COLLEGE CODE : 1120

COLLEGE NAME : JAYA SAKTHI ENGINEERING COLLEGE

DEPARTMENT : COMPUTER SCIENCE AND ENGINEERING

STUDENT NM-ID : aut1120CS50

ROLL NO : 112023104029

DATE : 14-05-2025

TECHNOLOGY-PROJECT NAME : QUALITY CONTROL AND MANUFACTURING

SUBMITTED BY,

VIGNESH.M
ARUNSANKAR.S
BALAGURUBARAN.D
GURUPRASATH.D

AI-Powered Quality Control in Manufacturing

Abstract :

The AI-Powered Quality Control in Manufacturing project is designed to enhance product consistency and minimize defects in industrial environments by leveraging artificial intelligence, computer vision, and IoT technologies. This final phase integrates advanced AI models for defect detection, real-time monitoring through sensors and cameras, and secure data management. The system is scalable and can be integrated with Enterprise Resource Planning (ERP) systems for streamlined operations. This document presents a detailed report of the project, including demonstration outcomes, system architecture, codebase, performance metrics, and testing reports. Screenshots, diagrams, and code excerpts are included to explain the system's design and functionality.

INDEX :

1. Project Demonstration

Overview:

The AI-Powered Quality Control system will be demonstrated to stakeholders to highlight key features such as defect detection, real-time monitoring, and system responsiveness. It will showcase the use of computer vision, IoT data integration, and performance metrics.

Demonstration Details :

System Walkthrough: A live demonstration showing the inspection of products on a production line, detecting defects using AI.

AI Accuracy: Real-time demonstration of AI classifying products as pass/fail based on trained defect parameters.

Performance Metrics: Focus on response time, data throughput, and system stability under continuous monitoring.

Security & Privacy: Demonstration of secure data handling and logging using encryption protocols.

Outcome :

Stakeholders will witness the system’s capability in automating quality checks, improving inspection accuracy, and integrating seamlessly with existing production environments.

2. Project Documentation

Overview :

Detailed documentation is provided covering the system’s architecture, development process, and usage instructions for both operators and administrators.

Documentation Sections :

System Architecture: Diagrams showing the flow from image capture to AI analysis and output decisions.

Code Documentation: Description and samples of the AI model training, sensor integration, and the control dashboard.

User Guide: Instructions for operators to use the inspection system, interpret alerts, and manage product data.

Administrator Guide: Guidelines for system setup, performance tuning, model retraining, and issue troubleshooting.

Testing Reports: Results of system testing including accuracy metrics, latency, and stress-testing scenarios.

Outcome :

Provides comprehensive knowledge transfer for future developers or adopters to maintain or expand the system.

3. Feedback and Final Adjustments

Overview :

Feedback will be collected post-demonstration to refine the system for broader deployment.

Feedback Collection: Gather input from production supervisors, stakeholders, and test users via forms and live observation.

Final Testing: Conduct final rounds of validation to ensure stability, accuracy, and real-time performance.

Outcome :

A refined and deployment-ready system optimized based on real-world feedback.

4. Final Project Report Submission

Overview :

This report will summarize the journey, challenges, outcomes, and performance improvements of the project.

Report Sections :

Executive Summary: Overview of goals, achievements, and industry relevance.

Phase Breakdown: Details of each phase including AI model development, hardware integration, and dashboard creation.

Challenges & Solutions: Issues such as false positives or data loss and the solutions implemented.

Outcomes: Final system performance, defect detection accuracy, and readiness for implementation.

Outcome :

A complete record of the project lifecycle, from concept to final implementation.

5. Project Handover and Future Works

Overview:

Planning for the future development of the system.

Handover Details:

Next Steps: Suggestions include scaling to multiple production lines, expanding AI to new defect types, and adding multilingual interfaces.

Outcome:

Formal handover with future roadmap and system sustainability guide.

