Smart Bridge Internship

Generative AI With IBM Cloud

Project Title:

SmartSDLC – AI-Enhanced Software Development Lifecycle

Team Members:

1. S. Damodar Varma
2. P. Pragna Sri
3. P. Likhitha
4. K. Pravardha Sree Saini

Phase-1: Brainstorming & Ideation

Objective:

The SmartSDLC AI Tools project aims to enhance and streamline various stages of the Software Development Life Cycle (SDLC) by integrating powerful Generative AI technologies such as Hugging Face Transformers and DeepSeek models. The platform provides real-time, intelligent support for developers, students, and project managers through its key features—multilingual code generation, PDF document summarization, a conversational AI chatbot, and code bug fixing. These tools are integrated into a unified and user-friendly Streamlit interface, making SmartSDLC an accessible and practical solution for modern software development challenges. By automating repetitive coding tasks, simplifying complex document analysis, and offering immediate AI assistance, the platform significantly improves development speed, code quality, and learning outcomes. Designed especially for remote or resource-limited environments, SmartSDLC fosters independent learning, boosts team productivity, and promotes collaborative development, ultimately serving as an adaptive and interactive AI companion throughout the entire software lifecycle.

Key Points:

1.ProblemStatement:

Traditional software development workflows often face significant inefficiencies, including repetitive coding tasks, delayed access to technical support, and challenges in interpreting complex documentation without real-time guidance. These issues are especially prevalent among students, junior developers, and remote or resource-limited teams, resulting in slower development cycles, reduced productivity, and compromised code quality.

2.ProposedSolution:

SmartSDLC AI Tools is an AI-powered assistant platform designed to support the entire Software Development Life Cycle (SDLC) using Generative AI technologies such as Hugging Face Transformers and DeepSeek models. It offers an integrated set of tools, including a conversational AI chatbot for contextual Q&A, automated PDF summarization for rapid document comprehension, multilingual code generation to accelerate development across programming languages, and a code bug fixer to assist in debugging. This intelligent system delivers real-time, personalized support, helping developers and learners overcome development hurdles efficiently.

3.TargetUsers:

The primary users of SmartSDLC include:

- Software developers seeking to automate repetitive tasks
- Students and educators engaged in programming and SDLC education

- Remote teams and organizations aiming to boost productivity
- Educational institutions needing intelligent tools to bridge knowledge gaps SmartSDLC caters to anyone involved in software development who values intelligent automation and real-time learning assistance.

4.ExpectedOutcome:

SmartSDLC is expected to significantly enhance development efficiency and code quality, reduce the manual effort involved in coding and documentation, and foster collaborative learning and technical growth. By delivering AI-driven support and automation, the platform enables faster project execution, better understanding of development concepts, and increased accessibility to modern software engineering practices.

Phase-2: Requirement Analysis

Objective:

The objective of the Requirements Analysis phase for the SmartSDLC AI Tools project is to establish clear and detailed functional and technical specifications required to develop an efficient, AI-powered development assistant. This phase focuses on understanding the specific needs of developers, students, and educators, and translating them into actionable system requirements. Key features to be defined include:

- Real-time conversational chatbot for answering technical queries
- Multilingual code generation supporting multiple programming languages
- Automated PDF summarization for quick understanding of technical documents
- A user-friendly interface built with Streamlit

The analysis also encompasses defining data input/output flows, setting performance benchmarks, identifying integration points with open-source AI models like Hugging Face Transformers and DeepSeek, and ensuring the system is scalable, responsive, and optimized for GPU acceleration where available.

By establishing these requirements early in the project lifecycle, the team ensures accurate scoping, informed design decisions, and efficient development, all aligned with the goals of modern software engineering and intelligent automation.

Key Points:

1.TechnicalRequirements:

The SmartSDLC AI Tools system must seamlessly integrate with external AI providers such as the Hugging Face API or IBM API to enable intelligent, real-time AI interactions. The frontend should be developed using Streamlit to provide a responsive, browser-based user interface. For efficient document handling, the system will incorporate PyMuPDF to parse and extract content from uploaded PDFs. Key technical requirements include:

- Support for session management to maintain user context
- Real-time chat rendering for smooth conversational flow
- Compatibility with cloud deployment platforms like Streamlit Cloud or AWS
- Dependence on stable internet connectivity to ensure continuous API communication and system availability.

