**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 26 june 2025 |
| Team ID | LTVIP2025TMID37102 |
| Project Name | SmartSDLC – AI-enhanced SDLC Automation Platform |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

**Example: Order processing during pandemics for offline mode**

**Reference:** [**https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/**](https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/)

Guidelines:

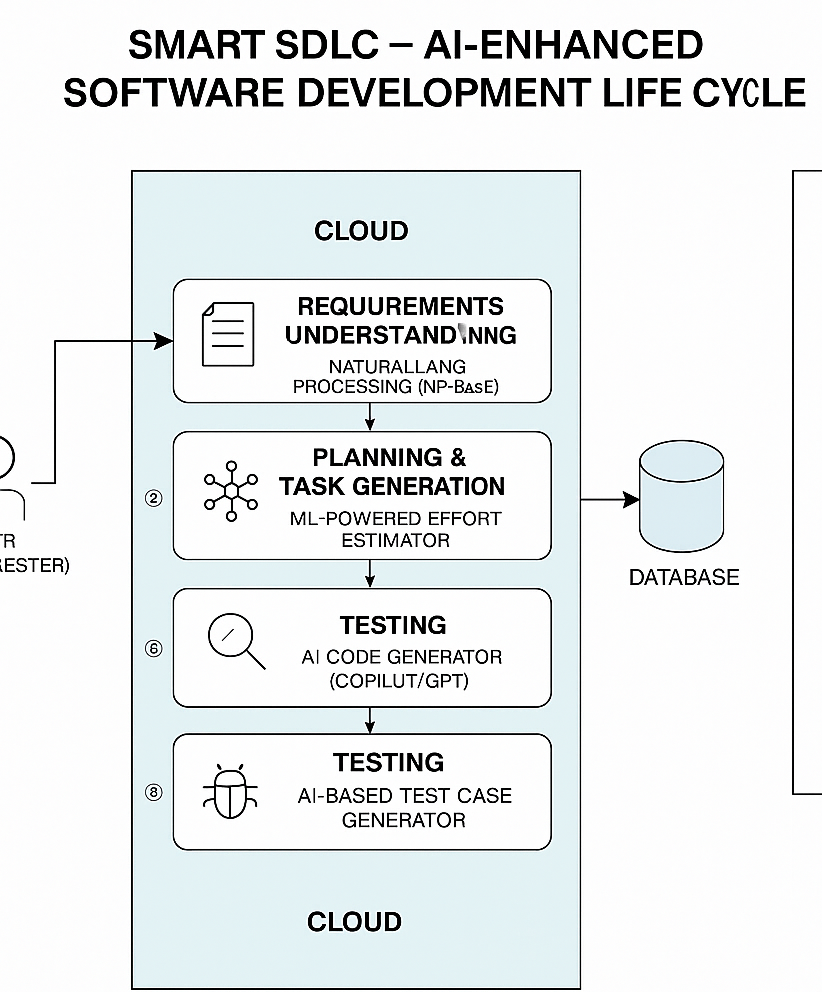
Show all process blocks (e.g., input, prediction, alerts).

Separate local (Gradio) and cloud (IBM Watson) components.

Mark third-party APIs (e.g., Watson, Twilio).

Indicate storage (e.g., IBM DB2, Cloud Object Storage).

Highlight ML model interface if used.

****

**Table-1 : Components & Technologies:**

| **S.No** | **Component** | **Description** | **Technology** |
| --- | --- | --- | --- |
| **1** | **User Interface** | **User interface for uploading requirements, initiating AI tasks, and chatting** | **Gradio (Blocks or ChatInterface), integrated with IBM Watson Assistant UI** |
| **2** | **Application Logic – 1** | **Core backend logic for processing input, managing sessions, and routing tasks** | **Python with FastAPI or Flask** |
| **3** | **Application Logic – 2** | **Manages interaction with LLMs for code generation, test case creation, etc.** | **IBM Watsonx, Hugging Face Transformers via Python SDK** |
| **4** | **Database** | **Stores requirements, generated code, summaries, user logs** | **SQLite (dev) or MySQL/PostgreSQL, with SQLAlchemy in Python** |
| **5** | **File Storage** | **Stores uploaded documents (PDF/Word), generated outputs, and code files** | **Local filesystem or IBM Cloud Object Storage** |
| **6** | **Machine Learning Model** | **Powers NLP tasks like code generation, summarization, and bug detection** | **LLMs from IBM Granite / Hugging Face, integrated via LangChain / Transformers** |
| **7** | **Infrastructure** | **Hosts app backend, UI, and ML services in a scalable manner** | **Dockerized Python App on IBM Cloud (Kubernetes or Cloud Foundry)** |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | List the open-source frameworks used | Technology of Opensource framework |
|  | Security Implementations | List all the security / access controls implemented, use of firewalls etc. | e.g. SHA-256, Encryptions, IAM Controls, OWASP etc. |
|  | Scalable Architecture | Justify the scalability of architecture (3 – tier, Micro-services) | Technology used |
|  | Availability | Justify the availability of application (e.g. use of load balancers, distributed servers etc.) | Technology used |
|  | Performance | Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN’s) etc. | Technology used |

**References:**

[**https://c4model.com/**](https://c4model.com/)

[**https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/**](https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/)

[**https://www.ibm.com/cloud/architecture**](https://www.ibm.com/cloud/architecture)

[**https://aws.amazon.com/architecture**](https://aws.amazon.com/architecture)

[**https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d**](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d)