Project Overview:

This project focuses on developing a Python-based web scraping solution with the scraped data being stored in a PostgreSQL database and displayed on a static website. FastAPI is used to create an API that allows the website to access the data from the PostgreSQL database.

Technical Stack:

- **Programming Language:** Python
- **Web Framework:** FastAPI
- **Database:** PostgreSQL
- **Frontend:** Static HTML, JavaScript
- **Hosting:** FastAPI static files

Project Components:

1. **Web Scraping:**
    - **Objective:** Extract data from target websites (specify the source websites if needed).
    - **Tools:** Python libraries such as BeautifulSoup and Requests were used for scraping the content.
    - **Data Storage:** The scraped data was stored in a PostgreSQL database for further access and manipulation.
2. **API Development:**
    - **Framework:** FastAPI was used to build the API.
    - **Functionality:** The API provides endpoints to access the scraped data stored in PostgreSQL. These endpoints allow the static website to query and retrieve the required data.
    - **Code to Start the Server:** The FastAPI server can be started using the command:
      lua
      Copy code
      ```
      uvicorn apis:app --reload
      ```
    -
    - **Security:** Basic security features such as input validation and error handling were implemented to ensure the reliability of the API.
3. **Static Website:**
    - **Hosting:** The static website is hosted using FastAPI's static files feature.
    - **Frontend Development:** The frontend was built using HTML and JavaScript. JavaScript is used to send API requests to FastAPI and display the data dynamically on the website.
    - **User Interaction:** The website allows users to view the data retrieved from the PostgreSQL database, providing an interactive and responsive interface.
4. **Database Management:**

- **Database:** PostgreSQL was chosen for its reliability and ability to handle large datasets.
- **Schema Design:** The database schema was designed to store and manage the scraped data efficiently.
- **Integration with FastAPI:** The API interacts with PostgreSQL using Python's SQLAlchemy or similar ORM (specify if different).

Conclusion:

The project successfully implemented a full-stack solution that integrates Python web scraping, data storage in PostgreSQL, and frontend access via a FastAPI-exposed API. This project showcases the ability to combine backend and frontend technologies to create a dynamic and data-driven web application.