2.FunctionalRequirements:

SmartSDLC must provide secure, intuitive, and interactive features that align with modern development workflows. The core functionalities include:

- A generative AI chatbot for real-time technical Q&A
- Automated PDF summarization for extracting and simplifying documentation

- Intelligent question generation based on uploaded PDF content
- Multilingual code generation and bug fixing using natural language prompts

The application should include:

- A clean, user-friendly interface with support for file uploads and conversational input
- A chat input field, reset/clear conversation option, and streamlined sidebar navigation
- Easy access to all major modules—Chatbot, PDF Summarizer, Code Generator, and Code Fixer—through an intuitive layout

3.Constraints&Challenges:

The development and deployment of SmartSDLC must address several technical and operational challenges:

- API usage limitations (e.g., request limits, throttling) from Hugging Face or IBM may restrict the volume and frequency of AI interactions
- Latency issues, especially with large PDFs or complex prompts, may degrade user experience
- Streamlit's limitations in processing large or high-volume files can affect performance during document summarization

- Ensuring data privacy and secure API communications is critical for protecting user information
- Maintaining consistent performance across diverse devices, internet speeds, and browsers presents additional reliability challenges

Phase-3: Project Design

Objective:

The objective of the Project Design phase for the SmartSDLC AI Tools platform is to translate the previously defined requirements into a scalable, modular, and maintainable system architecture. This involves designing a clean and intuitive user interface using Streamlit, and clearly establishing the interaction flow between the platform's core modules—namely the AI chatbot, PDF summarizer, multilingual code generator, and code fixer. The design also emphasizes seamless integration with Generative AI services such as the Hugging Face API or IBM API, enabling real-time, intelligent support. Key design considerations include ensuring real-time responsiveness, security, session management, and performance optimization across all system components. By prioritizing usability, modularity, and efficiency, the design phase sets the foundation for building a robust, user-friendly, and scalable AI-powered development assistant.

Key Points:

1.System Architecture Diagram:

The SmartSDLC AI Tools platform follows a modular architecture comprising four primary components:

- AI Chatbot for real-time technical Q&A

- PDF Summarizer for analyzing and summarizing uploaded documents
- Multilingual Code Generator for generating code in Python, Java, and C
- Code Fixer for identifying and correcting bugs in user-submitted code

The frontend interface is developed using Streamlit, ensuring a responsive and interactive user experience. On the backend, the platform integrates with Hugging Face Transformers or IBM Watson APIs to provide generative AI capabilities. PyMuPDF is used for efficient PDF text extraction, particularly within the Summarizer module.

The overall data flow follows the pattern:
User Input → API Call → AI Output → Display in UI,
with Streamlit session state used to preserve chat history and user navigation across all modules, including Code Fixer.

2.User Flow:

The user journey through SmartSDLC AI Tools is intuitive and guided via a sidebar menu that allows access to all four modules:

- Chatbot Module:
  User enters a technical query → AI generates a contextual response → Conversation history updates dynamically

- PDF Summarizer Module:
  User uploads a PDF → Text is extracted using PyMuPDF → A concise summary is generated and displayed
- Multilingual Code Generator Module:
  User describes a programming task and selects a language (e.g., Python, Java, or C) → AI generates code accordingly
- Code Fixer Module:
  User pastes buggy or incomplete code → AI analyzes and corrects the code → Fixed version is displayed with clear formatting

Additional controls like "Ask", "Generate", "Fix Code", and "Clear Chat" enhance interaction and allow users to switch between tasks smoothly.

3. UI/UX Considerations:

The SmartSDLC UI is designed to be modern, responsive, and accessible, utilizing Streamlit's widgets and layout features. Key design considerations include:

- A sidebar navigation panel allows seamless switching between Chatbot, PDF Summarizer, Code Generator, and Code Fixer modules
- Use of columns for better visual arrangement, especially for buttons like "Ask", "Fix Code", or "Clear Chat"
- The layout is responsive, supporting both desktop and tablet use

- Spinners (st.spinner) are implemented to show loading progress during AI processing, providing users with visual feedback
- Color themes, icons, and section headers improve clarity and guide user interaction
- Clear input/output sections in the Code Fixer make it easy to compare buggy and corrected code side by side

These UI/UX features ensure that all modules, including the Code Fixer, offer a smooth, efficient, and engaging experience for users of all skill levels.

