Skin disease Portal

Information and Communication Technology

2024-25

CERTIFICATE

This is to certify that the project entitled **project** has been carried out by Diya Kaneriya (92201133034) & Shyama Vagadia (92200133005) under my guidance in partial fulfilment of the degree of Bachelor Engineering in Information and Communication Technology (6th Semester) of Marwadi University, Rajkot during the academic year 2023-24.

Date: 16-05-2025

Internal Guide	**Head of the Department**
Mr. Shamsagazarzoo Alam Subject Co-Ordinator C.T.O. Ally Soft Solutions	Prof. C. D. Parmar Head of Department ICT Engineering

Abstract

The Skin Disease Detection Portal is an AI-integrated web-based healthcare application that showcases the full potential of advanced web technologies to solve real-world problems. It offers a user-friendly portal where individuals can upload images of skin conditions and receive preliminary insights based on automated image analysis. The system is designed to simulate a production-level healthcare diagnostic portal using modular microservices and a decoupled frontend-backend architecture.

The portal combines the strengths of a modern frontend developed in React.js, a Node.js backend for API handling, and a Flask-based Python microservice responsible for image preprocessing and disease classification using machine learning. This integration allows seamless user experience, asynchronous communication, and scalable deployment, exemplifying a full-stack advanced web system.

Table of Contents

Introduction

Healthcare applications often demand reliability, user-friendliness, and intelligent functionality. With the increasing availability of high-speed internet and smart devices, web portals can be powerful tools for initial medical screening and awareness. This project is a web-based platform that demonstrates how advanced web technologies can be integrated with artificial intelligence to provide accessible and scalable solutions for preliminary skin disease detection.

Problem Statement

In rural or underdeveloped regions, dermatological expertise is often scarce. Delays in skin disease diagnosis can lead to worsening conditions. Even in urban areas, consultation appointments can be costly and time-consuming. There is a need for an accessible web portal that can accept user-submitted skin images and offer basic condition detection using automated tools. The challenge lies in building a robust, real-time, and user-centric web application that communicates effectively with a remote ML service.

Objectives

Develop a fully responsive web portal for skin disease detection.

Use asynchronous REST APIs to connect frontend, backend, and ML service.

Ensure clean UI/UX for both desktop and mobile experiences.

Create a modular backend that separates business logic and API control.

Integrate a machine learning microservice for inference.

Provide real-time predictions and display results with confidence scores.

Enable scalability for future feature additions like user profiles and history.

Scope

Supports classification of a limited number of predefined skin conditions.

Can be deployed on cloud servers or local machines.

Only supports image input; does not integrate text or symptom-based input.

No patient-specific record keeping or diagnosis history in current scope.

Literature Review

Numerous studies and startups have attempted to automate dermatological diagnoses. Projects like DermAI and apps like SkinVision leverage mobile interfaces with AI to identify suspicious moles or lesions. However, many are either closed-source, medically regulated, or paid. This portal aims to provide an open educational platform to explore these ideas using current full-stack web technologies.

Key findings include:

CNNs outperform traditional ML methods in image classification.

React-based SPAs ensure better user engagement.

Microservice patterns allow better maintainability in production systems.

Technologies Used

Frontend

React.js: Core frontend framework for UI.

Vite: Development server and build tool.

Tailwind CSS: For consistent, modern UI styling.

Framer Motion: For smooth transitions and animations.

Backend

Node.js: Server-side JavaScript runtime.

Express.js: Web server framework.

Multer: Middleware for handling file uploads.

Axios: Used to call ML microservice from Node.js.

ML Microservice

Python: For model implementation.

Flask: For serving REST API endpoints.

OpenCV & PIL: For image processing.

scikit-learn: For feature extraction and classification.

Tools

Postman: API testing.

VS Code: Development environment.

Git & GitHub: Version control.

System Design and Architecture

The application follows a microservice-oriented architecture, separating concerns among different services.

Modules:

`frontend/`: React application for user interaction.

`backend/`: Express-based server for routing and image forwarding.

`ml-service/`: Python Flask application for feature processing and inference.

Block Diagram:

```
User
↓
Frontend (React)
↓
Backend (Node.js + Express.js)
↓
Microservice (Flask + ML Model)
↓
ML Prediction Response
↓
Frontend Display
```

Communication:

All components communicate via HTTP/REST APIs.

Images are sent as multipart/form-data.

JSON is used for response payloads.

