Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	27 June 2025
Team ID	LTVIP2025TMID38248
Project Name	SmartSDLC - AI-Enhanced Software Development Lifecycle
Maximum Marks	4 Marks

Technical Architecture:

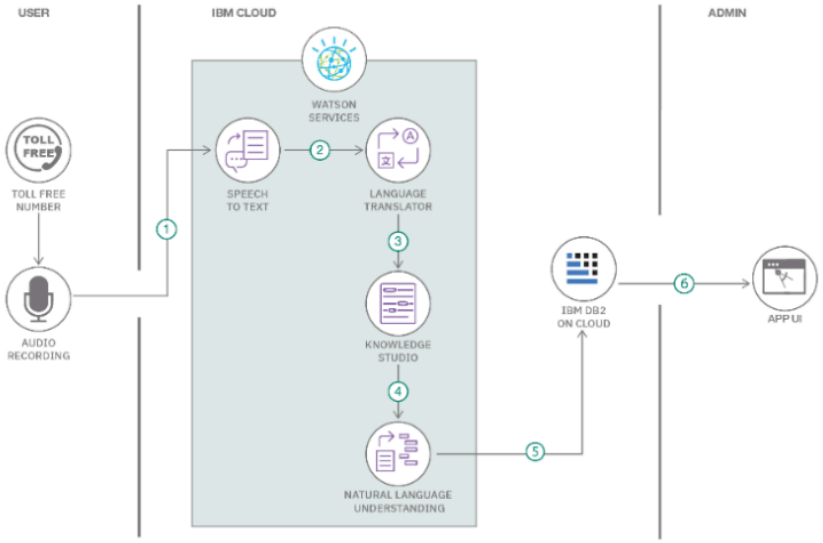

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with the application. Provides a responsive web interface with multiple tabs for different functionalities and a floating chatbot.	HTML, CSS, JavaScript
2.	Application Logic(Backend)	The core logic for handling requests, orchestrating AI model calls, and processing data. Each AI functionality (code gen, bug fix, etc.) is a distinct logical process.	Java / Python
3.	PDF Processing	Component responsible for extracting textual content from uploaded PDF documents.	PyMuPDF
4.	External API	Provides access to the hosted ibm-granite AI model for inference. This is an external service consumed by the backend.	Hugging Face Hub (Model as a Service)

5.	Infrastructure (Local Development)	Application deployment and execution environment for development and demonstration purposes.	Google Colab (Jupyter environment)
6.	Machine Learning Model	The generative AI model used for all intelligent functionalities, including code generation, summarization, classification, and chatbot responses.	ibm-granite/granite-3.3-2b-instruct
7.	Infrastructure (Cloud Deployment - Future)	Target environment for scalable, production-ready deployment.	Kubernetes, Docker
8.	File Storage	Temporary storage for uploaded PDF files during processing. (No persistent database storage is currently implemented for user data).	Local Filesystem (within the Colab environment for processing)

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	FastAPI, Hugging Face Transformers, PyMuPDF
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	HTTPS ,CORS (configured in FastAPI to allow frontend communication), API Token Management (Hugging Face Token used for model access).
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Layered Architecture (Frontend-Backend-AI Model)
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	Deployment on Cloud: Future deployment on cloud platforms with Kubernetes ensures high availability through container orchestration, self-healing capabilities, and load balancing across multiple instances/nodes.
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	FastAPI: High-performance Python web framework. Prompts are designed to be concise to minimize token generation and improve response times.