**Pattern-Sense: Classifying Fabric Patterns Using Deep Learning**

1. **Introduction** 
   * **Project Title:**

Pattern-Sense: Classifying Fabric Patterns Using Deep LearningTeam Member**s:**

**Name Role**

Pechetti Rohit Sri Sai Team Lead / Domain Expert /

Machine Learning Engineer

Pasalapudi v v s s l

Padmavathi Frontend Developer (React)

Mandapalli Mounika Backend Developer (Node.js)

1. **Project Overview** 
   * **Purpose:**

This project aims to harness the power of deep learning to automatically classify fabric patterns from images into categories such as floral, geometric, abstract, or plain. The system is designed to help textile manufacturers, e-commerce platforms, and fashion designers manage and search fabric collections efficiently, reducing manual labor and ensuring consistency.

**Features:**

* + User-friendly web interface for image upload and prediction.
  + Secure user authentication and role-based access control.
  + Integration with a deep learning model for pattern classification.
  + Admin dashboard for managing pattern records and analytics.
  + Visual analytics for exploring classification trends and results.

1. **Architecture**

* **Frontend (React):**
  + Built using React with React Router for navigation.
  + Axios used for RESTful API communication.
  + Styled with Tailwind CSS for modern, responsive design.
  + Component-based structure (e.g., Navbar, Upload Form, Results Card).
* **Backend (Node.js & Express.js):**
  + REST API to handle frontend requests.
  + Routes for user management, image upload, prediction, and analytics.
  + ML model integration via Python shell or API endpoint using Flask.
  + Middleware for error handling and token verification.
    - **Database (MongoDB):**
* User schema: name, email, hashed password, role.
* Pattern prediction schema: image metadata, predicted class, confidence scores.
* Mongoose ODM for schema modeling and queries.

1. **Setup Instructions** 
   * **Prerequisites:**
     + Node.js (v18+)
     + MongoDB (local or Atlas)
     + Git
     + Python 3.8+ (for ML model)
     + npm or yarn

* **Installation Steps:**

# Clone the repository

git clone https://github.com/your-org/pattern-sense.git

cd pattern-sense

# Setup backend

cd server

npm install

# Setup frontend

cd ../client

npm install

# Setup Python ML model

cd ../ml-model

pip install -r requirements.txt

* + **Environment Variables (.env in server folder):**

PORT=5000

MONGO\_URI=mongodb+srv://<username>:<password>@cluster.mongodb.net/db

JWT\_SECRET=your\_jwt\_secret\_key

PYTHON\_SCRIPT\_PATH=./ml-model/predict.py

1. **Folder Structure** 
   * **Client (React):**
   * client/
   * ├── src/
   * │ ├── components/
   * │ ├── pages/
   * │ ├── services/
   * │ ├── App.js
   * │ └── index.js
   * ├── public/
   * └── package.json
   * **Server (Node.js):**
   * server/
   * ├── controllers/
   * ├── routes/
   * ├── models/
   * ├── middleware/
   * ├── utils/
   * ├── server.js
   * └── .env
   * **ML Model (Python):**

ml-model/

├── predict.py

├── model.pkl

└── requirements.txt

1. **Running the Application** 
   * **Front end:**
   * cd client
   * npm start

* **Back end:**
  + cd server
  + npm start
  + **ML Model Server (optional if standalone Flask app):**
  + cd ml-model
  + python app.py

|  |  |  |  |
| --- | --- | --- | --- |
| **7. API Documentation**  **Endpoint Method Description** | | **Request Body / Params Sample Response** | |
| /api/auth/register POST | Register a new user | { name, email, password}  { token, user } | |
| /api/auth/login POST | Login user | { email, password } | { token, user } |
| /api/predict POST | Submit image for pattern classification | { image File} | { pattern: "Floral", confidence: 92.4} |
| /api/records GET | Retrieve all predictions (admin) | JWT Token | [{ id, imageMeta, result }] |
| **8. Authentication**.   * Tokens stored securely in local storage. * Protected routes enforced through middleware. * User roles (admin, user) control access to advanced features like analytics and record management. | |  |  |

1. **User Interface**

(Add images in actual README or documentation PDF)

* Login / Register Screens
* Upload Form for pattern prediction
* Prediction Result View with pattern label and confidence
* Admin Dashboard for managing predictions
* Visual Analytics Dashboard displaying trends in fabric types

1. **Testing** 
   * **Tools Used:**

* Jest (unit testing for backend logic)
* React Testing Library (component testing)
* Postman (manual API testing)
* PyTest (testing the Python ML model)
  + **Strategy:**
  + Unit tests for validation and utility functions.
  + Integration tests for REST API endpoints.
  + Snapshot/UI testing for React components.
  + Performance testing for large batch predictions.

1. **Screenshots or Demo** 
   * 📷 *Screenshots of Key Pages:*

* Login Page
* Fabric Upload Form
* Prediction Results
* Admin Dashboard
* Analytics Charts

1. **Known Issues** 
   * Performance may vary with very high-resolution images.
   * Limited testing on various mobile devices.
   * Bulk image uploads not yet supported.
   * Lighting variations in images can affect prediction accuracy.

1. **Future Enhancements** 
   * Add support for bulk image upload and batch classification.
   * Improve model accuracy with more diverse training datasets.
   * Implement multilingual support for global users.
   * Create a mobile app using React Native.
   * Integrate explainable AI methods (e.g., Grad-CAM) for visual explanations of model predictions.