Data Flow & Component Interaction

1. User selects an image and clicks upload.
2. React component triggers POST `/upload` endpoint.
3. Express backend saves the image and sends it to ML API.
4. Python Flask service extracts features and returns class.
5. Node.js backend forwards prediction to frontend.
6. React frontend displays the result with user-friendly labels.

Implementation Details

Folder Structure

```
- frontend/
    - components/
    - pages/
    - App.jsx
- backend/
    - routes/
    - app.js
```

- ml-service/
  - app.py
  - model.pkl
```

# APIs

`POST /upload` – handles file input.

`POST /predict` – called by backend to get ML results.

# Frontend Design

Built with React.js, the frontend emphasizes clarity, responsiveness, and accessibility. Major UI sections include:

Landing Page with Project Overview

Upload Section (Drag & Drop)

Result Screen with Prediction Feedback

Error Handling for unsupported files

# Features

Component-based architecture
```

Tailwind responsive grid

Framer Motion animations

Axios for async requests

Backend Services

The Node.js backend handles:

File reception via Multer

Calling Python service with image path or stream

Returning predictions to the frontend

Handling CORS and server security

Sample Code:

```js
app.post('/upload', upload.single('image'), async (req, res) => {
  const formData = new FormData();
  formData.append('image', fs.createReadStream(req.file.path));

  const response = await axios.post('http://localhost:5000/predict', formData);
  res.json(response.data);
});
```

```

# Microservice Functionality

The ML microservice is responsible for loading a model and classifying the input image.

# Function Flow:

1. Load image using PIL.
2. Extract texture and color features.
3. Use trained model (e.g., SVM) to predict class.
4. Return JSON response.

# Sample Code (Flask):

```python
@app.route('/predict', methods=['POST'])
def predict():
    img = Image.open(request.files['image'].stream)
    features = extract_features(img)
    prediction = model.predict([features])
    return jsonify({'prediction': prediction.tolist()})
```
```

```

# Database Integration (Optional Extension)

Although not implemented in the current version, the portal can integrate Supabase or MongoDB to store:

User accounts and login

Upload history

Feedback and results

Deployment Strategy

The project can be deployed using:

Frontend: Vercel or Netlify

Backend: Render, Railway, or Heroku

ML Service: Dockerized Flask app on AWS EC2

Environment files (`.env`) should store sensitive API URLs.
```

Security and Performance

CORS Handling: Configured in backend for secure cross-origin requests.

Rate Limiting: Prevent abuse with basic request throttling.

Input Sanitization: Ensures safe file handling.

Compression: Optimized image size during upload.

Testing and Validation

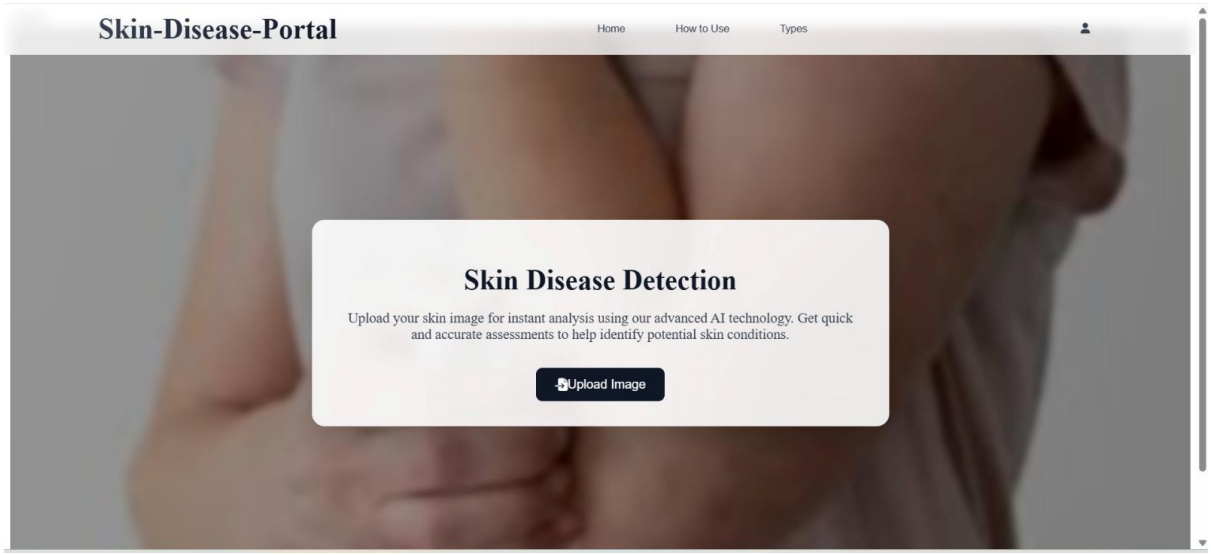
Manual testing of edge cases (wrong format, blank image)

