Artificial Intelligence And Machine Learning Project Documentation

1. Introduction:

- **Project Title**: Classifying Fabric Patterns

Fabric Pattern Classifier is an AI-powered web application designed to classify fabric patterns such as Stripes, Checks, Floral, and Geometric using convolutional neural networks (CNNs). The system offers a solution to tedious manual classification processes, enabling fashion designers, manufacturers, and e-commerce platforms to automate pattern recognition tasks with high accuracy.

Team Members:

- [Member 1] – Full Stack Developer (Frontend + Backend Integration)
- [Member 2] – AI/ML Developer (Model Training and Prediction)
- [Member 3] – UI/UX Designer (Frontend Components and Layout)
- [Member 4] – Database Engineer (MongoDB Management and Integration)

2. Project Overview:

The aim of this project is to create a smart application that simplifies the identification of fabric patterns. Users can upload fabric images to the system, which processes the image using a CNN model and classifies the pattern. This helps streamline workflows in areas such as:

- Fashion design: Pattern selection made easier.
- Textile manufacturing: Quality control improvement.
- E-commerce: Automatic product tagging.

Key Features:

- Upload and preview fabric images.
- CNN-based AI model for prediction.
- Real-time prediction results.
- Confidence scores for each classification.
- Data storage with MongoDB.
- Intuitive and responsive UI.
- Suggestions for visually similar patterns.

3. Architecture:

Frontend:

* Built with React.js (component-based architecture).

* Styled with Tailwind CSS.

* Uses Axios for API calls.

* Key Components: UploadForm, ResultDisplay, Navbar.

***Backend**:

* Built with Flask in Python.

* Loads pre-trained CNN model (e.g., ResNet50).

* Accepts image uploads, processes them, and returns prediction.

* Connects to MongoDB to store metadata and results.

***Database**:

* MongoDB database with collections like:

  * predictions: stores image data, pattern result, timestamp.

  * feedback: user corrections and notes (optional).

4. Setup Instructions:

***Prerequisites:**

* Node.js (Frontend)

* Python 3.8+ (Backend)

* MongoDB (local or Atlas)

* npm, pip

***Steps:**

1. Clone the repository.

2. In frontend/, run npm install.

3. In backend/, create virtual environment and run pip install -r requirements.txt.

4. Set .env file with MONGO_URI.

5. Start MongoDB service.

6. Run frontend: npm start.

7. Run backend: flask run.

5. Folder Structure:

```
Fabric-Pattern-Classifier/

├──── frontend/

│   ├──── src/components/

│   │   ├──── UploadForm.jsx

│   │   └──── ResultDisplay.jsx

│   └──── App.jsx

├──── backend/

│   ├──── app.py

│   ├──── model/

│   │   └──── model.h5

│   ├──── routes/

│   │   └──── predict_route.py

│   └──── database/

│     └──── mongo_connection.py
```

6. Running the Application:

*Frontend:

Run npm start → Opens at http://localhost:3000

*Backend:

Run flask run → Available at http://localhost:5000

Ensure MongoDB is running before launching the servers.

7. API Documentation:

*POST /predict

* Request Type:multipart/form-data

*Request Body: Image file

*Response Example:

json

```
{
  "prediction": "Floral",
  "confidence": 0.89
}
```

Other endpoints may include:

* /feedback: Submits user correction.

* /history: Fetches past predictions (if authentication is enabled).

8. Authentication (Optional):

***JWT-based Authentication Flow**:

1. User logs in with credentials.

2. Server issues JWT token.

3. Token is passed in headers on each request.

4. Server verifies token before allowing access.

*Advantages:

* Personal history tracking

* Secured access to API

* Role-based permissions (e.g., admin dashboard)

9. User Interface:

The UI is designed to be:

* Clean, minimal, and responsive

* Usable across desktop and mobile devices

***User Flow:**

1. Upload fabric image

2. Submit for classification

3. View prediction and confidence

4. Optionally provide feedback

Visual elements (e.g., preview cards, loading spinners, charts) enhance interactivity.

10. Testing:

***Unit Testing:**

* CNN model loading

* Image preprocessing functions

* MongoDB connection logic

Integration Testing:

* Upload-to-classification flow

* Data saving and retrieval

***Tools Used:**

* PyTest (backend logic)

* Postman (API testing)

* Jest (frontend unit tests)

* MongoDB Compass (data inspection)

11. Demo Link:

[Demo Video](#) - Replace with actual link when available.

12. Known Issues:

* Model may misclassify rare or mixed patterns.

* UI responsiveness might have glitches on older mobile devices.

* File size limit: 5MB (can cause timeout beyond this).

* Currently supports only single-label classification.

13. Future Enhancements:

Live Camera Classification: Use real-time webcam/mobile feed for direct classification.

Multi-Label Model: Classify images with hybrid patterns like “Floral + Stripes.”

Mobile App: Develop a cross-platform app using React Native or Flutter.

Feedback Loop: Collect user corrections to retrain the model over time.

Recommendation Engine: Suggest similar or complementary patterns.

Shopify/Amazon Integration: Automatically tag and list fabrics using this classifier.

AR-Based Preview: Allow users to apply fabrics to garments virtually via AR.

Web Crawlers for Dataset Expansion: Auto-collect tagged images from open sources.

Explainable AI: Use Grad-CAM to show how the model made its decision.

Multilingual UI: Add language support for global usability.

THANK YOU