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 implementation successfully meets the following objectives:

- 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.
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 or graphical results.