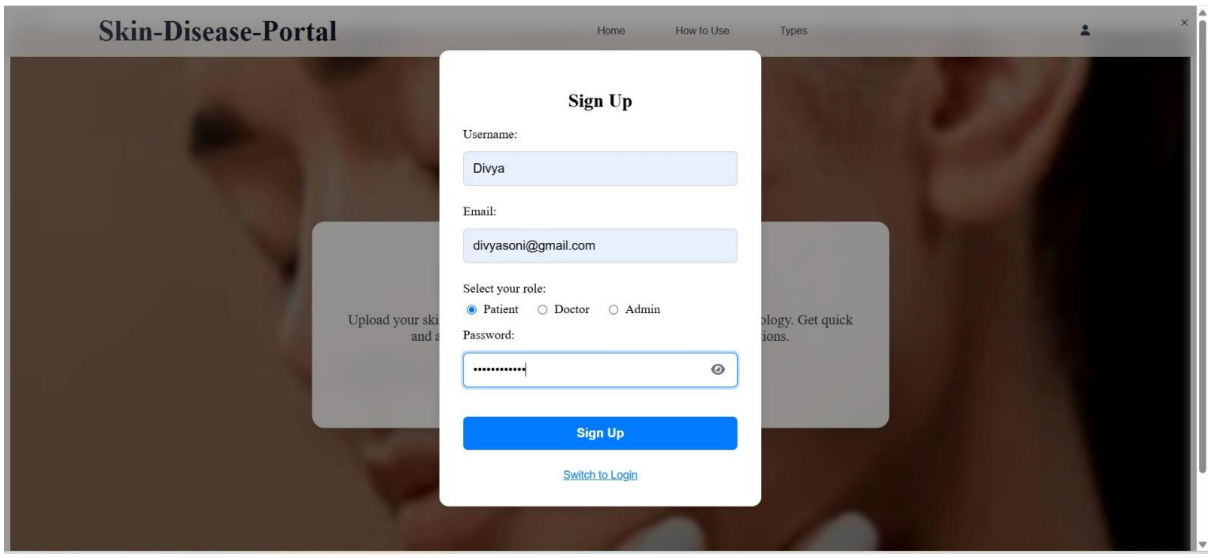
Postman used for backend/microservice testing

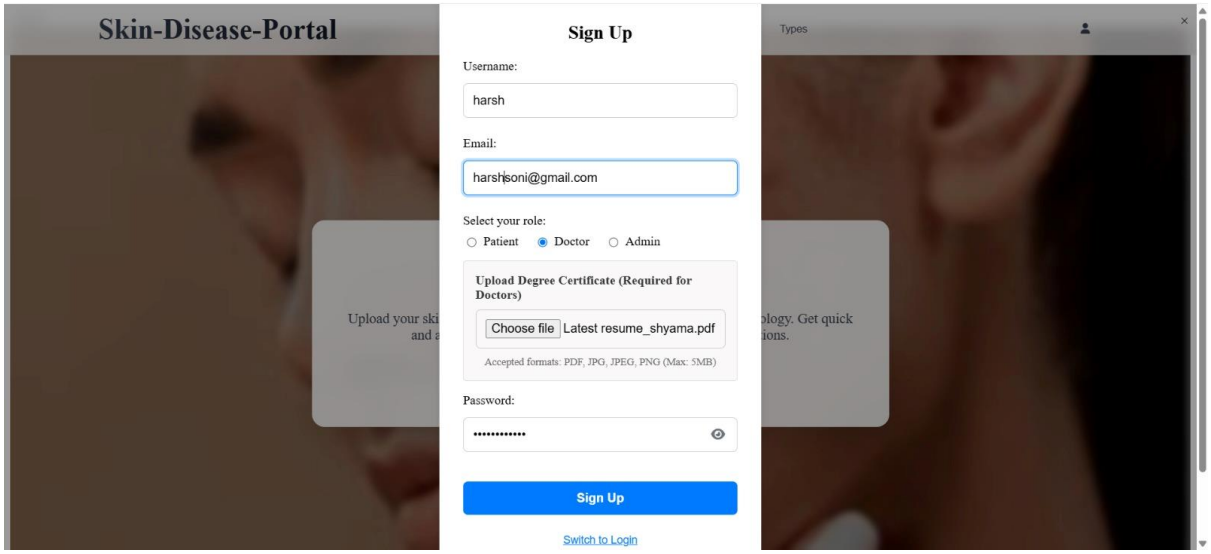
Console-based logging at every stage for debugging

