# Suggested prototype

**1. Project Structure:**

```
sp500_client/
├── domain/
│   ├── entities/
│   │   └── sp500_data.py
│   ├── services/
│   │   ├── data_processing_service.py
│   │   └── interfaces.py
│   └── value_objects/
│       └── data_point.py
├── infrastructure/
│   └── database.py
├── application/
│   └── data_retriever.py
├── presentation/
│   ├── cmd/
│   │   └── main.py
│   └── controller/
│       └── api_client.py
├── config.py
├── requirements.txt
└── docker-compose.yml
```

**2. Docker Compose (docker-compose.yml):**

```yaml
version: "3.8"
services:
  db:
    image: postgres:14
    restart: always
    environment:
      POSTGRES_USER: sp500_user
      POSTGRES_PASSWORD: sp500_password
      POSTGRES_DB: sp500_db
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data

volumes:
  postgres_data:
```

**3. Configuration (config.py):**

```
import os

API_KEY = os.getenv("ALPHA_VANTAGE_API_KEY")
API_URL = "https://www.alphavantage.co/query?
function=TIME_SERIES_INTRADAY&symbol=SPY&interval=1min&apikey=" + API_KEY

DB_HOST = "db"
DB_USER = "sp500_user"
DB_PASSWORD = "sp500_password"
DB_NAME = "sp500_db"
```

**4. Domain Layer (domain/):**

- **domain/entities/sp500_data.py:**

```
from dataclasses import dataclass
from datetime import datetime
from typing import Dict, Any

@dataclass
class SP500Data:
    timestamp: datetime
    data: Dict[str, Any]
```

- **domain/value_objects/data_point.py:**

```
#If you needed to use this value object, for example, to validate the
data.
#from dataclasses import dataclass
#from decimal import Decimal

#@dataclass
#class DataPoint:
#    open: Decimal
#    high: Decimal
#    low: Decimal
#    close: Decimal
#    volume: int
```

- **domain/services/interfaces.py:**

```
from typing import Protocol
from domain.entities.sp500_data import SP500Data

class DataProvider(Protocol):
    def fetch_data(self) -> SP500Data:
```

```
        ...

class DataStorage(Protocol):
    def store_data(self, data: SP500Data) -> None:
        ...
```

- **domain/services/data_processing_service.py:**

```
from domain.services.interfaces import DataProvider, DataStorage
from domain.entities.sp500_data import SP500Data

class DataProcessor:
    def __init__(self, data_provider: DataProvider, data_storage:
DataStorage):
        self.data_provider = data_provider
        self.data_storage = data_storage

    def process_and_store(self) -> None:
        data = self.data_provider.fetch_data()
        if data:
            self.data_storage.store_data(data)
```

**5. Infrastructure Layer (infrastructure/):**

- **infrastructure/database.py:**

```
import psycopg2
import json
from domain.entities.sp500_data import SP500Data
from domain.services.interfaces import DataStorage
from config import DB_HOST, DB_USER, DB_PASSWORD, DB_NAME

class PostgresStorage(DataStorage):
    def store_data(self, data: SP500Data) -> None:
        try:
            conn = psycopg2.connect(
                host=DB_HOST,
                user=DB_USER,
                password=DB_PASSWORD,
                dbname=DB_NAME,
            )
            cur = conn.cursor()
            cur.execute(
                "INSERT INTO sp500_data (timestamp, data) VALUES (%s,
%s::jsonb)",
                (data.timestamp, json.dumps(data.data)),
            )
            conn.commit()
            cur.close()
```

```
            conn.close()
        except psycopg2.Error as e:
            print(f"Database Error: {e}")
```

**6. Application Layer (application/):**

- **application/data_retriever.py:**

```
from domain.services.data_processing_service import DataProcessor
from presentation.controller.api_client import AlphaVantageAPIClient
from infrastructure.database import PostgresStorage

def retrieve_and_store_data() -> None:
    api_client = AlphaVantageAPIClient()
    db_storage = PostgresStorage()
    processor = DataProcessor(api_client, db_storage)
    processor.process_and_store()
```

**7. Presentation Layer (presentation/):**

- **presentation/cmd/main.py:**

```
import schedule
import time
import os
from application.data_retriever import retrieve_and_store_data

def run_scheduler():
    schedule.every(1).minutes.do(retrieve_and_store_data)
    while True:
        schedule.run_pending()
        time.sleep(1)

if __name__ == "__main__":
    if "ALPHA_VANTAGE_API_KEY" not in os.environ:
        raise Exception("ALPHA_VANTAGE_API_KEY environment variable is not
set")
    run_scheduler()
```

- **presentation/controller/api_client.py:**

```
import requests
import datetime
from domain.entities.sp500_data import SP500Data
from domain.services.interfaces import DataProvider
from config import API_URL
from typing import Dict, Any
```

```
class AlphaVantageAPIClient(DataProvider):
    def fetch_data(self) -> SP500Data:
        try:
            response = requests.get(API_URL)
            response.raise_for_status()
            data = response.json()
            time_series = data.get("Time Series (1min)")
            if time_series:
                latest_minute = next(iter(time_series))
                minute_data: Dict[str, Any] = time_series[latest_minute]
                timestamp =
datetime.datetime.fromisoformat(latest_minute.replace(" ", "T"))
                return SP500Data(timestamp=timestamp, data=minute_data)
            else:
                return None
        except requests.exceptions.RequestException as e:
            print(f"API Error: {e}")
            return None
        except Exception as e:
            print(f"Unexpected Error: {e}")
            return None
```

**8. Requirements (requirements.txt):**

```
requests
psycopg2-binary
schedule
```

**To Run:**

1. **Start PostgreSQL:** `docker-compose up -d`
2. **Install Dependencies:** `pip install -r requirements.txt`
3. **Set API Key:** `export ALPHA_VANTAGE_API_KEY=YOUR_API_KEY`
4. **Run the application:** `python presentation/cmd/main.py`

This prototype is now structured according to DDD best practices, with a clear separation of concerns between the domain, infrastructure, application, and presentation layers.