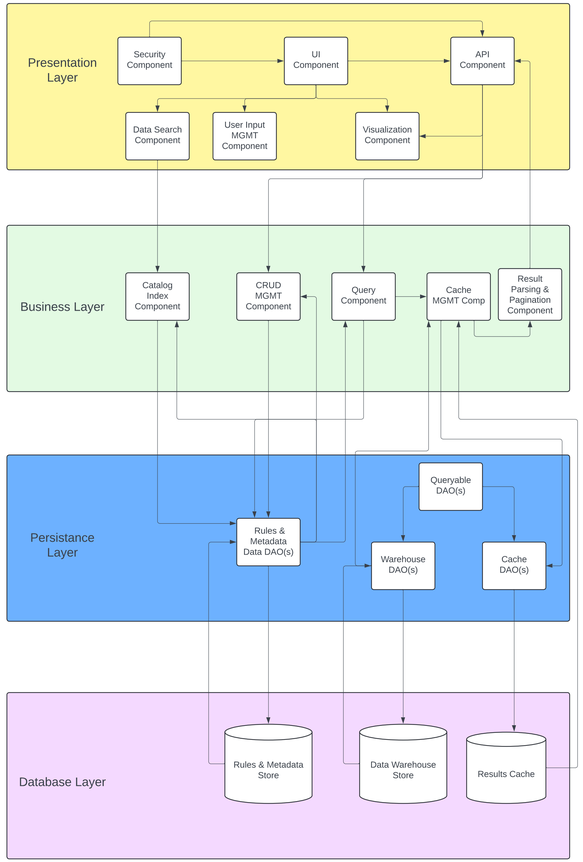
Architecture desicions and reasoning:

The proposed architecture for the "How People Shop" web app is a layered architecture which helps us seggregate the components in different layers. It makes the development easier and each component can be individually constructed.

Since, we don't have many services and the essentially, this product provides a web view for a data product, hence this app can be developed in a monolithic style where we have multiple stores which are storing different types of data but often times, input from one store will be required to query the data from the other store. Hence, Service oriented architecture or a microservices architecture will be an overkill for the simple use case of a web based data product.



Presentation Layer: This layer consists of user facing components. It is responsible for taking users' input, providing output to users either in the form of visualizations or data from the API calls. Security is also plugged into this layer.

Security Component: This component is responsible for the security, which includes identity and access management and is plugged into the enterprise security management. Each request to access the UI or data through the APIs has to pass through the security component to access whether a request has appropriate rights or not.

UI Component: The UI component is the main entrypoint of our web based data product and it consists of multiple child components each having its own responsibility and functionality.
User Input Management Component: This component is ressponsible for taking user input for managing triggers, user defined data and custom input master data.
Visualization Component: This component is responsible for visually representing the data about how people shop in dashboards/charts and easy to understand web pages. This component gets input from the API component which is parsed and paginated according to the specifications provided by the visualization request which makes the information easy to render.
Data Search Component: This component enables search on the data catalogue which is stored in the metadata data store. The cataloging of data enables the users to find the exact information that they need and the business meaning of the data stored in the facts and dimensions in data warehosue.
API Component: Finally, we have the API component available in the presentation layer which is responsible for requesting the data and receiving responses based of provided specifications. These APIs can also be exposed for external consumers of the data produced by our data product.

Business Layer: This layer contains business logic. All the operations performed by the users or requests done by them are received by this layer from the presentation layer and this layer creates the object representation of all the requested data and also is responsible for parsing and paginating the responses according to the wishes of the consumers.
The **Catalog Index Component** works with the data search component and is a representation of the searchable data from the data catalog or metadata stored in the Rules & Metadata store. The **CRUD Management Component** is responsible for the user inputs regarding trigger rules and user managed data. The **Query Component** is responsible for converting the requests for data into queries which will be forwarded to the **Cache Management Component.** This Component will check first in the cache if the requested data is available and respond with the data, if the data is not available in the cache, this component will forward the request to **Data Warehouse Store** the result will be stored in the cache store. Finally, the **Result Parsing & Pagination Component** is used to parse the data according to the user requirements provided through the API compoonent.

Persistance Layer: This layer consists of ORM with the data stores.

Database Layer: This is where all the data is stored.

Rules & Metadata Store: This store is used for storing all the business rules, user managed data and metadata about the facts and dimensions of the data stored in the data warehouse. This metadata provides the data catalog and is used for providing answers to users' questions on where the data is stored and where can they find the answer of certain questions. This store can be integrated with an industry standard data cataloging tool.
Data Warehouse Store: This is where the large facts and dimensions are stored which help our product scale to size of billions. This store provides fast query mechanisms and performs better for reads rather than writes. This store is only used to query data and the data is inserted in this store via the data pipelines running on our data platform. This warehouse can contain data in different schemas serving different use cases and the metadata provided by the rules store will be used by the query component to correctly access which object from which schema is requested.
Results Cache: This cache is used for storing the data that has already been queried. The **TTL** of the keys against which the data is stored is determined by the rules against data objects stored in the rules store. The store provides extremely fast lookup against keys. Proposedly, the keys are generated based on the user input by using some hashing or serialization technique.