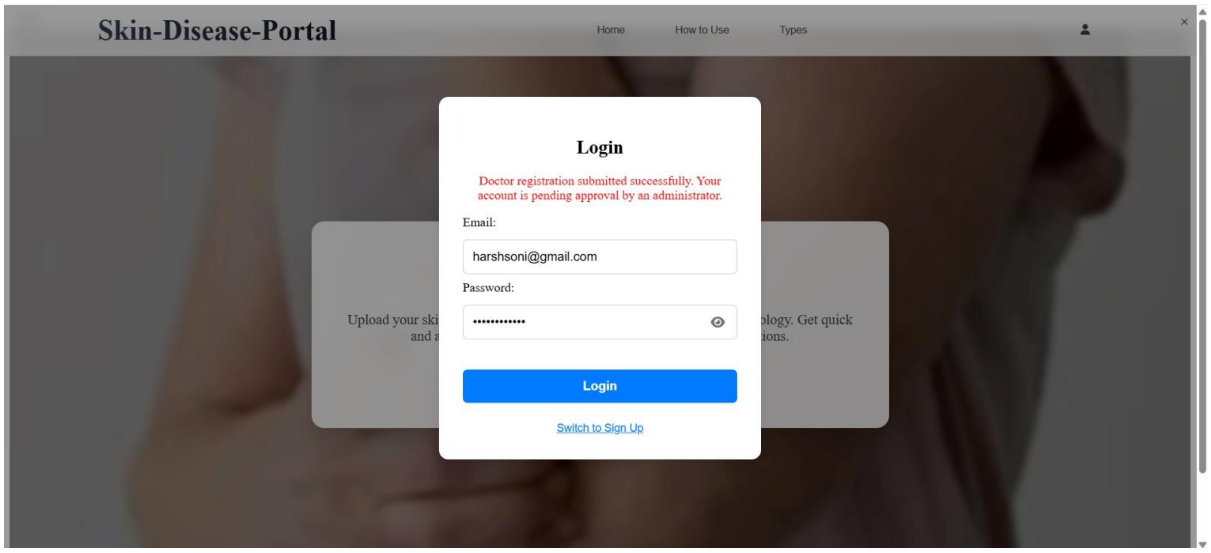
Optional unit test coverage for ML logic

