**Overview**

This project is a full-stack web application that uses a Django backend (for the server and API) and a React frontend (for the user interface). We use Docker and Docker Compose to make it easy to run everything together without a complicated setup process.

**Backend Requirements**

* **Django (Version 5.0.8)**: The core Python web framework for building the server-side logic.
* **Django REST Framework (Version 3.14.0)**: Simplifies building RESTful APIs in Django.
* **Django REST Framework SimpleJWT (Version 5.2.2)**: Provides JSON Web Token (JWT) authentication support.
* **django-cors-headers (Version 3.14.0)**: Allows cross-origin resource sharing between the frontend and backend.
* **django-social-auth (Version 0.7.28)**: Enables social authentication (e.g., logging in with Google, Facebook, etc.).
* **pyodbc (Version 4.0.35)**: Allows Django to communicate with ODBC databases, which might be necessary for certain database backends.
* **django-rest-framework-jsonapi (Version 6.0.1)**: Provides JSON:API support for DRF.
* **django-extensions (Version 3.2.1)**: Adds extra Django management commands and tools to ease development.
* **xarray (Version 2023.8.0)** and **xarray-datatree (Version 0.2.1)**: Useful for handling and analyzing multi-dimensional labeled data.
* **tqdm (Version 4.68.0)**: A utility for progress bars in Python scripts.
* **nptdms (Version 1.7.3)**: Enables reading and working with TDMS files.
* **requests (Version 2.31.0)**: A simple HTTP library to handle backend requests if needed.

**Frontend Requirements**

* **React**: A JavaScript library for building user interfaces.
* **React-DOM**: Integrates React with the DOM (browser environment).
* **React Router DOM**: Handles routing on the client-side, allowing for single-page application behavior.
* **Axios**: A popular HTTP client for making API requests from the frontend to the backend.
* **Vite** and **@vitejs/plugin-react**: A fast build tool and plugin setup for developing and bundling the React application efficiently.

**How Dependencies Are Installed**

* **Backend Dependencies**:  
  These are handled via pip based on the requirements listed at the top. Once you run the Docker commands (detailed below), Docker will install these dependencies automatically inside the container.
* **Frontend Dependencies**:  
  The frontend uses Node.js and npm. The package. Json will list and manage dependencies like React, React Router, Axios, and Vite. When you run the frontend container or run npm install (or yarn install) locally, these dependencies are installed automatically.

**How Do I Get This Running on My Computer?**

1. **Install Docker and Docker Compose:**Make sure you have Docker and Docker Compose installed on your machine. These tools allow you to run the backend and frontend in separate containers, so you don’t have to set up everything manually.
2. **Build the Images:**Open a terminal (or command prompt) in the project’s root directory and run:

***docker-compose build --build-arg PIP\_DEFAULT\_TIMEOUT=100***

This command will build the Docker images, installing all the necessary backend dependencies. The --build-arg PIP\_DEFAULT\_TIMEOUT=100 just gives the installer more time if your network is slow.

1. **Start the Containers:**

***docker-compose up***

This will start both the backend and the frontend containers. The backend API will run using Django’s development server, and the frontend will be served via Vite’s development server.

**Frontend**:  
Once everything is running, open your web browser and navigate to:

<http://localhost:5173>

**Backend**:  
The Django backend is exposed at:

<http://localhost:8000>

**Summary**

In short, this setup provides a convenient way to develop and test a Django/React full-stack application. The Docker configuration ensures consistency and avoids environment setup complexities. Once you’ve built and run the Docker containers, you can interact with the frontend through http://localhost:5173 and with the backend through http://localhost:8000. This approach offers a straightforward workflow that allows you to focus more on coding and less on environmental management.