For your SmartSleep UI on a Raspberry Pi 5, you have several options for frameworks and servers to store and display sleep data effectively. Here are some suggestions:

**Backend Storage**

1. **SQLite**: Lightweight and easy to set up, SQLite can handle small to medium datasets locally on the Raspberry Pi. It's great for logging sleep data.
2. **InfluxDB**: If you're dealing with time-series data (like EEG, EKG, and movement over time), InfluxDB is an excellent choice. It’s optimized for high write and query performance for time-stamped data.
3. **PostgreSQL**: A more robust option if you anticipate needing more complex queries or relationships between different types of data.

**Backend Framework**

1. **Flask (Python)**: A lightweight web framework that’s easy to set up and can handle your data interactions. You can build a REST API to serve data to your UI.
2. **FastAPI (Python)**: Similar to Flask but more modern and faster, FastAPI is great for building APIs with automatic documentation.
3. **Node.js (Express)**: If you prefer JavaScript, using Node.js with Express can be a great way to handle backend logic and data serving.

**Frontend Framework**

1. **React**: A popular library for building user interfaces. It’s component-based, making it easier to manage different parts of your UI.
2. **Vue.js**: Another progressive framework that's simple to learn and can help create interactive UIs.
3. **D3.js**: If you need to visualize data (like EEG or EKG readings), D3.js is powerful for creating custom visualizations.

**Displaying Data**

* **Chart.js**: For simpler data visualizations, Chart.js can integrate well with both React and Vue.
* **Bootstrap or Tailwind CSS**: For styling and responsive design.

**Additional Considerations**

* **WebSocket**: If you want real-time data updates (for instance, live EEG readings), consider implementing WebSocket support.
* **Docker**: To manage dependencies and ensure your app runs consistently on the Raspberry Pi.

**Summary**

* **Storage**: SQLite or InfluxDB for simplicity; PostgreSQL for complexity.
* **Backend**: Flask or FastAPI for Python; Express for Node.js.
* **Frontend**: React or Vue for UI; D3.js for data visualization.

Combining these tools should give you a solid foundation for your SmartSleep system!

Given your project’s requirements, Django could be a very suitable choice for your SmartSleep system. Here’s why:

**1. Data Management:**

* With multiple sensors collecting data (EEG, EKG, gyroscope, microphone), you’ll likely need a structured way to store and manage that data. Django’s ORM makes it easier to define your data models and interact with the database.

**2. Built-in Features:**

* Django provides many built-in features like user authentication, an admin interface for managing data, and security measures. This can help you avoid reinventing the wheel and let you focus on your application logic.

**3. Scalability:**

* If your project grows or if you decide to add more features later (like integrating machine learning for sleep analysis), Django is designed to scale with you.

**4. Community and Resources:**

* Since you’re inexperienced, Django’s extensive documentation and community support can be valuable as you learn. There are many tutorials and resources available specifically for beginners.

**5. Handling Multiple Data Types:**

* With various types of data (sensor readings, sound files), Django can help you organize and process this information effectively. You can define models for each data type and use Django’s views to handle the logic for processing and displaying that data.

**6. Frontend Integration:**

* You can easily integrate Django with frontend frameworks like React or Vue, or you can build a basic frontend using Django’s templating engine if you prefer a simpler setup to start with.

**Conclusion:**

Given the complexity of your project—handling various data types, needing real-time processing, and the potential for future expansion—Django is a solid choice. It will help you manage the complexity while providing a structured way to learn web development. Starting with Django might have a steeper learning curve than Flask, but the long-term benefits could outweigh that, especially as your project scales.