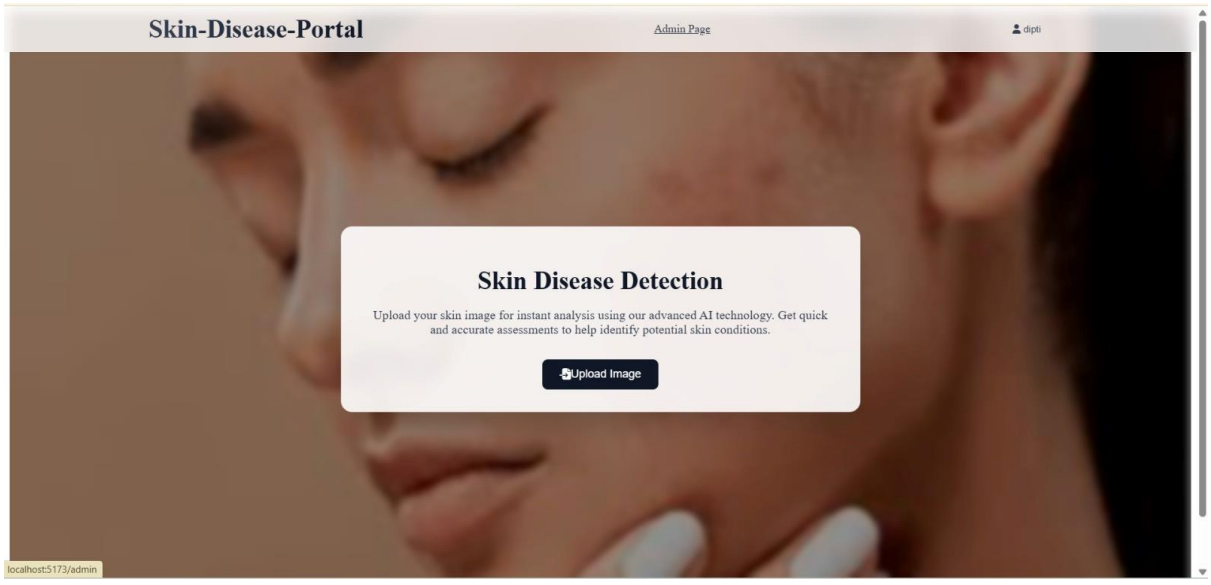
Screenshots & Code Snippets

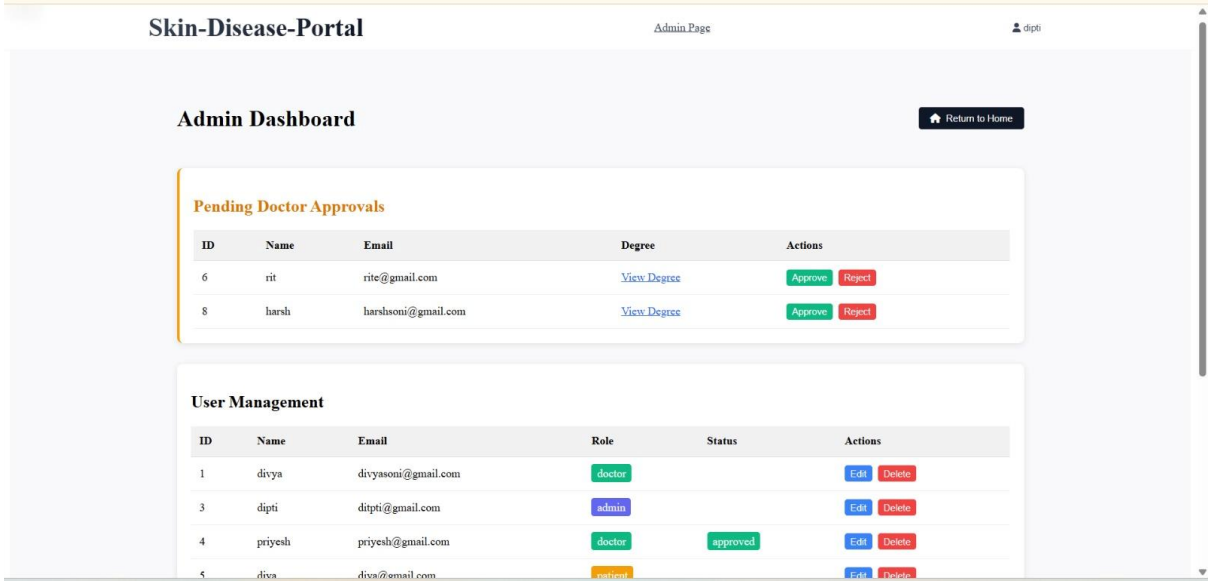

Skin-Disease-Portal
Home
How to Use
Types
Skin Disease Detection
Upload your skin image for instant analysis using our advanced AI technology. Get quick and accurate assessments to help identify potential skin conditions.
Upload Image

