Sending sensor data to Firebase Cloud and fetching data from a Django web application involves integrating two different technologies. Here's a step-by-step guide on how to achieve this:

Sending Sensor Data to Firebase Cloud:

1. **Set Up Firebase:**
    - Create a Firebase project and set up a Realtime Database.
    - Obtain the Firebase project's API key and database URL.
2. **Microcontroller Integration:**
    - Write code on your microcontroller (e.g., Arduino, Raspberry Pi) to read data from the sensors.
    - Use a suitable library to send HTTP requests (POST) to Firebase's REST API with the collected data.
    - Include the API key in the headers of your requests.
3. **Send Data to Firebase:**
    - Parse the sensor data and format it as JSON.
    - Send the JSON data to Firebase's Realtime Database using the appropriate endpoint URL.

Fetching Data from Django Web Application:

1. **Set Up Django Web Application:**
    - Create a Django project and app for your child health monitoring system.
    - Set up models to define the structure of your data.
2. **Connect to Firebase:**
    - Install the **firebase-admin** package in your Django project.
    - Initialize the Firebase Admin SDK with the credentials obtained from Firebase.
3. **Fetch and Display Data:**
    - In your Django views, use the Firebase Admin SDK to connect to the Realtime Database.
    - Retrieve data from Firebase using the appropriate endpoint URL.
    - Process the fetched data and pass it to your Django templates.
4. **Create API Endpoints (Optional):**
    - If you want to fetch data dynamically via APIs, create API endpoints using Django REST Framework.
    - Design serializers to format the data fetched from Firebase.
5. **Integrate Front-End:**

- In your Django templates, use HTML, CSS, and JavaScript to display the fetched data in a user-friendly format.
- Utilize AJAX or Fetch API to asynchronously fetch data and update the UI without page reloads.

For Fetching Historical Data:

1. **Firebase Querying:**
    - Use Firebase's querying capabilities to retrieve historical data based on time ranges.
    - Create appropriate query conditions to filter data within specific time periods.
2. **Django Data Processing:**
    - In your Django application, implement views or API endpoints that fetch historical data.
    - Process the fetched data and present it to users through your web application.

Remember that security is crucial when dealing with sensitive health data. Ensure proper authentication and authorization mechanisms are in place to protect the data in transit and at rest.

This integration requires programming skills in both microcontroller programming (for sensor data sending) and Django development (for web application and data fetching). Refer to the official documentation of Firebase, Django, and related libraries for detailed instructions and best practices.