5. FastAPI Backend Setup

The backend for the SOC Co-Pilot project is built using FastAPI, a modern, high-performance web framework for building APIs with Python. This backend will handle the logic for processing data, interacting with the PostgreSQL database, and serving data to the frontend applications.

5.1. Project Initialization

To set up the FastAPI backend, create a dedicated directory for it within your project structure and navigate into it:

```
mkdir -p ~/soc-copilot/backend
cd ~/soc-copilot/backend
```

This command creates the `backend` directory inside your `soc-copilot` project folder and changes your current directory to it. This is where all your FastAPI application files, including Python scripts, dependencies, and configurations, will reside.

5.2. Initial Setup (Conceptual)

After creating the directory, the typical next steps involve:

1. **Create a Python Virtual Environment**: It is highly recommended to create a virtual environment to manage project dependencies:

   `bash python3 -m venv venv source venv/bin/activate`

2. **Install FastAPI and Uvicorn**: Install the necessary libraries. `uvicorn` is an ASGI server that will run your FastAPI application:

   `bash pip install fastapi uvicorn`

3. **Install PostgreSQL Driver**: Install a PostgreSQL adapter for Python, such as `psycopg2` or `asyncpg`:

   ```bash pip install psycopg2-binary
   ```

or pip install asyncpg

```

4. **Create Your First FastAPI Application**: Create a file (e.g., `main.py`) and add basic FastAPI code:

```python

# main.py

from fastapi import FastAPI

app = FastAPI()

@app.get("/") async def read_root(): return {"message": "Hello, SOC Co-Pilot Backend!"} ```

5. **Run the FastAPI Application**: You can run your application using Uvicorn:

`bash uvicorn main:app --reload`

This will start the development server, typically accessible at `http://127.0.0.1:8000`.

### 5.3. Database Integration (Conceptual)

Integrating with the PostgreSQL database involves:

- **Database Connection**: Establishing a connection to the `soc_copilot` database using the `soc_admin` user.
- **SQLAlchemy/Alembic**: Using an ORM like SQLAlchemy for database interactions and Alembic for database migrations.
- **API Endpoints**: Developing API endpoints to receive data from the AI processing pipeline and push it into the database.