**Project Design Phase-II**

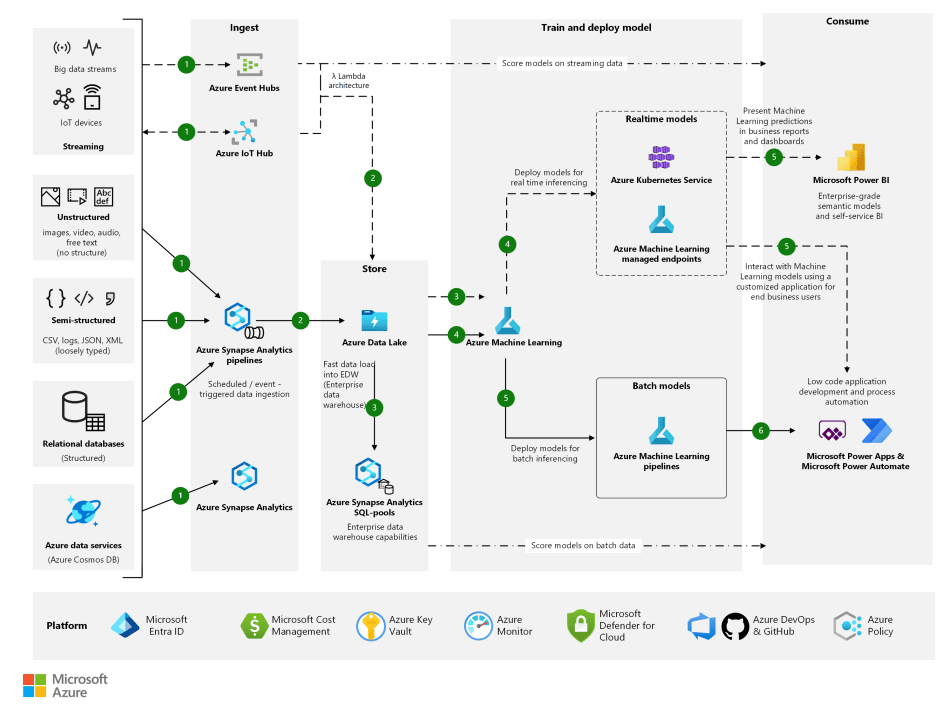
**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 03 August 2025 |
| Team ID | LTVIP2025TMID59168 |
| Project Name | SmartSDLC – AI-Enhanced Software Development Lifecycle |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

The Deliverable shall include the architectural diagram as below

Reference :<https://media.licdn.com/dms/image/v2/D5612AQGoOy9qq1XaLQ/article-cover_image-shrink_720_1280/article-cover_image-shrink_720_1280/0/1718681607278?e=2147483647&v=beta&t=ZaJc-o7l_AmDm7MWsf0Hnlq1rtqUQWVQhGIzCEUq7hg>



|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Component | Description | Technology |
| 1 | User Interface | Interface to interact with SmartSDLC features | Streamlit, HTML, CSS |
| 2 | Application Logic-1 | AI-based requirement classification logic | Python (Scikit-learn, spaCy) |
| 3 | Application Logic-2 | Bug Fixing logic based on LLMs and AST parsing | Python, OpenAI API (GPT models) |
| 4 | Application Logic-3 | Code generation from natural language input | Hugging Face Transformers, Python |
| 5 | Database | Stores user queries, classified requirements, feedback | SQLite / MongoDB |
| 6 | Cloud Database | Store results & feedback securely in the cloud | Firebase / MongoDB Atlas |
| 7 | File Storage | For saving user-uploaded files or temporary code files | Local FileSystem / Firebase Storage |
| 8 | External API-1 | Used for bug-fix suggestions via AI | OpenAI API |
| 9 | External API-2 | Used for code generation logic | Hugging Face Inference API |
| 10 | Machine Learning Model | Classifies requirements, generates code, detects bugs | NLP Classifier, CodeGen Model, LLM APIs |
| 11 | Infrastructure | Deployment of full stack application | Localhost (Streamlit), Ngrok for public URL |

**Table-2: Application Characteristics:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Characteristics | Description | Technology |
| 1 | Open-Source Frameworks | Used for frontend and ML integration | Streamlit, Scikit-learn, Transformers |
| 2 | Security Implementations | Basic protection on API calls, local access controls, data validation | API Key Authentication, Form Validation |
| 3 | Scalable Architecture | Modular architecture for requirement classifier, bug fixer, and code generator | Microservices-style modules, Streamlit Apps |
| 4 | Availability | Can be run locally or made available via ngrok/public endpoints | Ngrok, Cloud Hosting (Firebase optional) |
| 5 | Performance | Optimized AI processing, async responses, local caching for performance | Python Asyncio, LRU Cache, Fast APIs |

References:

* <https://c4model.com/>
* <https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/>
* <https://www.ibm.com/cloud/architecture>
* <https://aws.amazon.com/architecture>
* <https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d>