STUDENTS PERFORMANCE PREDICTOR

1. Approach

The approach taken for student performance prediction involves a frontend web interface that collects student scores and a backend machine learning model that processes the input and predicts whether the student will "Pass" or "Fail." The key steps in the implementation are:

- **Frontend Design:**
  - A simple and interactive HTML interface allows users to input Attendance, Assignment Score, Quiz Score, and Final Exam Score.
  - The design features a dark, modern UI with a semi-transparent container and background image for better aesthetics.
  - A button triggers the prediction process when clicked.
- **Backend Prediction Model:**
  - The frontend communicates with a Flask-based backend using the Fetch API.
  - The backend is expected to run a trained k-Nearest Neighbours (KNN) model to predict performance.
  - The prediction result is received as a JSON response.
- **Dynamic Display of Results:**
  - The "Prediction:" text is initially hidden and appears only after a prediction is made.
  - The result is dynamically updated with "Pass" displayed in green and "Fail" in red to enhance readability.

2. Results

- The user interface is functional and visually appealing.
- The input fields allow users to enter relevant scores smoothly.
- The system correctly communicates with the backend and displays predictions dynamically.
- The styling improvements make it clear whether the student is passing or failing.
- The result visibility logic works correctly, showing predictions only when data is entered.

3. Challenges Faced

Despite the success, several challenges were encountered:

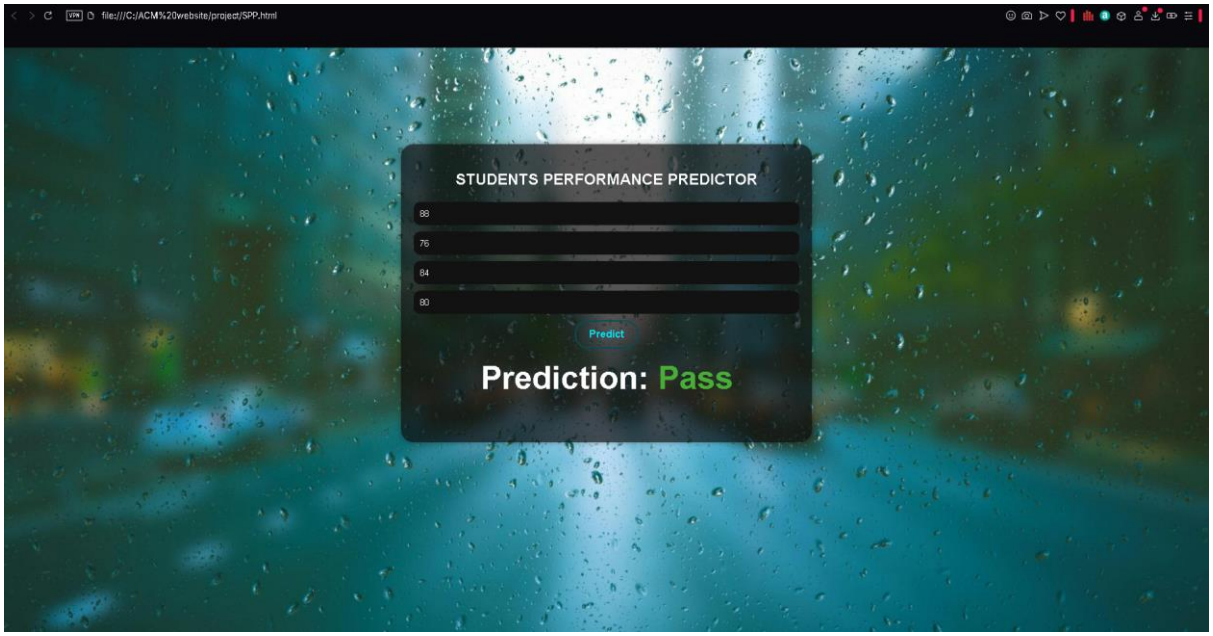
1. **Styling Issues for Prediction Text:**
    - Initially, the "Prediction:" label was always visible, even before any prediction was made. This was fixed by hiding it by default and displaying it only when a result is available.
    - Ensuring the "Prediction:" text remains neutral while highlighting "Pass" in green and "Fail" in red required additional styling.
2. **Frontend-Backend Communication Issues:**
    - Errors occurred when fetching predictions due to incorrect endpoint configurations.
    - Implementing proper error handling was necessary to display a meaningful message when the server was unreachable.
    - The backend should now be running at http://127.0.0.1:5000/.
    - Ensure that the frontend is making requests to this backend to receive predictions.
3. **User Experience (UX) Improvements:**
    - Ensuring the input fields were clear and responsive.
    - Providing instant feedback through colour changes for prediction results.