Attachments:

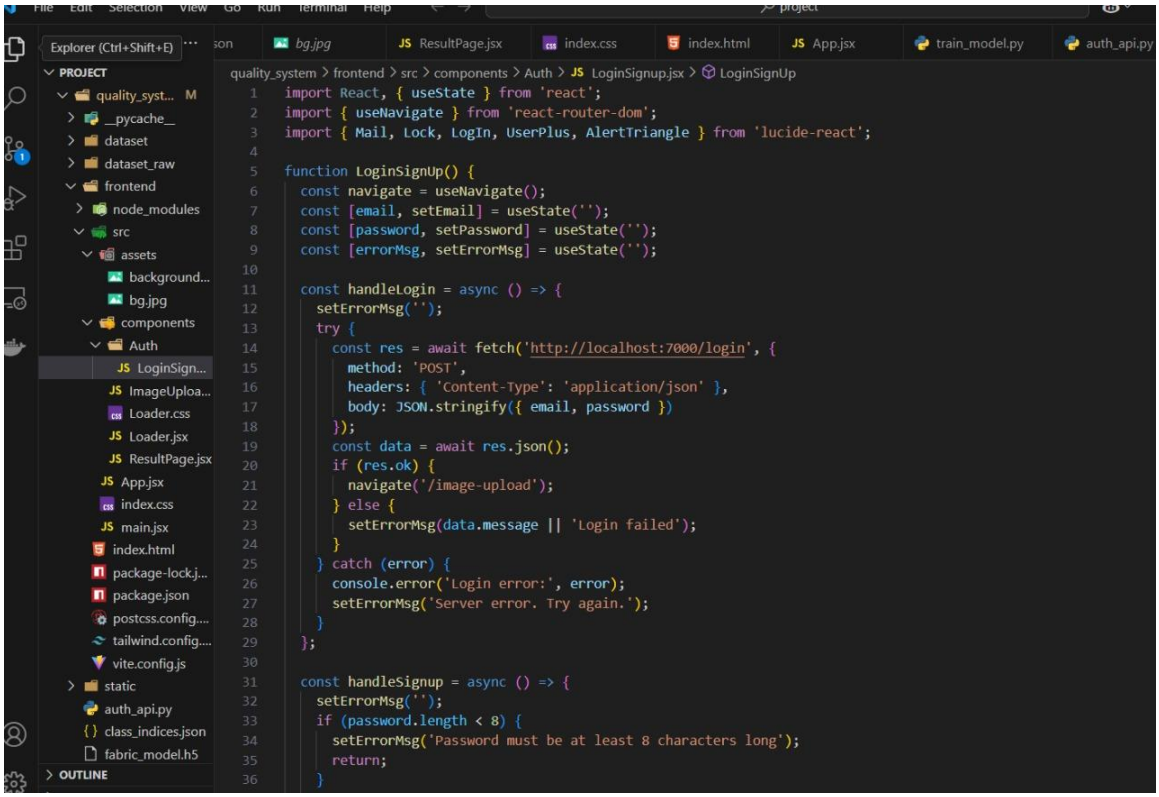
source code :

```
from flask import Flask, request, jsonify

from flask_cors import CORS

import tensorflow as tf

import numpy as np

from PIL import Image

import io

import os


app = Flask(_name_)

CORS(app, resources={r"/predict": {"origins": "http://localhost:5173"}})


# Load the model

MODEL_PATH = "fabric_model.h5"

model = tf.keras.models.load_model(MODEL_PATH)
```