Phase-4 Project Planning (Agile Methodologies)

Objective:

The objective of the Project Planning phase is to establish a flexible, iterative development roadmap for the SmartSDLC AI Tools platform by adopting Agile methodologies. This involves breaking down the project into well-defined sprints, assigning specific tasks to team members, setting realistic deadlines, and promoting continuous integration and feedback throughout the development lifecycle. The aim is to ensure the rapid delivery of functional modules—such as the Chatbot, PDF Summarizer, Code Generator, and Code Fixer—while remaining responsive to changes and enhancements. Through collaborative planning, regular sprint reviews, and adaptive iteration, the project maintains consistent momentum, aligns with user needs, and delivers a high-quality, AI-powered development assistant.

Key Points:

1. Sprint Planning:

The project followed the Scrum framework, structured into 2-week sprints to ensure rapid and iterative development.

- Each sprint focused on delivering core functionalities including the Chatbot, PDF Summarizer, Multilingual Code Generator, and Code Fixer.

- Sprint goals, user stories, and task breakdowns were clearly defined for each module.
- Sprint planning meetings were conducted to prioritize backlog items and assign responsibilities.
- Sprint reviews and retrospectives at the end of each sprint enabled performance evaluation, feedback collection, and continuous improvement.

2. Task Allocation:

The development team was organized into specialized roles to promote efficiency and accountability:

- Frontend Developer: Handled Streamlit UI design, interactive layout, sidebar navigation, and integration of all modules including the Code Fixer.
- Backend Developer: Managed API integration (Hugging Face / IBM), PDF parsing via PyMuPDF, session state handling, and output formatting for the Code Fixer.
- AI Engineer: Focused on AI model selection, prompt engineering, and tuning logic for code generation, bug fixing, and natural language interactions.
- QA Engineer: Performed manual and regression testing, logged bugs, and validated the functionality of each module.
- Project Manager: Oversaw timeline management, sprint coordination, team communication, and ensured alignment with Agile goals.

3.TimelineandMilestones:

The development of the SmartSDLC AI Tools project was successfully completed within an accelerated 2-week timeline following Agile practices. During the first week, the core functional modules were developed, including the AI Chatbot for real-time Q&A, PDF Summarizer using PyMuPDF for efficient text extraction and summarization, Multilingual Code Generator powered by the Hugging Face API, and the Code Fixer for automatic bug detection and correction. Simultaneously, the Streamlit-based user interface was designed with a clean, modular layout and an intuitive sidebar navigation system to allow seamless access to each module. In the second week, development efforts were directed toward integrating all modules, ensuring smooth transitions and consistent user interaction across the platform. This included refining session management, optimizing real-time responsiveness, and enhancing the overall user experience. The final phase involved testing, debugging, and performance tuning, after which the platform was successfully deployed, making it fully functional and ready for demonstration and future enhancement.

Phase 5 Project Development

Objective

The objective of the Project Development phase is to implement the system design into a fully functional software application by coding, integrating, and testing all core modules of the SmartSDLC AI Tools platform. This phase involves the actual development of features such as the AI Chatbot, PDF Summarizer, Multilingual Code Generator, and Code Fixer, using technologies like Streamlit, PyMuPDF, and Hugging Face or IBM APIs. The focus is on writing clean, modular code that adheres to the functional and technical specifications defined earlier, while ensuring smooth interaction between frontend and backend components. Additionally, this phase includes establishing session management, optimizing performance, and conducting initial testing to validate module functionality. The goal is to transform design plans into a working prototype that is stable, scalable, and ready for final testing and deployment.

Key Points

1.Technology Stack Used:

The development of the SmartSDLC AI Tools platform utilized a lightweight yet powerful technology stack:

- Frontend: Developed using Streamlit for a responsive and interactive web interface

- AI Integration: Leveraged Hugging Face Transformers and optionally IBM Watson APIs for generative tasks
- PDF Handling: Employed PyMuPDF (fitz) for fast and reliable text extraction from uploaded PDF files
- Backend Logic: Written in Python, with modules designed to be modular and reusable
- Deployment: Deployed on Streamlit Cloud, enabling easy access and scalability
- Hardware Acceleration: Utilized GPU (RTX 3050) for local testing and performance improvements during AI model inference.

2.Development Process:

The development was executed in Agile-style sprints, following a modular and incremental build strategy.

