Overview:

This chatbot is designed to assess the environmental risk level based on user input about various environmental factors. It uses machine learning to predict the risk level (e.g., Low, Moderate, High) based on the values provided by the user.

Here’s the overall flow:

⸻

1. User Input (Frontend):

• The React frontend is the UI where users can enter values for various environmental parameters. These parameters represent the chemical concentration, air quality index, water toxicity, and soil contamination.

• Fields for Input:

• Chemical Concentration (in parts per million, ppm)

• Air Quality Index (a numerical index)

• Water Toxicity (on a scale from 1 to 10)

• Soil Contamination (on a scale from 1 to 10)

• The user enters these values into a form and clicks the “Predict Risk” button.

⸻

2. Form Submission (Frontend):

• When the user clicks “Predict Risk”, the frontend (React) sends a POST request to the FastAPI backend with the form data. The data includes the four environmental parameters (chemical concentration, air quality index, water toxicity, and soil contamination).

• The API URL for this request is http://127.0.0.1:8000/predict/ (or the deployed URL if hosted).

⸻

3. Backend Processing (FastAPI):

• The backend receives the input data via a POST request at /predict/. The data is passed to a machine learning model for processing.

Machine Learning Model:

• The backend uses a trained machine learning model (in the form of epa\_risk\_model.pkl) to predict the risk level based on the provided inputs.

• The model is built using scikit-learn and was trained on historical data that maps environmental parameters (chemical concentration, air quality index, etc.) to risk levels. It can predict a Low, Moderate, or High risk level based on the features.

• For example:

• If chemical concentration is high and water toxicity is also high, the model will likely predict High risk.

• If all the parameters are at moderate levels, the model might predict Moderate risk.

• The model is loaded and used to make predictions:

model = joblib.load("epa\_risk\_model.pkl")

prediction = model.predict([[chemical\_concentration, air\_quality\_index, water\_toxicity, soil\_contamination]])

4. API Response (Backend to Frontend):

• After the model makes its prediction, the backend sends the risk level back to the frontend.

• The response might look something like this:

{"risk\_level": “Moderate"}

5. Display Prediction (Frontend):

• The frontend receives the response and displays the predicted risk level to the user.

• The result is shown in the UI, like:

• “The predicted risk level is: Moderate.”

⸻

Summary of What Happens Behind the Scenes:

1. Frontend: User enters values (chemical concentration, air quality, water toxicity, and soil contamination).

2. Backend (FastAPI): The frontend sends this data to the backend.

3. Machine Learning: The backend uses a pre-trained machine learning model (epa\_risk\_model.pkl) to predict the environmental risk level.

4. Backend Response: The backend sends the risk level (Low/Moderate/High) back to the frontend.

5. Frontend: The frontend displays the risk level to the user.

⸻

What This Chatbot Is Doing in Summary:

• It acts as an EPA Risk Assessment tool.

• It allows users to input environmental factors, then uses a machine learning model to predict the risk level (Low, Moderate, or High).

• It then displays the prediction to the user.

⸻

Detailed Breakdown of the Chatbot Process:

1. User Input (Frontend):

The frontend (built with React) provides an interface where users can input data about different environmental factors. Here’s what happens in this step:

• The React UI consists of input fields where users enter values for the following factors:

• Chemical Concentration (e.g., amount of a pollutant in the environment)

• Air Quality Index (a numerical value representing the pollution level in the air)

• Water Toxicity (a scale to represent how toxic the water is)

• Soil Contamination (a scale to represent how contaminated the soil is)

• Example input:

• Chemical Concentration: 5 ppm

• Air Quality Index: 120

• Water Toxicity: 6

• Soil Contamination: 4

⸻

2. Form Submission (Frontend):

Once the user has entered their data, they click the “Predict Risk” button. Here’s what happens:

• The handleSubmit function (in the React app) is triggered by the form submission.

• This function:

1. Prevents the default form submission behavior.

2. Collects the values from the form fields (chemical concentration, air quality index, etc.) and packages them into a JSON object.

3. Sends this data to the backend using Axios (an HTTP client).

const handleSubmit = async (e) => {

e.preventDefault();

const response = await axios.post(API\_URL, formData);

setRiskLevel(response.data.risk\_level);

};

Here, API\_URL is the backend URL, and formData is the data that the user entered.

3. Backend Processing (FastAPI):

Once the data is sent to the backend, the FastAPI server takes over:

Machine Learning Model:

• The backend is responsible for loading a pre-trained machine learning model that was trained to assess environmental risk levels based on various parameters.

• The backend receives the data from the frontend (e.g., chemical concentration, air quality index, etc.) and uses it to make a prediction.

Here’s how the model prediction works:

• FastAPI uses scikit-learn, a Python library for machine learning, to load the pre-trained model (epa\_risk\_model.pkl).

• It feeds the user’s input into the model for prediction.

model = joblib.load("epa\_risk\_model.pkl")

prediction = model.predict([[chemical\_concentration, air\_quality\_index, water\_toxicity, soil\_contamination]])

model = joblib.load("epa\_risk\_model.pkl")

prediction = model.predict([[chemical\_concentration, air\_quality\_index, water\_toxicity, soil\_contamination]])

• The model makes a prediction about the risk level based on the input values. This could be a risk level of Low, Moderate, or High, depending on how the input data compares to patterns learned during training.

5. Display Prediction (Frontend):

Once the frontend receives the response from the backend:

• The React app reads the response from the backend (response.data.risk\_level) and stores the predicted risk level in the component’s state (setRiskLevel).

• This predicted risk level is then displayed to the user.

For example, the UI might show:

• “The predicted risk level is: Moderate.”

⸻

The Machine Learning Model:

The core of this chatbot’s functionality is the machine learning model that makes the risk predictions. Here’s a simple breakdown of how it works:

• Training the Model: The model was likely trained using historical data, where each entry includes environmental parameters (e.g., chemical concentration, air quality) and a corresponding risk level (Low, Moderate, High). This data is used to train the model.

• Making Predictions: When a user submits new data, the trained model uses this data to make a prediction about the risk level.

For example:

• If the chemical concentration is high and the water toxicity is also high, the model might predict a High risk.

• If the values are lower or more balanced, it might predict a Moderate or Low risk.

The Full Flow:

1. User Interaction (Frontend):

• The user enters environmental data (chemical concentration, air quality, etc.).

• The React frontend sends this data to the backend via a POST request.

2. Backend Processing:

• The FastAPI backend receives the data.

• The backend loads the pre-trained model and feeds the input data to it.

• The model predicts the risk level (Low/Moderate/High).

3. Response and Display:

• The backend sends the predicted risk level back to the frontend.

• The frontend displays the result to the user (e.g., “Moderate Risk”).

⸻

In Summary:

This chatbot is a Risk Assessment Tool that:

• Takes environmental data from the user.

• Sends the data to a backend that uses a machine learning model to predict the environmental risk.

• Displays the predicted risk level (Low, Moderate, High) to the user.

⸻

Key Components:

• Frontend: Built with React to provide an interactive user interface.

• Backend: Built with FastAPI to handle API requests and run the machine learning model.

• Machine Learning: A pre-trained model makes predictions based on user input.