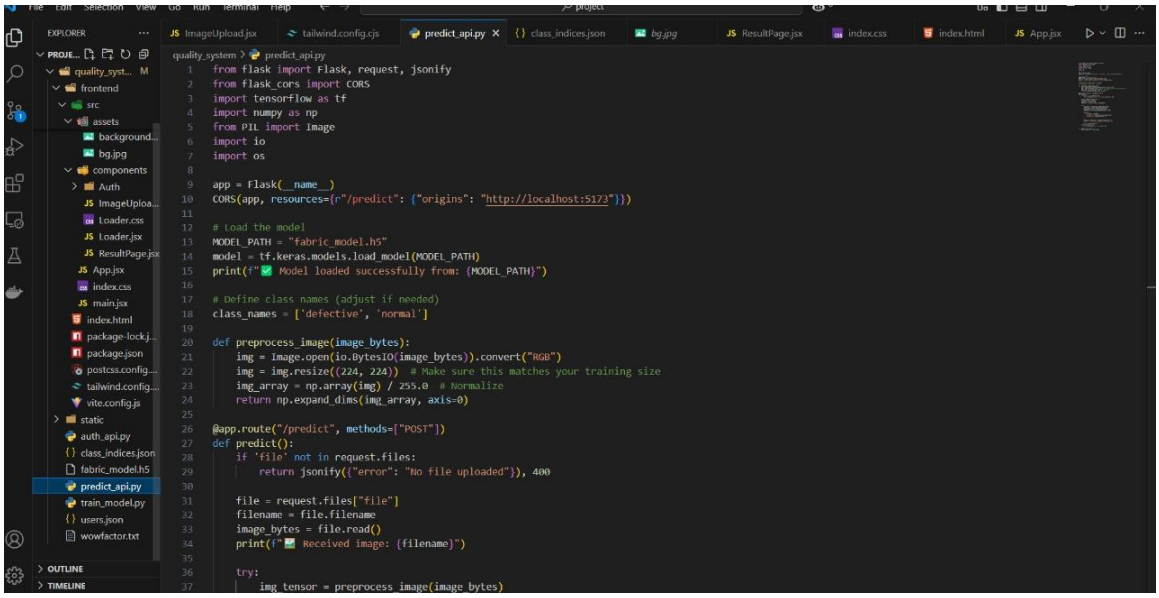
```
print(f"✅ Model loaded successfully from: {MODEL_PATH}")

# Define class names (adjust if needed)
class_names = ['defective', 'normal']

def preprocess_image(image_bytes):
    img = Image.open(io.BytesIO(image_bytes)).convert("RGB")
    img = img.resize((224, 224))  # Make sure this matches your training size
    img_array = np.array(img) / 255.0  # Normalize
    return np.expand_dims(img_array, axis=0)

@app.route("/predict", methods=["POST"])
def predict():
    if 'file' not in request.files:
        return jsonify({"error": "No file uploaded"}), 400

    file = request.files["file"]
    filename = file.filename
    image_bytes = file.read()
    print(f"🖼 Received image: {filename}")

    try:
        img_tensor = preprocess_image(image_bytes)
        predictions = model.predict(img_tensor)
        predicted_index = np.argmax(predictions)
        confidence = float(np.max(predictions) * 100)
```

```
        result = {
            "filename": filename,
            "prediction": class_names[predicted_index],
            "confidence": round(confidence, 2)
        }

        print(f"🔍 Prediction: {result['prediction']}")
        print(f"📈 Confidence: {result['confidence']}%")

        return jsonify(result)
    except Exception as e:
        return jsonify({"error": str(e)}), 500

if _name_ == "_main_":
    app.run(debug=True, port=5000)
```

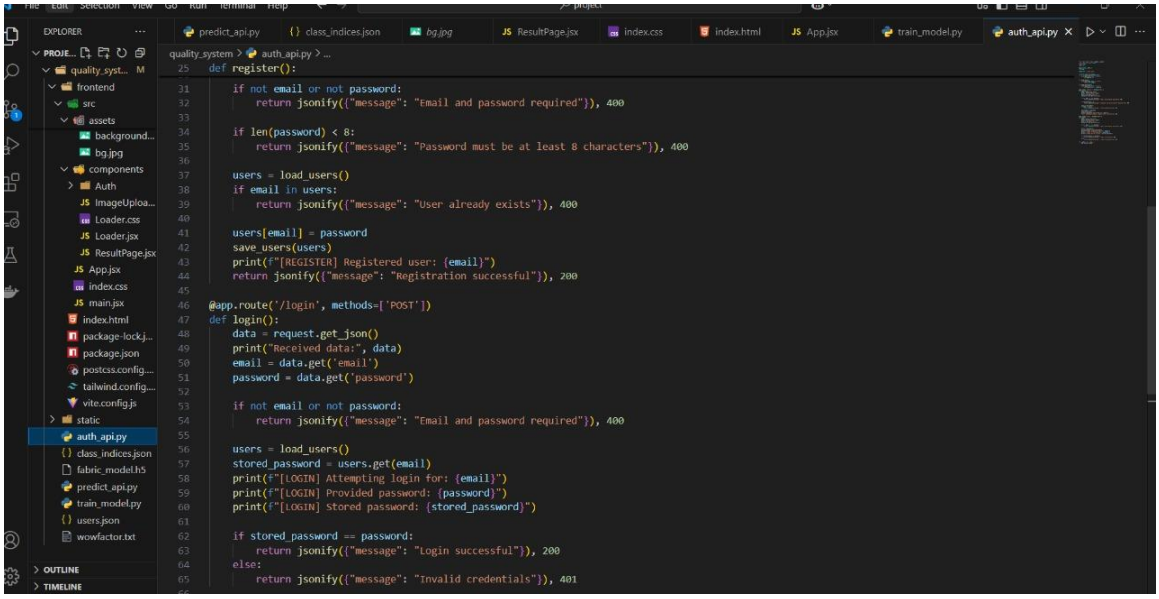
Screenshots:



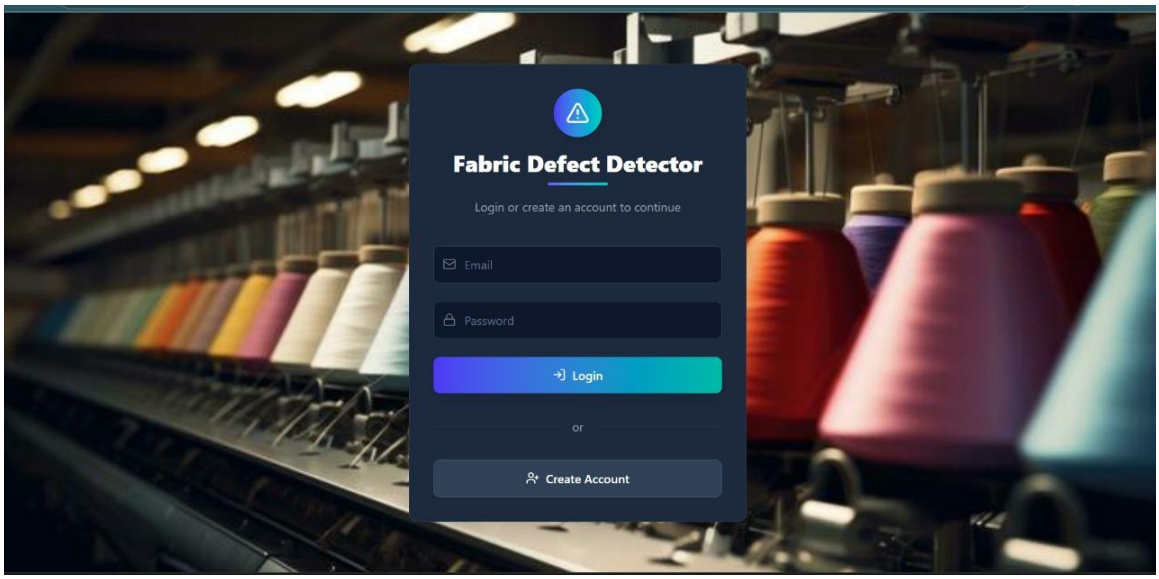
```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { Mail, Lock, LogIn, UserPlus, AlertTriangle } from 'lucide-react';

function LoginSignUp() {
  const navigate = useNavigate();
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [errorMsg, setErrorMsg] = useState('');

  const handleLogin = async () => {
    setErrorMsg('');
    try {
      const res = await fetch('http://localhost:7000/login', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ email, password })
      });
      const data = await res.json();
      if (res.ok) {
        navigate('/image-upload');
      } else {
        setErrorMsg(data.message || 'Login failed');
      }
    } catch (error) {
      console.error('Login error:', error);
      setErrorMsg('Server error. Try again.');
    }
  };

  const handleSignup = async () => {
    setErrorMsg('');
    if (password.length < 8) {
      setErrorMsg('Password must be at least 8 characters long');
      return;
    }
```