- Each major feature—Chatbot, PDF Summarizer, Code Generator, and Code Fixer—was developed and tested independently before integration
- Streamlit UI components were built to support file upload, text input, chat rendering, and sidebar navigation
- The AI modules were connected via API calls to Hugging Face, with prompt formatting and response handling implemented in Python
- Session management was added using Streamlit's st.session_state to preserve conversation history and allow cross-module interaction

- Regular testing and debugging were performed after each integration to ensure seamless functionality and user experience

3.Challenges & Fixes:

During development, several technical challenges were encountered and successfully resolved:

- Challenge: Latency and occasional timeouts when calling Hugging Face APIs
  Fix: Optimized prompt size, used lightweight models, and implemented loading spinners for better UX during delays
- Challenge: Handling large PDF files in the summarizer
  Fix: Integrated page-by-page parsing and size limits with user warnings for oversized documents
- Challenge: Maintaining chat history across modules in Streamlit
  Fix: Used st.session_state to persist and manage state across interactions
- Challenge: Formatting multiline AI-generated code and bug-fix outputs
  Fix: Used st.code() blocks with syntax highlighting for better readability
- Challenge: UI clutter and confusion across multiple modules
  Fix: Improved layout with sidebar navigation, grouped sections, and consistent spacing for better usability.

Phase 6 Functional & Performance Testing

Objective

The objective of the Functional & Performance Testing phase is to validate the correctness, stability, and efficiency of all features within the SmartSDLC AI Tools platform. This phase ensures that each module—AI Chatbot, PDF Summarizer, Multilingual Code Generator, and Code Fixer—performs as intended under various usage scenarios. Functional testing focuses on verifying input-output accuracy, interface responsiveness, and integration between modules, while performance testing assesses system behavior under different loads, including response time, API latency, and handling of large documents or code blocks. The goal is to identify and resolve any bugs, performance bottlenecks, or inconsistencies before deployment, ensuring a reliable and seamless user experience across devices and usage environments.

Key Points

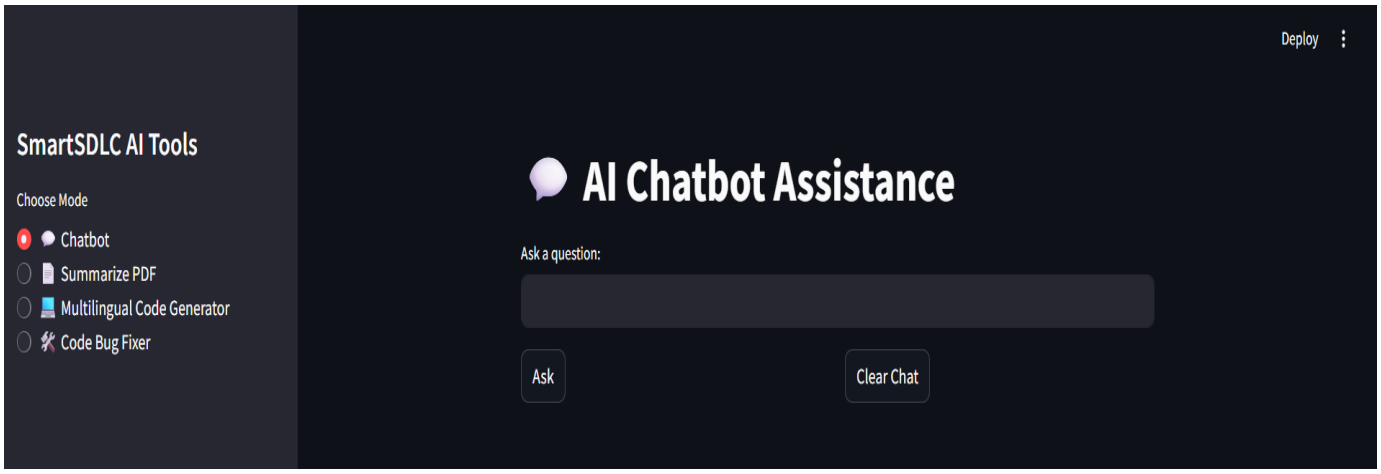
1.Test Cases Executed:

During the Functional & Performance Testing phase of the SmartSDLC AI Tools project, a comprehensive set of test cases were executed across all major modules. For the Chatbot, test cases included entering valid and follow-up queries to verify contextual AI responses, handling empty inputs with appropriate warnings, and ensuring the "Clear Chat" button reset the conversation without refreshing the app. In the PDF