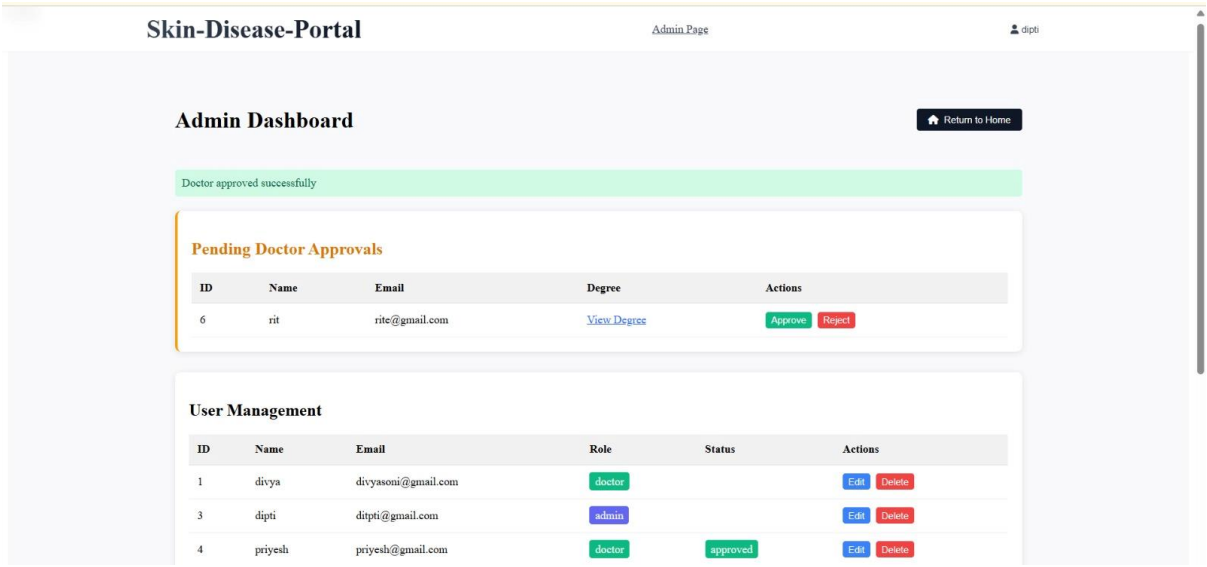

Skin-Disease-Portal
Home
How to Use
Types
Sign Up
Username:
Divya
Email:
divyasoni@gmail.com
Select your role:
Patient
Doctor
Admin
Password:
Sign Up
Switch to Login

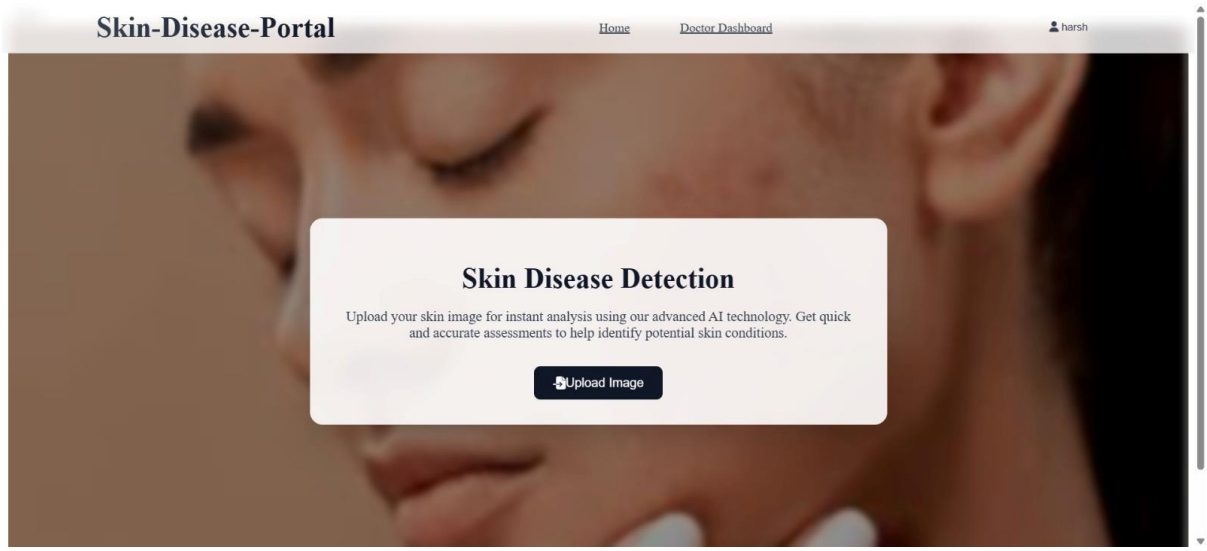

Skin-Disease-Portal
Types
Sign Up
Username:
harsh
Email:
harshsoni@gmail.com
Select your role:
Patient
Doctor
Admin
Upload Degree Certificate (Required for Doctors)
Choose file
Latest resume_shyama.pdf
Accepted formats: PDF, JPG, JPEG, PNG (Max: 5MB)
Password:
Sign Up
Switch to Login