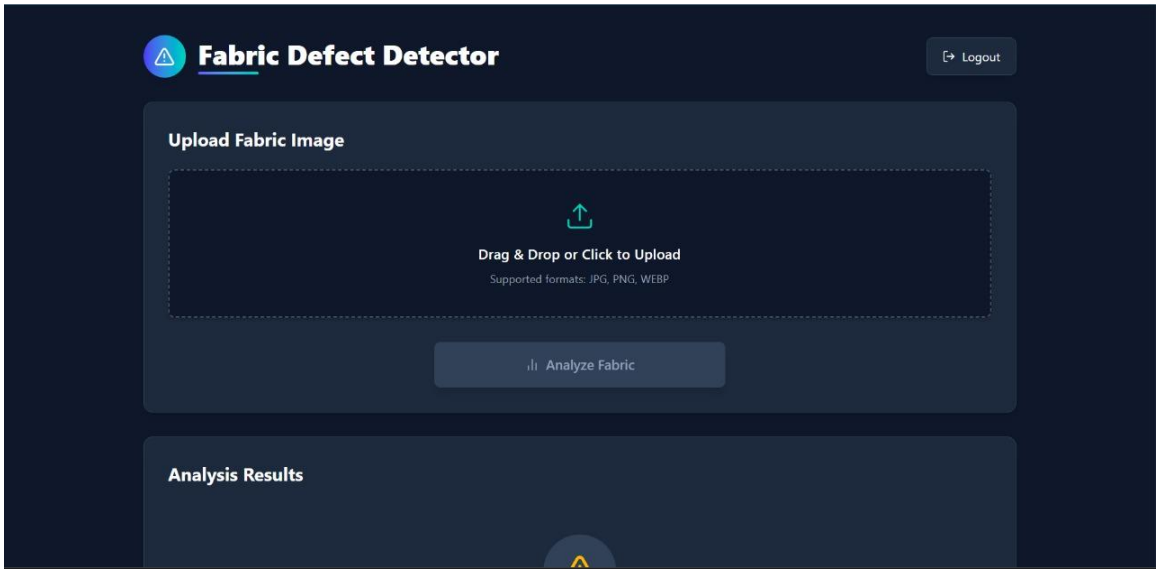


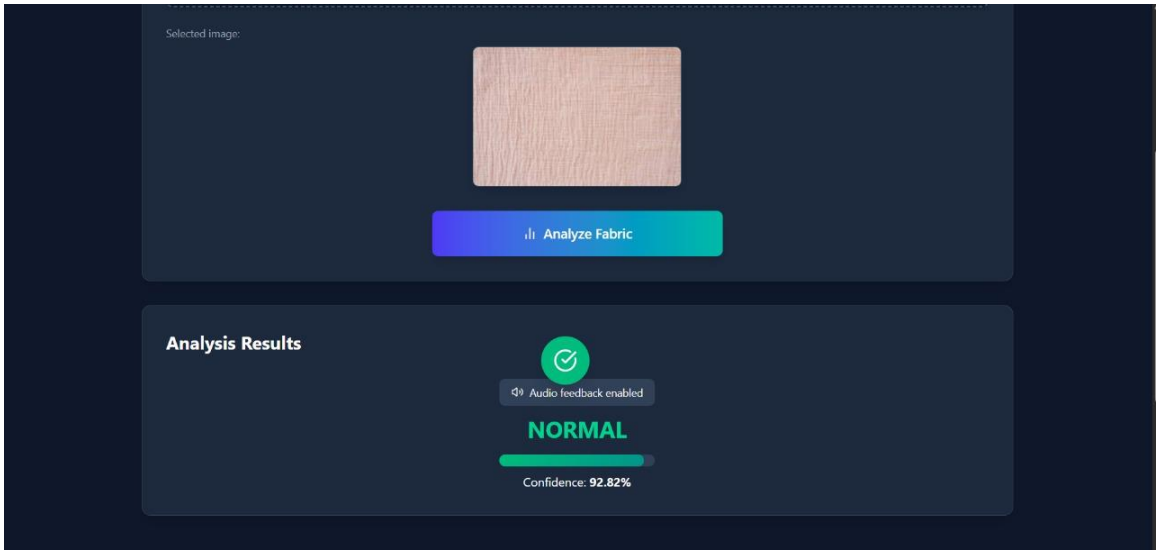
```
from flask import Flask, request, jsonify
from flask_cors import CORS
import tensorflow as tf
import numpy as np
from PIL import Image
import io
import os

app = Flask(__name__)
CORS(app, resources={r"/predict": {"origins": "http://localhost:5173"}})

# Load the model
MODEL_PATH = "fabric_model.h5"
model = tf.keras.models.load_model(MODEL_PATH)
print(f"✅ Model loaded successfully from: {MODEL_PATH}")

# Define class names (adjust if needed)
class_names = ['defective', 'normal']

def preprocess_image(image_bytes):
    img = Image.open(io.BytesIO(image_bytes)).convert("RGB")
    img = img.resize((224, 224))  # Make sure this matches your training size
    img_array = np.array(img) / 255.0  # Normalize
    return np.expand_dims(img_array, axis=0)

@app.route("/predict", methods=["POST"])
def predict():
    if 'file' not in request.files:
        return jsonify({"error": "No file uploaded"}), 400

    file = request.files["file"]
    filename = file.filename
    image_bytes = file.read()
    print(f"🖼️ Received image: {filename}")

    try:
        img_tensor = preprocess_image(image_bytes)
```

