**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

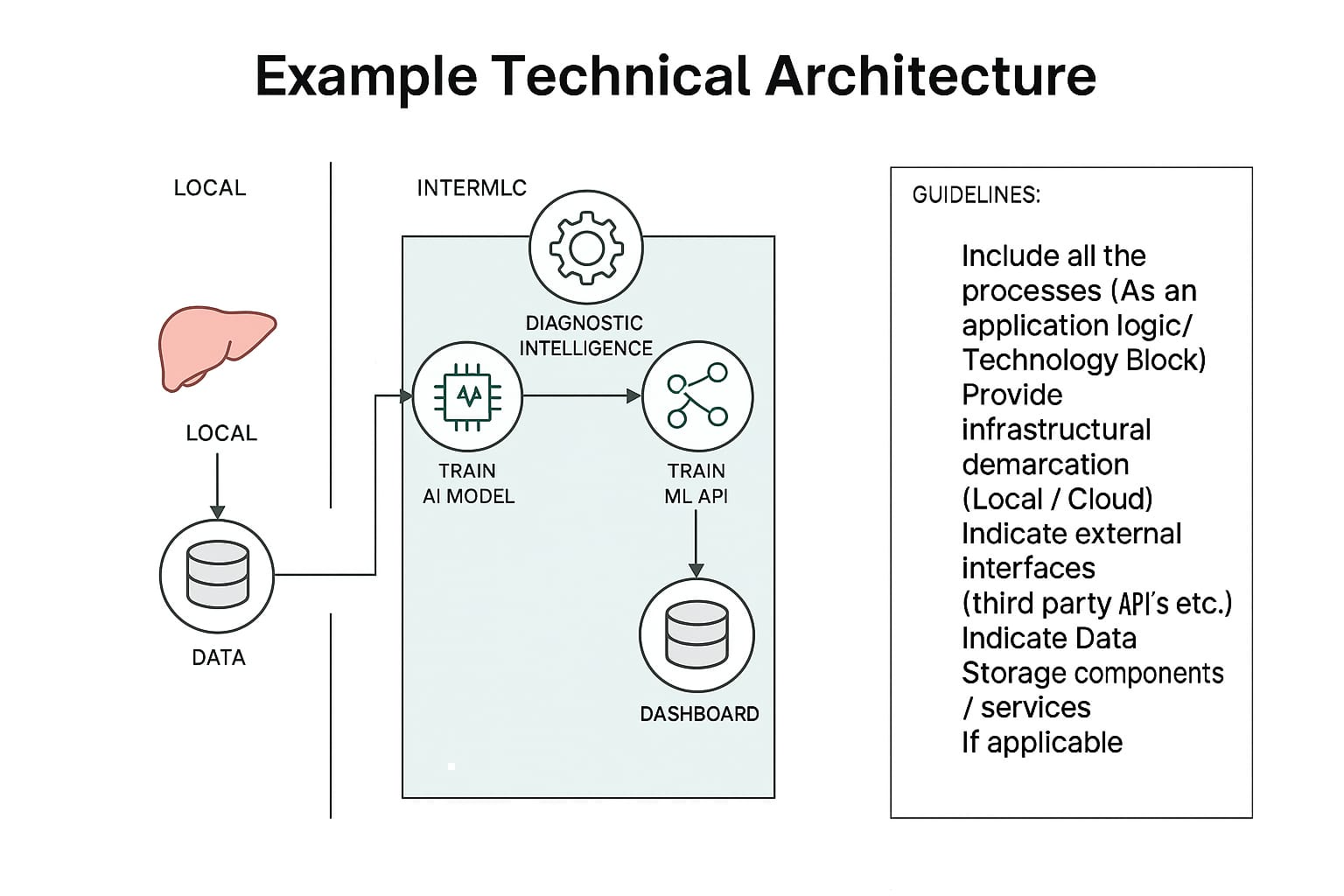
|  |  |
| --- | --- |
| Date | 24 JUNE 2025 |
| Team ID | LTVIP2025TMID33932 |
| Project Name | Revolutionizing Liver Care : Predicting Liver Cirrhosis using Advanced Machine Learning Techniques |
| Maximum Marks | 4 Marks |

**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/)

****

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

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interface | Web-based dashboard for doctors and analysts | . HTML, CSS, JavaScript, React.js |
|  | Application Logic-1 | Data ingestion and preprocessing logic | Python (Pandas, NumPy, Requests) |
|  | Application Logic-2 | ML model training and cirrhosis risk prediction | Python (scikit-learn, XGBoost, TensorFlow/Keras) |
|  | Application Logic-3 | API service to expose model predictions | Flask / FastAPI |
|  | Database | Store structured liver patient data | PostgreSQL / MongoDB |
|  | Cloud Database | Scalable cloud database for patient analytics | AWS RDS / Google Cloud Firestore |
|  | File Storage | Store raw CSV files, reports, and trained model artifacts | AWS S3 / Google Cloud Storage |
|  | External API-1 | |  | | --- | | Access medical terminology or drug info APIs  (if needed) |  |  | | --- | |  | | NIH API / FDA Drug API (optional based on expansion) |
|  | External API-2 | (Optional) Integration with EHR/EMR for patient records | HL7 / FHIR APIs |
|  | Machine Learning Model | Predict liver cirrhosis using clinical data | XGBoost / Random Forest / Logistic Regression |
|  | Infrastructure | Cloud deployment, container orchestration & scaling | Docker, Kubernetes, AWS / GCP / Azure |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | Use of open-source ML and web development tools | scikit-learn, Flask, React, MongoDB |
|  | Security Implementations | Secure APIs, encrypted data flow, and controlled system access | HTTPS, JWT, OAuth2, IAM (AWS/GCP) |
|  | Scalable Architecture | Microservice-based architecture with containerization and auto-scale | Docker, Kubernetes |
|  | Availability | High availability via multi-zone cloud deployment and load balancing | AWS ELB, GCP Load Balancer |
|  | Performance | Fast model inference with caching and optimized delivery | Redis, Flask async, Cloud CDN |

**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)