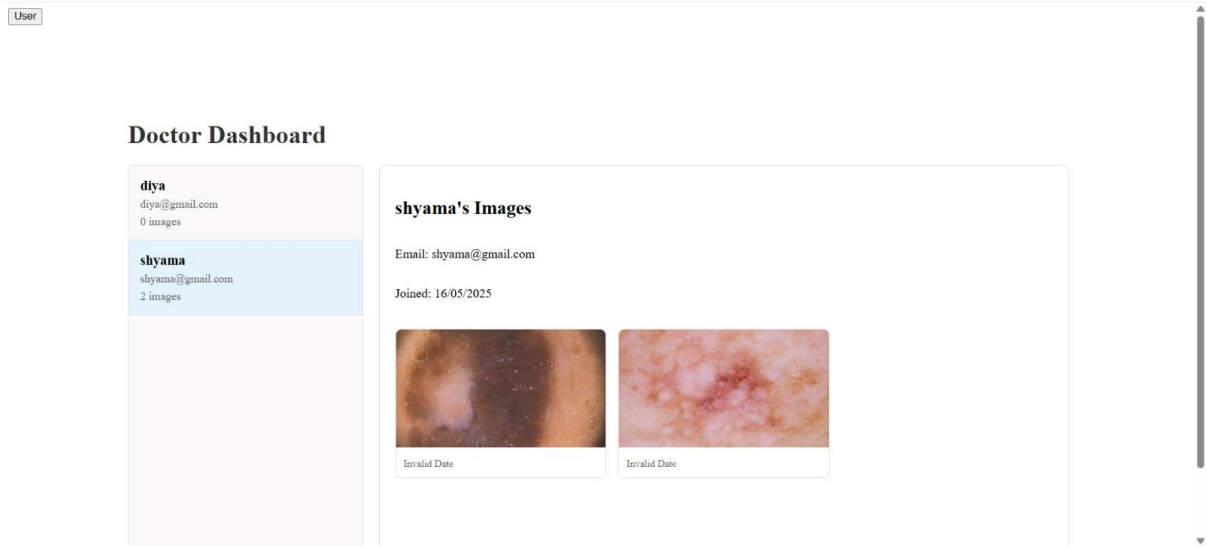

Skin-Disease-Portal
Home
How to Use
Types
Login
Doctor registration submitted successfully. Your account is pending approval by an administrator.
Email:
harshsoni@gmail.com
Password:
Login
Switch to Sign Up

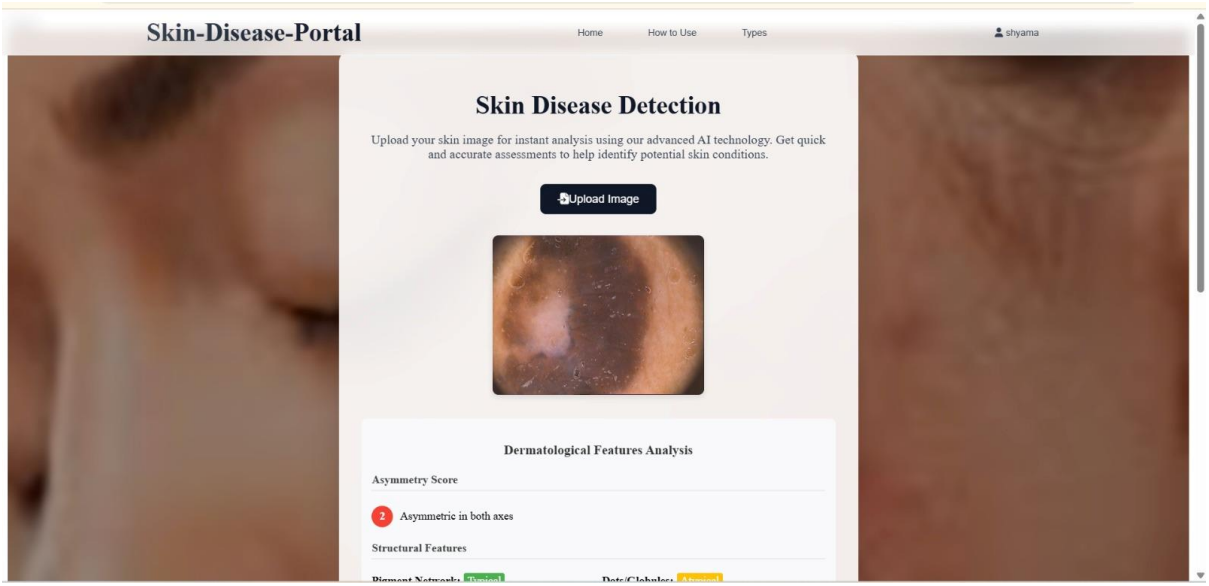

Skin-Disease-Portal
Admin Page
dipti
Skin Disease Detection
Upload your skin image for instant analysis using our advanced AI technology. Get quick and accurate assessments to help identify potential skin conditions.
Upload Image
localhost:5173/admin

