Backend (Node.js )

Result Display and Interpretation

Machine Learning Model (Flask and Tensor Flow)

Database service (MySQL / Cloud Storage )

Image Analysis and Prediction

Image Upload and Processing

Web Interface

(React.js)

Authentication service (Google OAuth)

User Device (Smart phone / Destop )

import React, { useState } from 'react';

import { GoogleOAuthProvider, GoogleLogin, googleLogout } from '@react-oauth/google';

import axios from 'axios';

import './App.css';

import placeholderImage from './Login\_image.png'; // Import the placeholder image

const CLIENT\_ID = '382878482898-9lfn4jhrv03vafkjgabm12ul7o04tjg8.apps.googleusercontent.com';

const App: React.FC = () => {

  const [isAuthenticated, setIsAuthenticated] = useState(false);

  const [user, setUser] = useState<any>(null);

  const [image, setImage] = useState<File | null>(null);

  const [analysisResult, setAnalysisResult] = useState<string | null>(null);

  const [resultImageUrl, setResultImageUrl] = useState<string | null>(null);

  const onSuccess = (response: any) => {

    console.log(response); // Log the response object to see its structure

    const idToken = response.credential; // Adjust to match the correct response property

    if (idToken) {

      const decodedToken = JSON.parse(atob(idToken.split('.')[1]));

      setUser(decodedToken);

      setIsAuthenticated(true);

    } else {

      console.error('No id\_token found in the response:', response);

      setIsAuthenticated(false);

    }

  };

  const onFailure = () => {

    console.error('Login Failed');

    setIsAuthenticated(false);

  };

  const onLogoutSuccess = () => {

    googleLogout();

    setIsAuthenticated(false);

    setUser(null);

    setImage(null);

    setAnalysisResult(null);

    setResultImageUrl(null); // Clear result image URL when logging out

  };

  const handleImageChange = (e: React.ChangeEvent<HTMLInputElement>) => {

    if (e.target.files && e.target.files.length > 0) {

      setImage(e.target.files[0]);

    }

  };

  const handleImageUpload = async () => {

    if (!image) {

      alert('Please select an image to upload.');

      return;

    }

    const formData = new FormData();

    formData.append('image', image);

    try {

      const response = await axios.post('http://localhost:5000/upload', formData, {

        headers: {

          'Content-Type': 'multipart/form-data',

        },

      });

      if (response.data && response.data.analysisResult && response.data.imageUrl) {

        setAnalysisResult(response.data.analysisResult);

        setResultImageUrl(response.data.imageUrl); // Set result image URL

      } else {

        console.error('Invalid response from the server:', response);

      }

    } catch (error) {

      console.error('Error uploading image:', error);

    }

  };

  return (

    <GoogleOAuthProvider clientId={CLIENT\_ID}>

      <div className="App">

        <header className="App-header">

          <h1>Diabetic Retinopathy Detection System</h1>

          {!isAuthenticated ? (

            <GoogleLogin

              onSuccess={onSuccess}

              onError={onFailure}

              useOneTap // Use one-tap login for a better user experience

            />

          ) : (

            <div>

              <button onClick={onLogoutSuccess}>Logout</button>

              <div>

                <h2>Welcome, {user?.name}</h2>

                {user?.picture ? (

                  <img src={user.picture} alt={user.name} style={{ borderRadius: '50%', width: '100px', height: '100px' }} />

                ) : (

                  <img src={placeholderImage} alt="placeholder" style={{ borderRadius: '50%', width: '100px', height: '100px' }} />

                )}

              </div>

              <div>

                <h2>Upload an Image for Analysis</h2>

                <input type="file" accept="image/jpeg,image/png" onChange={handleImageChange} />

                <button onClick={handleImageUpload}>Upload Image</button>

              </div>

              {analysisResult && (

                <div>

                  <h2>Analysis Result:</h2>

                  <p>{analysisResult}</p>

                  {resultImageUrl && <img src={resultImageUrl} alt="Result" />} {/\* Display result image \*/}

                </div>

              )}

            </div>

          )}

        </header>

      </div>

    </GoogleOAuthProvider>

  );

};

export default App;

.App {

  text-align: center;

}

.App-header {

  background-color: #282c34;

  min-height: 100vh;

  display: flex;

  flex-direction: column;

  align-items: center;

  justify-content: center;

  font-size: calc(10px + 2vmin);

  color: white;

}

input[type="file"] {

  margin: 10px;

}

button {

  margin: 10px;

  padding: 10px;

  font-size: 16px;

}