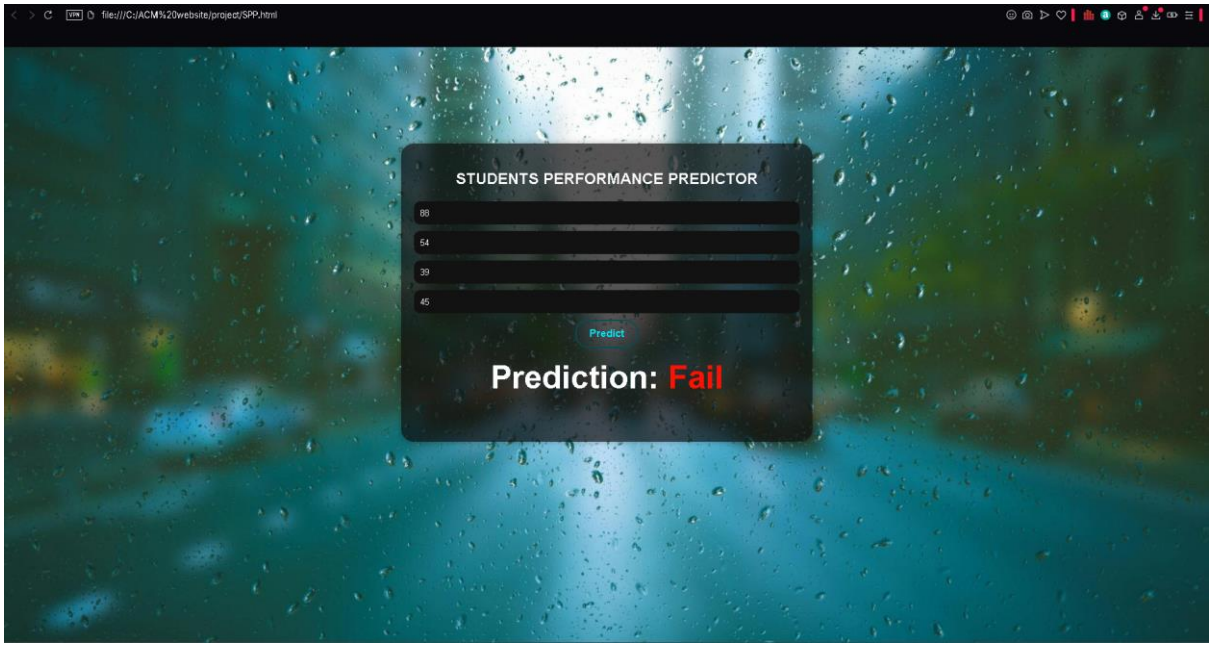
4. Conclusion

The student performance predictor is a functional web-based application with a clean UI and dynamic feedback system. It successfully integrates frontend and backend components to provide real-time predictions. Future improvements could include additional performance metrics, a more advanced ML model, and a better user experience with animations.

```python
from flask import Flask, request, jsonify
from flask_cors import CORS
import pandas as pd
import numpy as np
import os
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier

app = Flask(__name__)
CORS(app)  # Enable CORS for all routes

# Check if dataset exists
DATASET_PATH = 'student_performance_data.csv'
if not os.path.exists(DATASET_PATH):
    raise FileNotFoundError(f"Dataset not found: {DATASET_PATH}")

# Load dataset
df = pd.read_csv(DATASET_PATH)

# Preprocessing
FEATURES = ['Attendance (%)', 'Assignment Score', 'Quiz Score', 'Final Exam Score']
TARGET = 'Pass/Fail'

if not all(col in df.columns for col in FEATURES + [TARGET]):
    raise ValueError("Dataset does not contain the required columns.")

X = df[FEATURES]
y = df[TARGET]

# Normalize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Train KNN model dynamically based on dataset size
k_value = min(5, len(X_train) // 2)  # Avoid too large k for small datasets
knn = KNeighborsClassifier(n_neighbors=max(1, k_value), metric='euclidean')
knn.fit(X_train, y_train)
@app.route('/predict', methods=['POST'])
def predict():
    try:
        data = request.json
        required_fields = ['attendance', 'assignment_score', 'quiz_score', 'final_exam_score']

        # Validate input
        if not all(field in data for field in required_fields):
            return jsonify({'error': 'Missing input fields'}), 400

        try:
            input_values = [float(data[field]) for field in required_fields]
        except ValueError:
            return jsonify({'error': 'Invalid input values'}), 400

        # Transform input for model
        input_data = scaler.transform([input_values])
        prediction = knn.predict(input_data)[0]

        return jsonify({'predicted_result': 'Pass' if prediction == 1 else 'Fail'})
    except Exception as e:
        return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