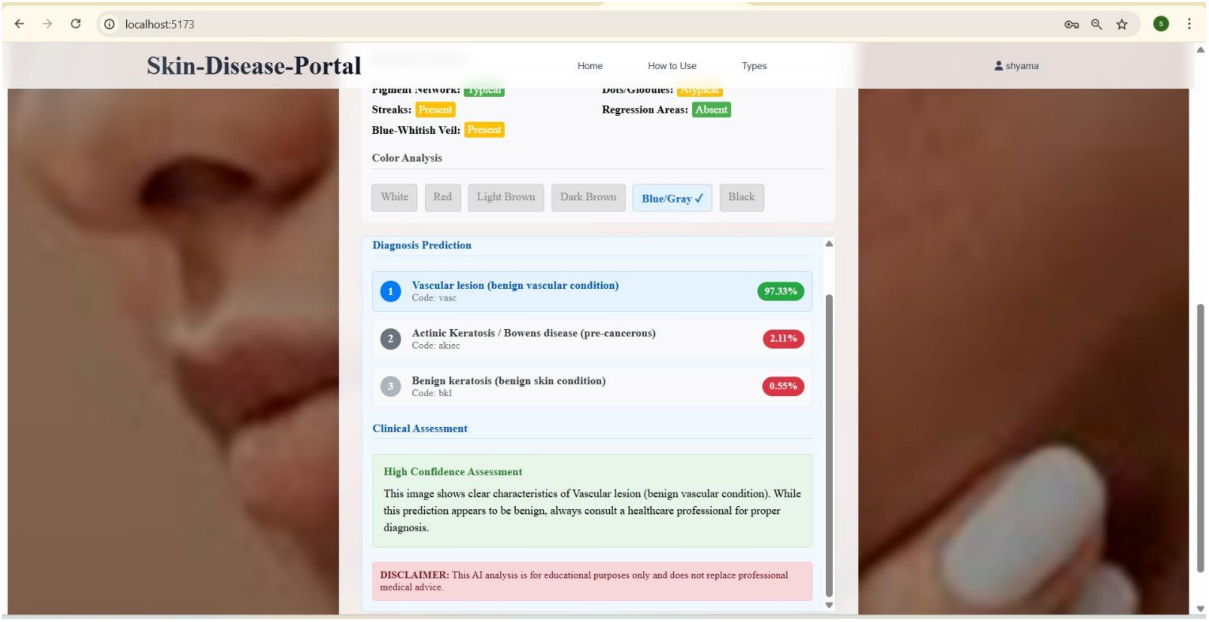

Skin-Disease-Portal
Admin Page
dipti
Admin Dashboard
Return to Home
Pending Doctor Approvals
ID Name Email Degree Actions
6 rit rite@gmail.com View Degree Approve Reject
8 harsh harshsoni@gmail.com View Degree Approve Reject
User Management
ID Name Email Role Status Actions
1 divya divyasoni@gmail.com doctor Edit Delete
3 dipti ditpti@gmail.com admin Edit Delete
4 priyesh priyesh@gmail.com doctor approved Edit Delete


Skin-Disease-Portal
Admin Page
dipti
Admin Dashboard
Return to Home
Doctor approved successfully
Pending Doctor Approvals
ID Name Email Degree Actions
6 rit rite@gmail.com View Degree Approve Reject
User Management
ID Name Email Role Status Actions
1 divya divyasoni@gmail.com doctor Edit Delete
3 dipti ditpti@gmail.com admin Edit Delete
4 priyesh priyesh@gmail.com doctor approved Edit Delete


Skin-Disease-Portal
Home
Doctor Dashboard
harsh
Skin Disease Detection
Upload your skin image for instant analysis using our advanced AI technology. Get quick and accurate assessments to help identify potential skin conditions.
Upload Image


User
Doctor Dashboard
diya
diya@gmail.com
0 images
shyama
shyama@gmail.com
2 images
shyama's Images
Email: shyama@gmail.com
Joined: 16/05/2025
Invalid Date
Invalid Date


Skin-Disease-Portal
Home
How to Use
Types
shyama
Skin Disease Detection
Upload your skin image for instant analysis using our advanced AI technology. Get quick and accurate assessments to help identify potential skin conditions.
Upload Image
Dermatological Features Analysis
Asymmetry Score
2
Asymmetric in both axes
Structural Features


localhost:5173
Skin-Disease-Portal
Home
How to Use
Types
shyama
Streaks: Present
Regression Areas: Absent
Blue-Whitish Veil: Present
Color Analysis
White
Red
Light Brown
Dark Brown
Blue/Gray ✓
Black
Diagnosis Prediction
1
Vascular lesion (benign vascular condition)
Code: vasc
97.33%
2
Actinic Keratosis / Bowens disease (pre-cancerous)
Code: akiec
2.11%
3
Benign keratosis (benign skin condition)
Code: bkl
0.55%
Clinical Assessment
High Confidence Assessment
This image shows clear characteristics of Vascular lesion (benign vascular condition). While this prediction appears to be benign, always consult a healthcare professional for proper diagnosis.
DISCLAIMER: This AI analysis is for educational purposes only and does not replace professional medical advice.