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

Date	31January 3035
Team ID	LTVIP2025TMID59527
Project Name	**Smart SDLC- AI Enhnced Software Development LifeCycle**
Maximum Marks	4Marks

Technical Architecture

Layer	**Technology / Tools**	**Purpose**
Frontend (UI)	Streamlit (Python-based)	Provides a responsive and interactive UI for all modules
Backend Logic	Python Functions, Streamlit Callbacks	Handles user input processing, module execution, and state management
AI Model Layer	IBM Granite 3.3-2B Instruct (via Hugging Face Transformers)	NLP-based reasoning, code generation, test generation, and prompt response
PDF/Text Parser	PyMuPDF (fitz)	Parses uploaded PDFs and extracts raw requirement text
Test Generator	Prompt-based test generation with AI	Generates unit, integration, and functional test cases

Code Generator	Prompt-based multi-language code synthesis using AI	Generates production-grade code across selected languages and frameworks
AI Assistant Module	Granular prompt handling and chat context management	Offers SDLC guidance, tool tips, and technical advice
Model Serving	Local model inference via PyTorch	Ensures fast and private AI model execution
Deployment Tunnel	Ngrok	Temporary public access to localhost for demo or team use
Session Management	Streamlit session state	Stores intermediate outputs like generated code, test cases, etc.
Output Export	Streamlit download APIs	Allows users to download generated files (code, tests, analysis)

References: 🔗 https://huggingface.co/ibm-granite

🔗 https://c4model.com/

🔗 https://docs.streamlit.io/

🔗 https://www.ibm.com/cloud/architecture

🔗 https://martinfowler.com/articles/practical-test-pyramid.html

Application Characteristics

Characteristic	Description
Architecture Type	Modular Monolithic (Single-page with distinct AI modules)
Execution Environment	Local machine or private server (No internet required for AI model)
Integration	Minimal – loosely coupled modules with shared session state
Security	Local processing ensures high data privacy (no API calls)
Performance	GPU-accelerated (if available) for real-time AI response
Scalability	Modular logic can be containerized or extended with APIs
Portability	Can run on any Python-compatible OS (Windows/Linux/macOS)
Extensibility	Easy to plug in new models, frameworks, or UI modules
Usability	Clean, simple Streamlit interface; low learning curve
Accessibility	Accessible via browser (localhost or ngrok tunnel for external users)