**Fashion LLM Summarization**

Backend

Components

* Scraper
* FashionServices
* Factory
* Fashion\_AI
* Fashion\_api
* Database
* Common
* Main

Components props:

* Each Scraper inherent from “AbstractScraper” class with scraping functulaity each fashion scraper must implement.  
  inherent classes: AsosScraper, SheinScraper
* Each FashionServices class must implement IFashionService interface. it must hold the

Needed Scraper (composition design pattern), and functionality for db data manipulation.  
inherent classes: AsosService, SheinService.

* AbstractFactory – defined one static method name –“build” which must return a class that implement the IFashionService interface.

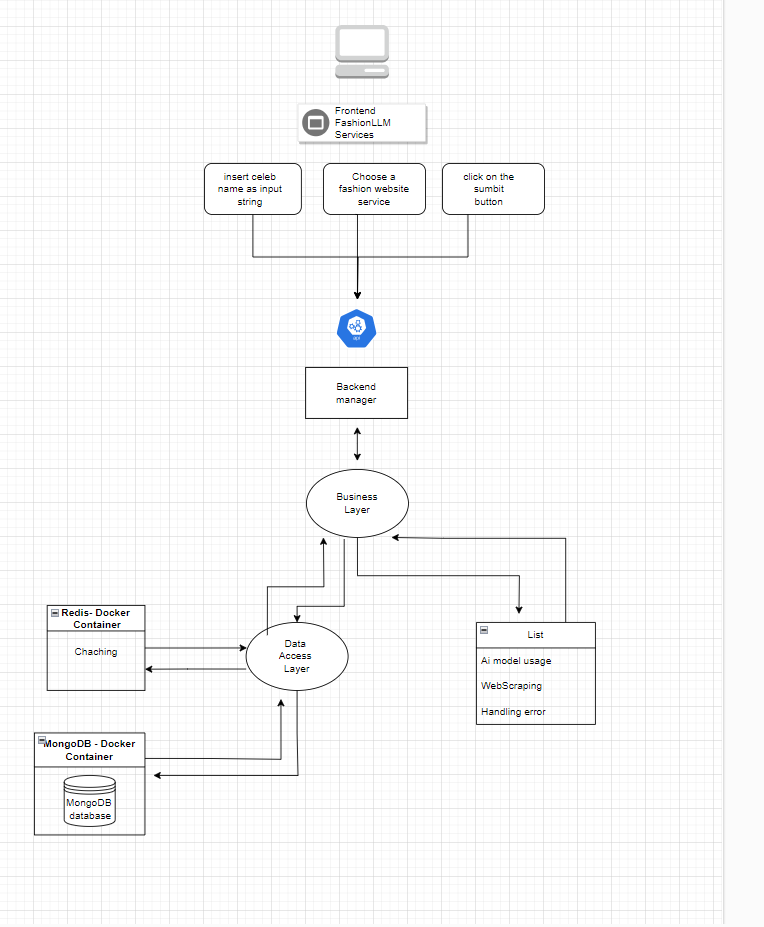
inherent classes: AsosFactory, SheinFactory.  
in the Business layer we use the “FashionServiceFacoty” with the singleton design pattern  
to init the correct FashionService class by the service name we received as an input.