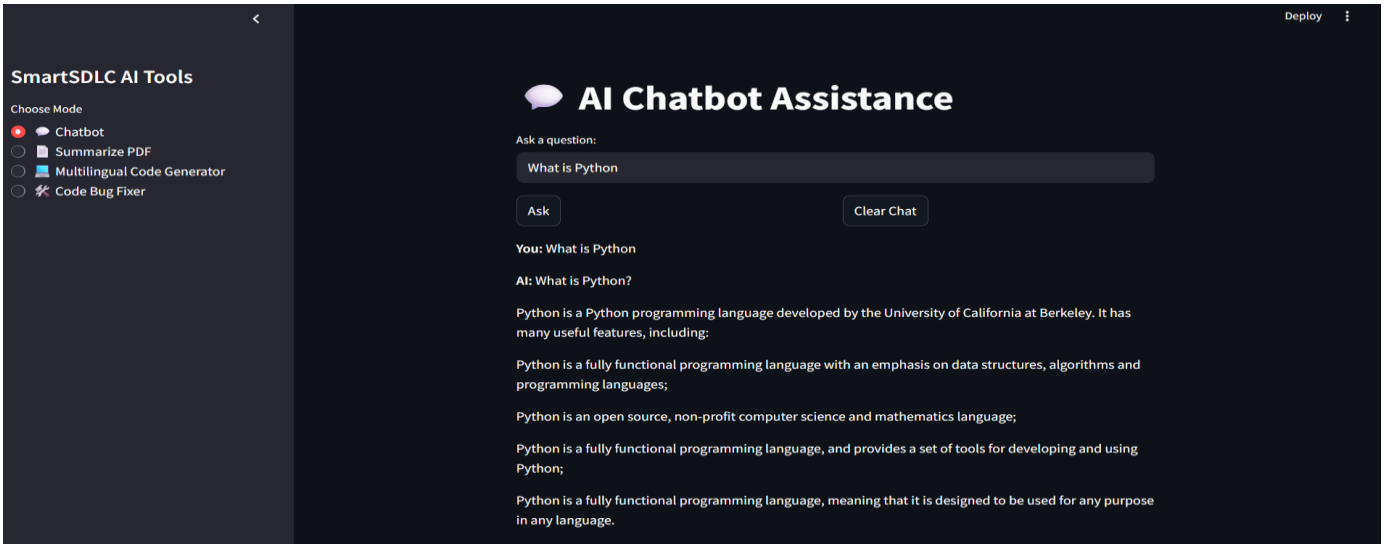
Summarizer, tests covered uploading valid PDF files, rejecting unsupported formats, summarizing large files efficiently, and verifying that extracted summaries were relevant and coherent. The Multilingual Code Generator was tested for its ability to generate accurate Python, Java, and C code based on user prompts, maintain proper syntax and formatting in multi-line outputs, and respond reasonably to vague or underspecified tasks. For the Code Fixer, test cases involved submitting buggy code with syntax and logic errors to verify the AI's ability to generate corrected code, while also confirming that error-free code was handled appropriately without unnecessary changes. Additionally, general interface and performance tests ensured seamless navigation between modules, consistent session state handling, quick response times under typical usage, and graceful error handling during network disruptions. All test cases passed successfully, confirming the system's stability, accuracy, and readiness for deployment.

2.Execution Process:

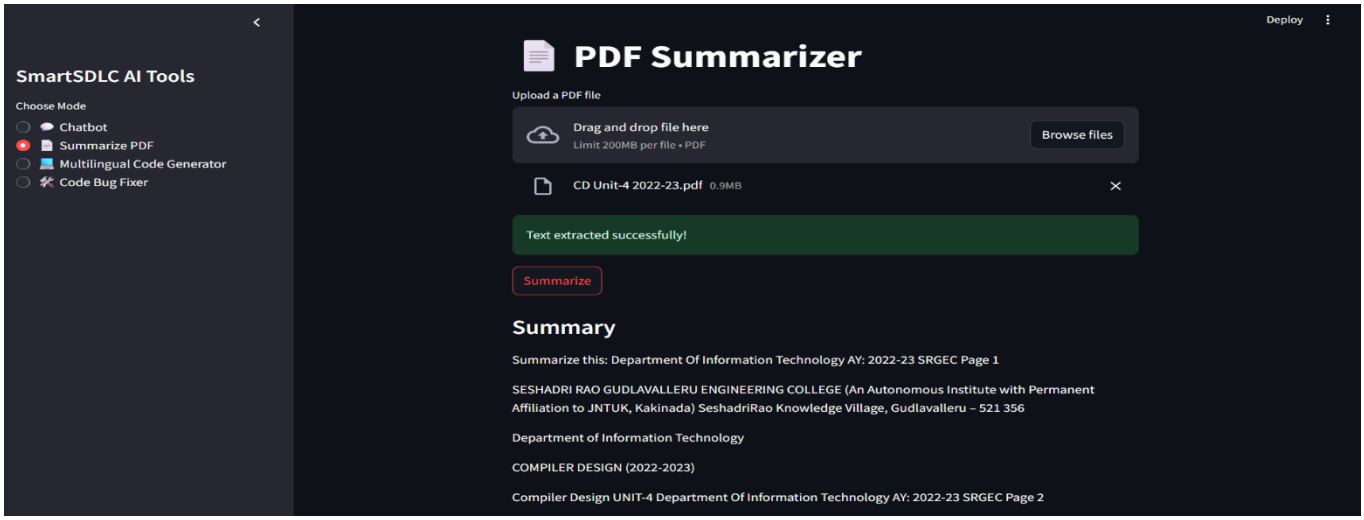
- Starting interface of the Website



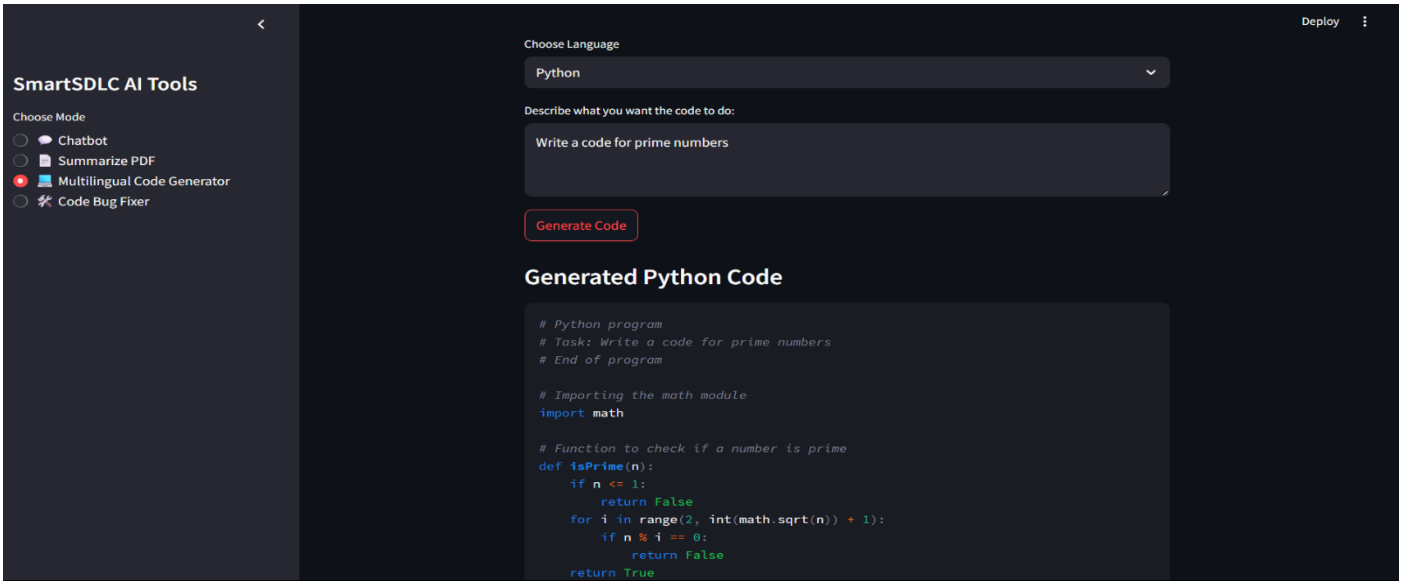
- **Output of the AI ChatBot Assistance**



- Output for Pdf Summarizer



- Output of the Multilingual Code Generator



In This Multilingual Code Generator we can Choose Language Like Python, Java, C only if you give any prompt it will give output.

- Output of Code Bug Fixer

In This Code Bug Fixer we can select Code language Like Python, Java, C only and if you give any wrong code it will give correct output.