**Project Design Phase-II**

|  |  |
| --- | --- |
| **Date** | **25 June 2025** |
| **Team ID** | **LTVIP2025TMID24654** |
| **Project Name** | **Shopez : one-stop shop for online purchases** |
| **Mentor Name** | **Dr Shaik Salma Begam** |
| **Maximum Marks** | **4 Marks** |

**Technology Stack (Architecture & Stack)**

**Technical Architecture**

Pattern Sense is a deep learning–based web application designed to classify fabric patterns with high accuracy and scalability. The system allows designers, manufacturers, and quality controllers to upload fabric images and receive precise pattern classification results. It combines a modern web frontend, backend services, a trained deep learning model, and cloud storage for large image datasets.

**Table 1: Technology Stack Components**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology / Service Used** |
| 1 | User Interface | Web UI for users and admins | React.js, HTML5, CSS3, JavaScript |
| 2 | Application Logic-1 | User authentication, registration, session management | Node.js, Express.js, JWT |
| 3 | Application Logic-2 | Fabric image upload, management, classification request handling | Node.js, Express.js |
| 4 | Database | Storage for user data, image metadata, classification results | MongoDB, Mongoose |
| 5 | Cloud Database | (Optional/Scalable) Cloud-hosted MongoDB | MongoDB Atlas |
| 6 | File Storage | Fabric image storage | AWS S3 (preferred), local filesystem |
| 7 | External API-1 | (Optional) Email notifications | SendGrid, Nodemailer |
| 8 | Machine Learning Model | Pattern classification using trained CNN model | Python (Flask or FastAPI), TensorFlow, Keras |
| 9 | Infrastructure | Application deployment, scalability, CI/CD | Docker, AWS EC2, AWS Lambda (model serving), GitHub Actions |
| 10 | Monitoring & Logging | System and application performance monitoring | AWS CloudWatch, ELK stack (Elasticsearch, Logstash, Kibana) |

**Table 2: Application Characteristics**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Characteristics** | **Description** | **Technology / Approach Used** |
| 1 | Open-Source Frameworks | Use of widely adopted frameworks and tools for faster development and community support | React.js, Node.js, Express.js, TensorFlow |
| 2 | Security Implementations | JWT authentication, encrypted storage, secure API endpoints, role-based access | JWT, bcrypt, HTTPS, Helmet, CORS |
| 3 | Scalable Architecture | Modular microservice-ready design, RESTful APIs, containerized deployments, model decoupling | Docker, AWS Lambda for model inference, MongoDB Atlas |
| 4 | Availability | High uptime using cloud infrastructure and auto-scaling | AWS EC2, AWS Elastic Load Balancer, MongoDB Atlas |
| 5 | Performance | Optimized image preprocessing and model inference, efficient API handling, CDN for static assets | TensorFlow optimizations, S3 storage, CloudFront CDN |