Pattern Sense: Classifying Fabric Patterns Using Deep Learning

---

1. Introduction

- **Project Title**: Pattern Sense: Classifying Fabric Patterns Using Deep Learning
- **Team Members**:
    - Anurag Nandamala – ML Developer & Project Lead
    - Ajmeera Harika – Frontend Developer
    - Adigopula Hemanth – Backend Developer
    - Arepalli Jeevan Surendra – UI/UX Designer

---

2. Project Overview

- **Purpose**:
  This project aims to provide predictive analytics capabilities using machine learning models through a web-based interface. The system allows users to input data and receive predictions powered by TensorFlow and scikit-learn.
- **Features**:
    - User authentication and dashboard
    - Model training and evaluation
    - Visualizations using matplotlib
    - Real-time prediction functionality

---

3. Architecture

- **Frontend (React)**:
    - Built with React.js
    - Uses functional components with hooks
    - Axios for API calls
    - React Router for navigation
- **Backend (Node.js + Express.js)**:
    - Handles API requests
    - Manages model inference and user sessions
    - Serves static React frontend in production
- **Database (MongoDB)**:

- Stores user data and model logs
- Mongoose used for schema and database operations

---

4. Setup Instructions

- **Prerequisites**:
  - Node.js (v16+)
  - npm
  - MongoDB
  - Python 3.9+
  - Virtualenv (optional)
- **Installation**:

```
git clone https://github.com/your-repo-url.git
cd your-repo
# Frontend Setup
cd client
npm install
# Backend Setup
cd ../server
npm install
# Python Setup
pip install -r requirements.txt
```

- Create .env file in server with:

```
MONGO_URI=your_mongodb_uri
JWT_SECRET=your_jwt_secret
```

5. Folder Structure

- **Client (React Frontend)**:

```
client/
  ├— public/
  ├— src/
  |   ├— components/
  |   ├— pages/
```

```
|   ├── services/
|   └── App.js
```

- **Server (Node.js Backend)**:
  ```
  server/
    ├── routes/
    ├── controllers/
    ├── models/
    ├── app.js
    └── .env
  ```

6. Running the Application

- **Frontend**:
  ```
  cd client
  npm start
  ```
- **Backend**:
  ```
  cd server
  npm start
  ```

7. API Documentation

Endpoint	Method	Description	Params/Input	Example Response
/api/predict	POST	Get prediction from model	{ features: [..] }	{ prediction: "Class A" }
/api/users	GET	Fetch user list (admin)	Authorization header required	[{name: "Hemanth"}]

8. Authentication

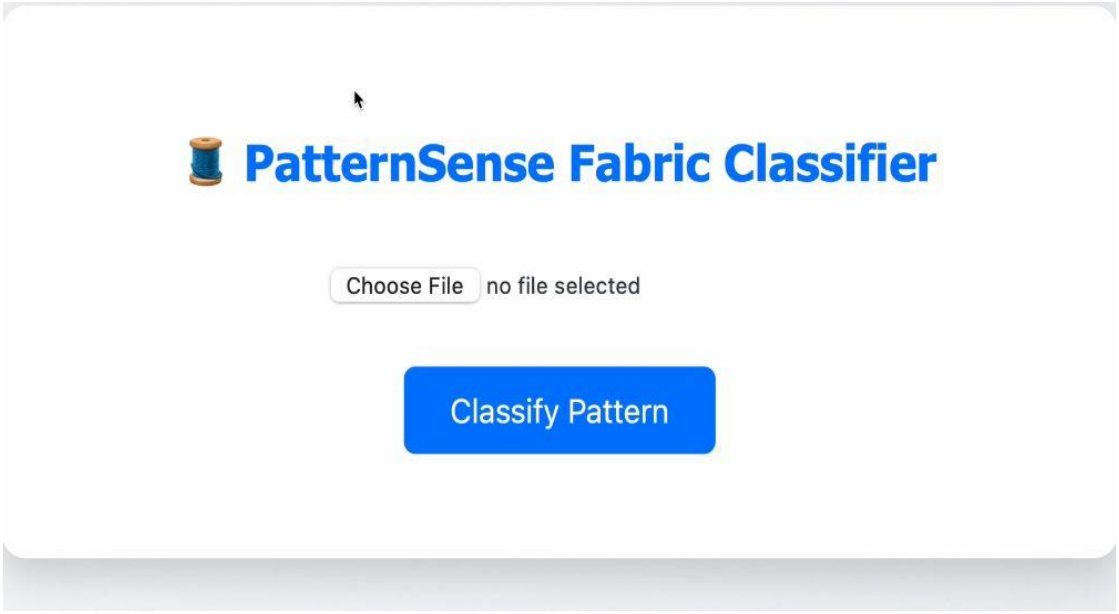
- JWT-based authentication
- Tokens stored in localStorage
- Secure endpoints using middleware (authMiddleware.js)
- Login and signup endpoints issue tokens on success

---

9. User Interface

***Key UI Features*:**

- Login/Signup Forms
- Dashboard with data input
- Graphs for predictions using Chart.js/matplotlib



10. Testing

- **Frontend**: Jest with React Testing Library
- **Backend**: Mocha + Chai for API routes
- **ML Model**: Accuracy, Precision, Confusion Matrix using scikit-learn

---

11. Screenshots or Demo

```
Epoch 1/10, Loss: 1.5783
Epoch 2/10, Loss: 1.2149
Epoch 3/10, Loss: 0.8326
Epoch 4/10, Loss: 0.6505
Epoch 5/10, Loss: 0.5608
Epoch 6/10, Loss: 0.4600
Epoch 7/10, Loss: 0.4196
Epoch 8/10, Loss: 0.3904
Epoch 9/10, Loss: 0.3574
Epoch 10/10, Loss: 0,3216

Classification Report: f1-score  suport
              precision  recall f1-score  sup
dots               0.92    0.85    0.820   20
stripes            0.89    0.91    0.920   20
checks             2  2    2.87    0.500   60
accuracy           0.87    0.87       60
weighted avg
```

12. Known Issues

- Long model prediction latency for large datasets
- No error handling on backend for invalid inputs

- Mobile UI responsiveness not fully optimized

---

13. Future Enhancements

- Integrate cloud storage for model files
- Add support for multiple model types (e.g., classification, regression)
- Deploy with Docker and Kubernetes
- Role-based access control for admin features