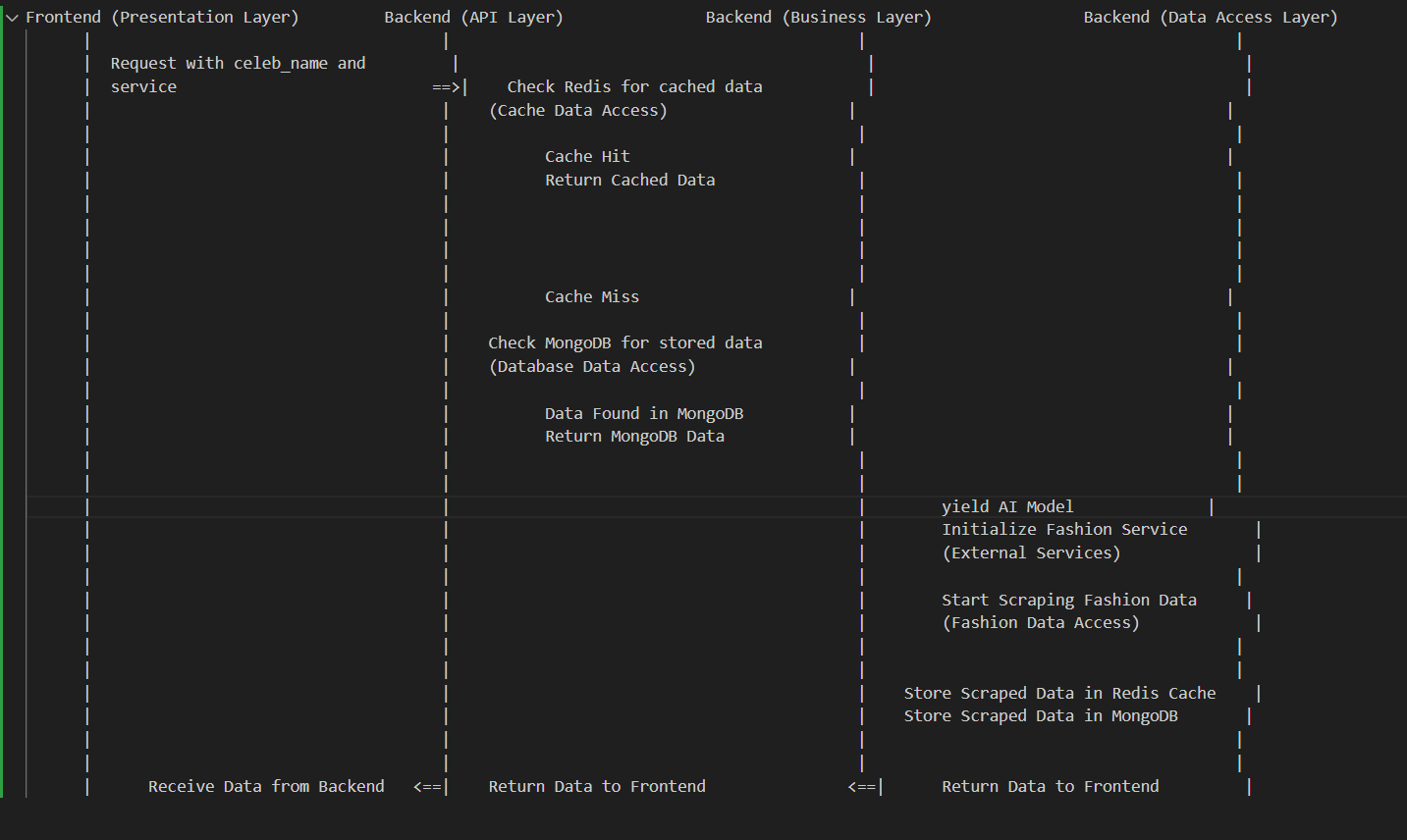
* Fashion\_Ai: class using LLM (such as openAi and hugging face) to train and init the ai models. The init parameters are config by the “prompt\_template.py” file
* DataBase – config the structure of the mongo db we want. Also define the Pydantic and SQL entity structure for user registration (not fully implemented yet)
* Common - 2 databases providers for both the Redis db (caching) and MongoDb (storage)

It connect to the docker container server and is responsible for the data manipulation on it.

* Fashion\_Api – 2 main files. Models.py- which define the pydantic structure for the data will be working on and return to the front end. Endpoits.py. receive an http get request from the frontend with “service” as a path parameter and a “celeb\_name” :str as a query parameter in the request. Then it will execute all the business layer and data access layer and will return the needed data to the backend (or the specific error exception)

**Frontend – Backend integration diagram:**

****

**Flow Diagram:**

**Significance of Layers:**

**Presentation Layer**: This is where user interactions take place. It sends requests to the backend API.

**API Layer:** It routes requests from the frontend to the appropriate parts of the backend, including the business layer.

**Business Layer**: This is the core of your application, handling caching, data retrieval, AI model usage, and fashion service scraping. It orchestrates the various components.

**Data Access Layer:** Responsible for interacting with data storage systems, both caching and long-term storage, using Redis and MongoDB.

In this architecture, the Business Layer plays a crucial role in coordinating the different components, implementing business logic, and handling the AI model and fashion service. The Data Access Layer is responsible for interacting with data storage systems, abstracting away the details of how data is stored and retrieved.

Remember that while the diagram and breakdown provide a high-level overview, actual implementations may vary based on the technologies and design patterns you choose to use.