```html
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Performance Prediction</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 0; padding: 0; text-align: center;
            background-image: url("https://images.pexels.com/photos/531880/pexels-photo-531880.jpeg?cs=srgb&dl=pexels-pixabay-531880.jpg&fm=jpg");
            background-position: center; background-size: cover; }
        .container { width: 600px; box-shadow: 0 0 10px rgba(0,0,0,0.3); padding: 20px; border-radius: 20px;
            margin: 8% auto; text-align: center; background: rgba(14, 13, 13, 0.7); color: white; }
        input { display: block; border-radius: 10px; padding: 10px; margin: 10px 0; width: 100%; border: none; }
        input:hover { background: rgba(255, 255, 255, 0.3); color: #000; box-shadow: 0 0 10px #0ef; }
        button { padding: 10px 20px; border-radius: 20px; font-weight: bold;
            border: 2px solid #0ef; background: transparent; font-size: 1rem; color: #0ef; transition: .5s ease; cursor: pointer; }
        button:hover { background: rgba(11, 194, 250, 0.137); color: #000; box-shadow: 0 0 10px #0ef; }
        #result { margin-top: 15px; font-size: 50px; font-weight: bold; display: none;  }
    </style>
</head>
<body>
    <div class="container">
        <h2>STUDENTS PERFORMANCE PREDICTOR</h2>
        <input type="number" id="attendance" placeholder="Attendance (%)" required>
        <input type="number" id="assignment_score" placeholder="Assignment Score" required>
        <input type="number" id="quiz_score" placeholder="Quiz Score" required>
        <input type="number" id="final_exam_score" placeholder="Final Exam Score" required>
        <button onclick="predictPerformance()">Predict</button>
        <p id="result">Prediction: <span id="prediction-text"></span></p>
    </div>
    <script>//javascript
        function predictPerformance() {
            const data = {
                attendance: document.getElementById('attendance').value,
                assignment_score: document.getElementById('assignment_score').value,
                quiz_score: document.getElementById('quiz_score').value,
                final_exam_score: document.getElementById('final_exam_score').value
            };
                fetch('http://127.0.0.1:5000/predict', {
                    method: 'POST',
                    headers: { 'Content-Type': 'application/json' },
                    body: JSON.stringify(data)
                })
                .then(response => {
                    if (!response.ok) {
                        throw new Error('Network response was not ok');
                    }
                    return response.json();
                })
                .then(data => {
                    const resultElement = document.getElementById('result');
                    const predictionText = document.getElementById('prediction-text');

                    predictionText.innerText = data.predicted_result;
                    predictionText.style.color = data.predicted_result === 'Pass' ? 'green' : 'red';

                    // Show the result when prediction is made
                    resultElement.style.display = 'block';
                })
                .catch(error => {
                    document.getElementById('result').innerText = 'Error: Could not reach the server';
                    console.error('Error:', error);
                    document.getElementById('result').style.display = 'block';
                });
            }
    </script>
</body>
</html>
```


file:///C:/ACM%20website/project/SPP.html
STUDENTS PERFORMANCE PREDICTOR
88
76
84
80
Predict
Prediction: Pass


file:///C:/ACM%20website/project/SPP.html
STUDENTS PERFORMANCE PREDICTOR
88
54
39
45
Predict
Prediction: Fail