The product I am developing is a price predictor designed to address the challenge of fluctuating sardine fish can prices across different regions, seasons, and dates. It is targeted towards consumers and retailers who need to make informed purchasing decisions based on these price variations. This tool stands out because it provides tailored predictions for five different sardine brands, accounting for regional differences (North, South, East, West) and seasonal impacts (Winter, Summer, Spring, Autumn), as well as daily fluctuations, making it a comprehensive and precise solution for price forecasting in this niche market.

The high-level architecture diagram for your Price Predictor App:

1. Solution Overview:

The Price Predictor App designed to predict the prices of products or services based on historical data, market trends, and other relevant factors. The app will have a user-friendly front-end interface where users can input data and view predictions. The back-end will handle data processing, prediction algorithms, and data storage. The communication between the front-end and back-end will be managed through RESTful APIs.

2. Components and Interactions:

Front-End:

- **User Interface (UI):**
    - **Description:** The main interface where users can input data such as product details, historical prices, and other variables. Users can also view the predicted prices.
    - **Technologies:** HTML, CSS, JavaScript, React.js or Angular.js
- **User Authentication Module:**
    - **Description:** Handles user login and authentication to access the app. Ensures secure access to the app.
    - **Technologies:** OAuth, JWT, etc.
- **API Client:**
    - **Description:** Sends user input data to the back-end and retrieves predictions via RESTful APIs.
    - **Technologies:** Axios or Fetch API

Back-End:

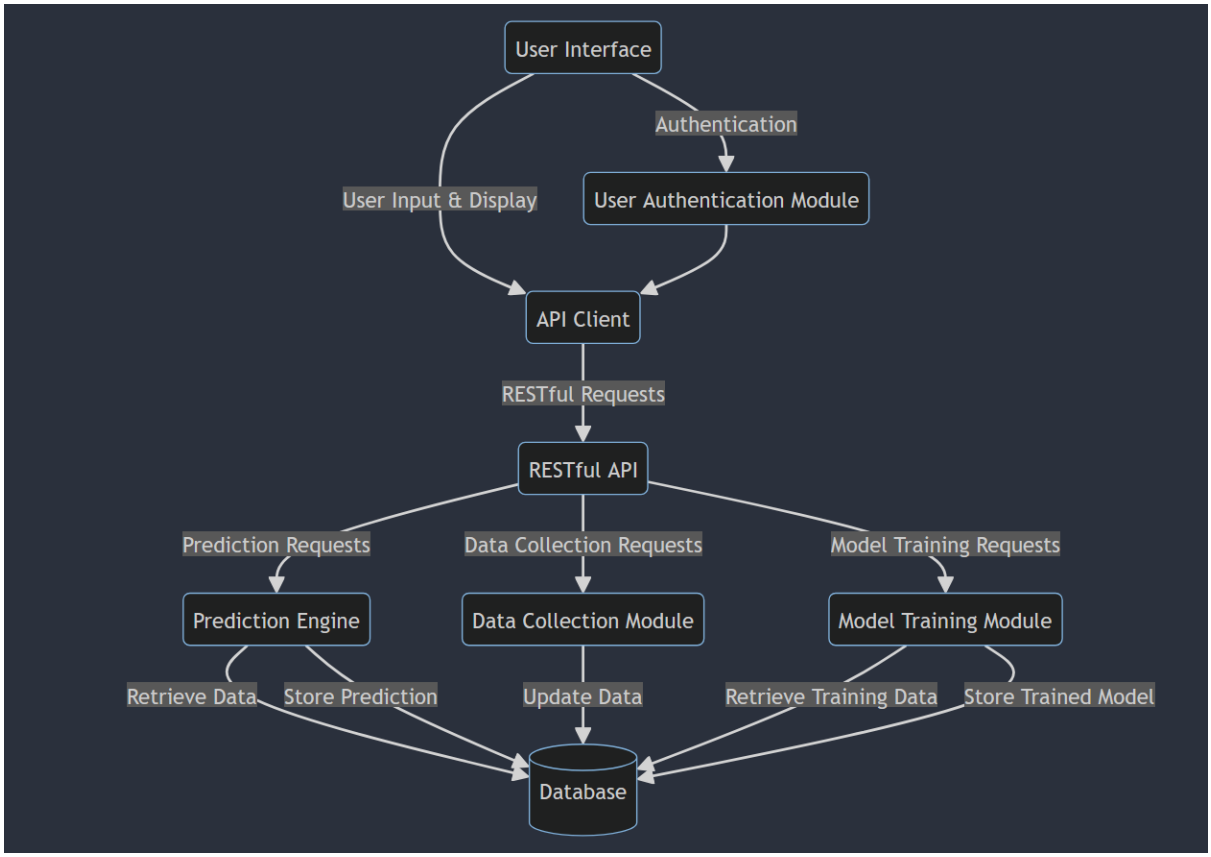
- **RESTful API:**
    - **Description:** Acts as the communication bridge between the front-end and back-end. Handles requests from the front-end, processes them, and returns the necessary data.

    - **Technologies:** Node.js/Express, Flask, or Django
- **Prediction Engine:**
    - **Description:** The core logic that processes the input data and uses machine learning models to predict prices.
    - **Technologies:** Python with scikit-learn, TensorFlow, or PyTorch
- **Data Processing Module:**
    - **Description:** Collects, cleans, and preprocesses the data before feeding it into the Prediction Engine.
    - **Technologies:** Python with Pandas, NumPy
- **Database:**
    - **Description:** Stores historical data, user data, and predicted results. Also stores logs for audit and debugging purposes.
    - **Technologies:** SQL Database (PostgreSQL, MySQL) or NoSQL Database (MongoDB)
- **Data Collection Module:**
    - **Description:** Fetches real-time data from external sources, such as market trends, competitor prices, etc.
    - **Technologies:** Python with Beautiful Soup, Scrapy, or API Integration
- **Model Training Module:**
    - **Description:** Responsible for training the machine learning models used in the Prediction Engine.
    - **Technologies:** Python with scikit-learn, TensorFlow, or PyTorch

Communication Flow:

- **User Interface (UI) ↔ API Client:**
    - Users interact with the UI, and the UI sends requests via the API Client to the back-end.
- **API Client ↔ RESTful API:**
    - The API Client sends user requests (like new data for prediction) to the RESTful API, which processes the requests and interacts with the Prediction Engine or the Database as needed.
- **RESTful API ↔ Prediction Engine:**
    - The RESTful API forwards the input data to the Prediction Engine, which processes the data and returns the predicted prices.

- **Prediction Engine ↔ Database:**
    - The Prediction Engine may need to retrieve historical data or store prediction results in the Database.
- **RESTful API ↔ Data Collection Module:**
    - Periodically, the RESTful API can trigger the Data Collection Module to update the data in the Database.
- **Model Training Module ↔ Database:**
    - The Model Training Module retrieves data from the Database to train the prediction models and may store the trained models back in the Database.



5. Legend/Explanation:

- **UI:** Represents the front-end interface.
- **API Client:** Manages communication between the UI and the RESTful API.
- **RESTful API:** Manages communication between the front-end and back-end components.
- **Prediction Engine:** The core component for making price predictions.
- **Data Collection Module:** Collects external data sources for predictions.
- **Model Training Module:** Trains machine learning models for predictions.
- **Database:** Stores all relevant data including user data, historical data, and predictions.