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

|  |  |
| --- | --- |
| Date | 27 June 2025 |
| Team ID | LTVIP2025TMID34546 |
| Project Name | Pattern sense:Classifying Fabric Patterns using Deep Learning |
| 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:** Automated fabric pattern classification in textile quality control systems to reduce manual errors and labor dependency.

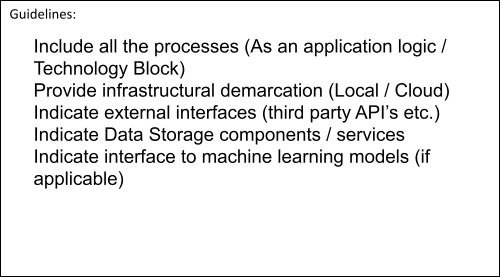
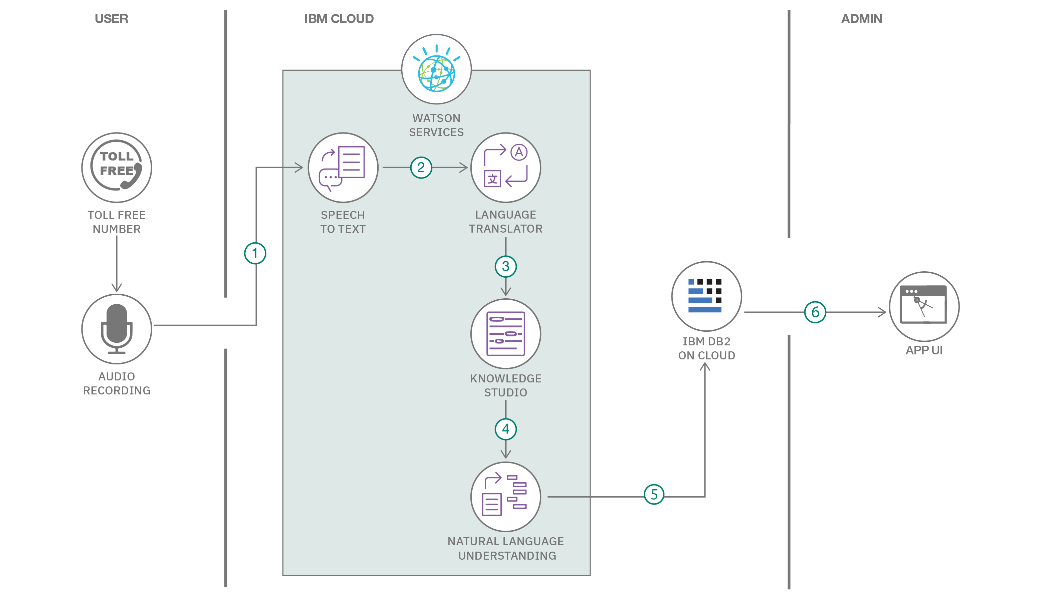


Table-1 : Components & Technologies:

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
| 1. | User Interface | Interface for uploading fabric images and viewing predictions | HTML, CSS, JavaScript ,React Js /etc. |
| 2. | Application Logic-1 | Handles image upload and sends to backend | Python |

|  |  |  |  |
| --- | --- | --- | --- |
| 3. | Application Logic-2 | Handles preprocessing of image (resize,  normalize) | OpenCV, PIL (Python) |
| 4. | Application Logic-3 | Model prediction logic | TensorFlow / PyTorch |
| 5. | Database | Store user upload logs, predictions | MySQL |
| 6. | Cloud Database | Optional cloud DB to store analytics/logs | Firebase Realtime DB / MongoDB Atlas |
| 7. | File Storage | Stores uploaded fabric images | AWS S3 / Local FileSystem |
| 8. | External API-1 | None required (can be NA or future extension like fabric dictionary API) | N/A |
| 9. | External API-2 | (Optional) To fetch real-world fabric names | N/A or Custom AP |
| 10. | Machine Learning Model | Predicts fabric pattern class | CNN (Custom Model with TensorFlow or PyTorch) |
| 11. | Infrastructure (Server) | Deploy model and backend | Localhost, AWS EC2, Google Cloud  Run, Render |

Table-2: Application Characteristics:

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Characteristics** | **Description** | **Technology** |
| 1. | Open-Source Frameworks | Python, TensorFlow, React | TensorFlow, Streamlit, Flask |
| 2. | Security Implementations | Restrict image types, sanitize file names, HTTPS | SSL, MIME checks, JWT auth (optional) |
| 3. | Scalable Architecture | Can deploy on cloud and use containerized services | Docker, Kubernetes (optional) |
| 4. | Availability | Hosted on cloud or local server, high uptime | AWS/GCP with load balancer (optional) |
| 5. | Performance | Image classification in under 5 seconds with GPU acceleration | GPU support (Google Colab / AWS GPU), Redis (optional) |

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>