**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 31 JANUARY 2025 |
| Team ID | LTVIP2025TMID20355 |
| Project Name | PATTERN SENSE |
| Maximum Marks | 4 Marks |

**System Architecture: Pattern Sense**

**🔧 High-Level Architecture Diagram**

sql

CopyEdit

+-----------------------+

| User Interface / |

| Client Application |

+----------+------------+

|

v

+------------------------+ +-----------------------+

| REST API Layer | <-----> | Authentication / |

| (Flask or Django API) | | User Management (opt)|

+----------+-------------+ +-----------------------+

|

v

+----------------------------+

| Inference Engine Layer |

| (TensorFlow Model Serving) |

+----------+-----------------+

|

v

+----------------------------+ +----------------------+

| Preprocessing Module | <------> | Pattern Dataset |

| (Resizing, Normalization) | | (Labeled Images) |

+----------+-----------------+ +----------------------+

|

v

+-----------------------------+

| Output (Prediction + Score)|

+-----------------------------+

**📋 Table 1: Components and Technology Stack**

| **Component** | **Technology / Tools Used** | **Description** |
| --- | --- | --- |
| User Interface (Optional) | HTML/CSS, React, Streamlit | Frontend for uploading fabric images and viewing predictions |
| API Backend | Flask / Django REST Framework | Interface between user and model, handles requests and responses |
| Model Serving | TensorFlow / TensorFlow Lite / TensorFlow Serving | Serves the trained deep learning model for real-time inference |
| Deep Learning Model | CNN / ResNet / EfficientNet (TensorFlow/Keras) | Classifies fabric patterns into predefined categories |
| Data Preprocessing Pipeline | NumPy, OpenCV, TensorFlow Image API | Resizing, normalization, augmentation, and batching |
| Dataset Storage | Local filesystem / Google Drive / AWS S3 | Stores curated and labeled image data |
| Model Training Environment | Google Colab / Jupyter Notebook / Local GPU System | Used for training the deep learning model |
| Version Control | Git / GitHub | Tracks project changes and enables team collaboration |
| Optional Authentication | Firebase Auth / JWT | Manages secure access to the platform (if needed) |

**📌 Table 2: Application Characteristics**

| **Characteristic** | **Description** |
| --- | --- |
| **Platform** | Web-based application with optional local deployment for internal use |
| **Input Type** | Fabric pattern image (JPEG/PNG) |
| **Output Type** | Classified pattern label (e.g., Floral, Stripe) + confidence score (%) |
| **Model Type** | Supervised CNN-based classifier using TensorFlow |
| **Number of Classes** | ~5 to 10 (customizable: Floral, Stripe, Geometric, Polka Dot, etc.) |
| **Inference Time** | ~40–60ms per image (on GPU); ~100–150ms (on CPU) |
| **Scalability** | Horizontally scalable using cloud APIs or TensorFlow Serving |
| **Real-time Inference** | Yes, via REST API layer |
| **Extensibility** | Easily extendable to new pattern types or anomaly detection |
| **Deployment Options** | Local server, Cloud (GCP, AWS), Dockerized container |
| **Security** | Optional auth + HTTPS via API Gateway (for production deployment) |