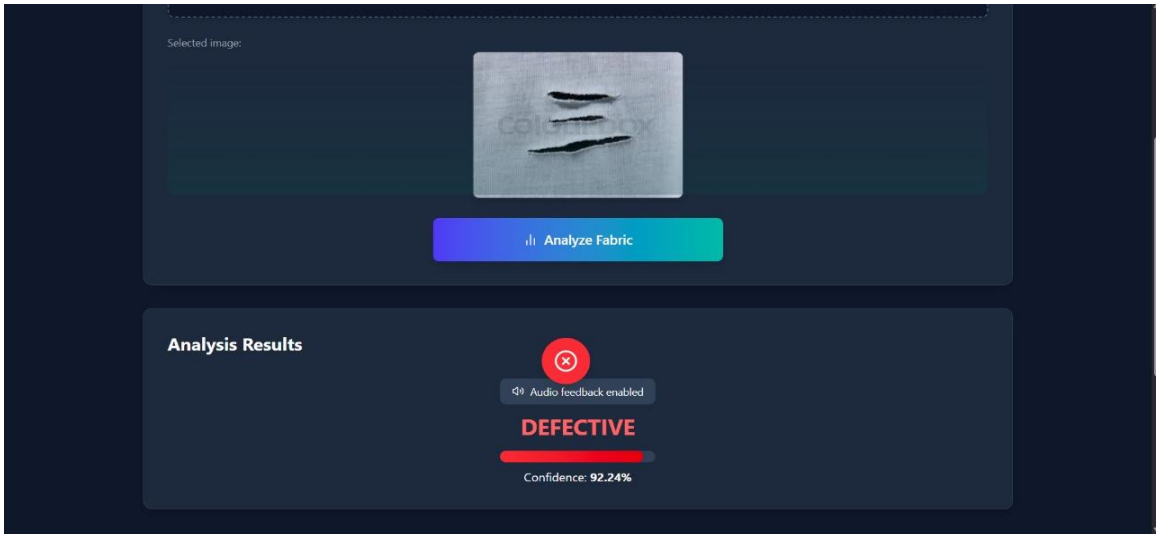

EXPLORER
predict_api.py
class_indices.json
bg.jpg
ResultPage.jsx
index.css
index.html
App.jsx
train_model.py
auth_api.py
quality_system > auth_api.py > ...
def register():
if not email or not password:
return jsonify({"message": "Email and password required"}), 400
if len(password) < 8:
return jsonify({"message": "Password must be at least 8 characters"}), 400
users = load_users()
if email in users:
return jsonify({"message": "User already exists"}), 400
users[email] = password
save_users(users)
print(f"[REGISTER] Registered user: {email}")
return jsonify({"message": "Registration successful"}), 200
@app.route('/login', methods=['POST'])
def login():
data = request.get_json()
print("Received data:", data)
email = data.get('email')
password = data.get('password')
if not email or not password:
return jsonify({"message": "Email and password required"}), 400
users = load_users()
stored_password = users.get(email)
print(f"[LOGIN] Attempting login for: {email}")
print(f"[LOGIN] Provided password: {password}")
print(f"[LOGIN] Stored password: {stored_password}")
if stored_password == password:
return jsonify({"message": "Login successful"}), 200
else:
return jsonify({"message": "Invalid credentials"}), 401
OUTLINE
TIMELINE

Output :


Fabric Defect Detector
Login or create an account to continue
Email
Password
Login
or
Create Account


Fabric Defect Detector
Logout
Upload Fabric Image
Drag & Drop or Click to Upload
Supported formats: JPG, PNG, WEBP
Analyze Fabric
Analysis Results


Selected image:
Analyze Fabric
Analysis Results
Audio feedback enabled
NORMAL
Confidence: 92.82%


Selected image:
Analyze Fabric
Analysis Results
Audio feedback enabled
DEFECTIVE
Confidence: 92.24%