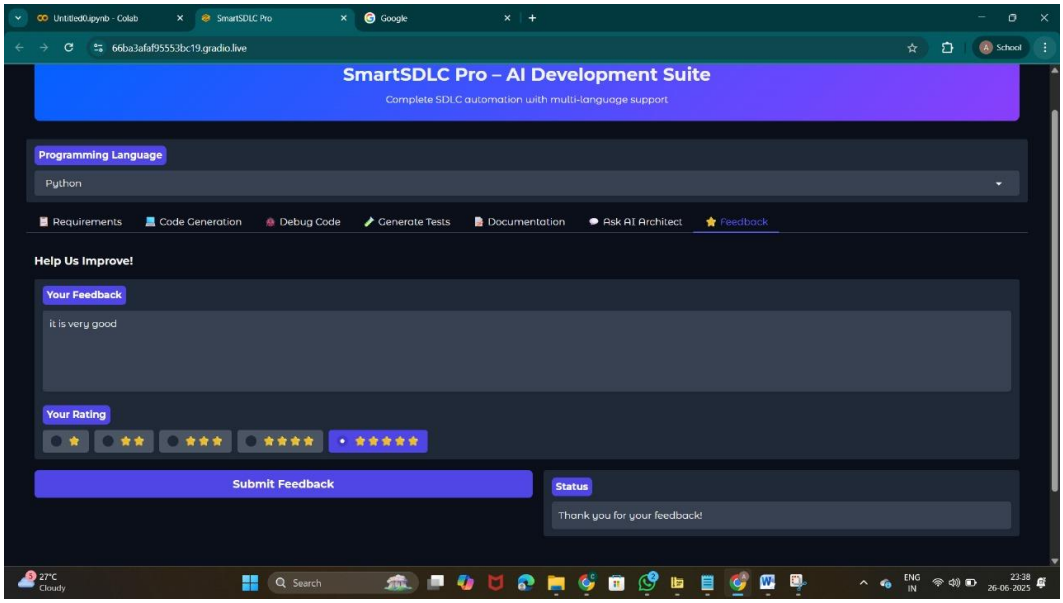
Debug Code Screenshot:



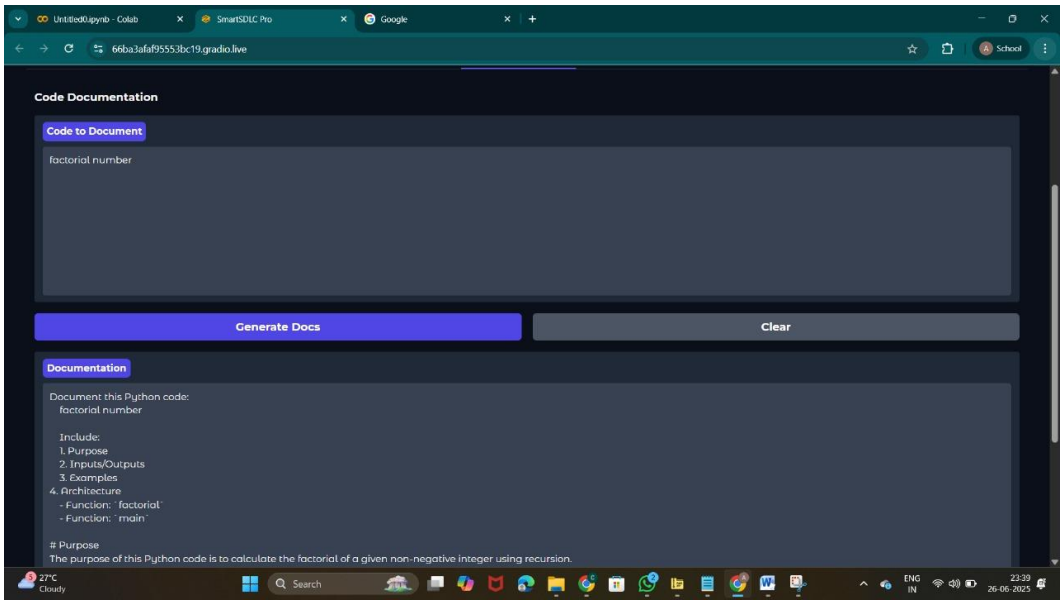
Test Generator Screenshot:



Documentation Screenshot:



Feedback Screenshot:



Phase-4: Project Planning (Agile)

Objective:

Organize development using Agile methodology.

Key Points:

1. Sprint Planning:
- Week 1: Research

- Week 2: UI Development
- Week 3: Model Integration
- Week 4: Feature Development
- Week 5: Testing
- Week 6: Deployment

2. Task Allocation:
- All tasks are managed by Akula Chaitanya Eswar (Backend Development, UI Design, Testing, Deployment)

3. Milestones:
- UI Ready (Week 2)
- Model Integrated (Week 3)
- Features Built (Week 4)
- Tested (Week 5)
- Deployed (Week 6)

Phase-5: Project Development

Objective:

Implement the features and integrate components.

Key Points:

1. Technology Stack:
- Python, Gradio, PyTorch, Transformers, PyPDF2, IBM Granite

2. Process:
- Set up environment
- Build UI with Gradio
- Integrate AI model
- Develop features (code gen, debug, docs, tests)
- Handle feedback and optimize

3. Challenges:
- GPU load → Fixed with token optimizations
- PDF parsing errors → Added validations
- Latency issues → Managed with GPU fallback

Phase-6: Functional & Performance Testing

Objective:

Verify and validate functional and performance aspects.

Key Points:

1. Test Cases:
- PDF parsing
- Code generation accuracy
- Debugging accuracy
- Test generation
- Documentation generation

2. Fixes:
- Improved error handling
- Managed token overflow
- UI stability enhancements

3. Deployment:
- Gradio share link
- GitHub repository

Conclusion:

SmartSDLC Pro proves AI's power in streamlining SDLC processes, enhancing productivity, and reducing errors with a single integrated tool.

Future Scope:

- GitHub integration for commits
- Real-time code collaboration
- Diagram generation (UML, DFD)
- Docker-based deployment
- AI chat memory for long discussions

Advantages and Disadvantages:

Advantages:

- Complete SDLC automation within a single tool
- Supports over 10 programming languages
- Provides explanations in addition to code
- Reduces manual effort and improves developer productivity
- User-friendly modular interface
- Fast inference with GPU support

Disadvantages:

- Relies on GPU for best performance
- Prompt quality impacts output significantly

- Costly inference for larger-scale applications
- PDF parsing accuracy can be affected by poor formatting

Demo Links:

GitHub Repository: https://github.com/ChaitanyaEswar/SmartSDLC-Pro

Video Demonstration: https://youtu.be/SmartSDLC_Demo_Link