1. Goal

Our goal question: What are the best products for the user based on their skin type and personal preferences?

Skincare is an essential part of many people's daily routine. Considering the major effect that it can have on the skin, it's important that one uses products that are suitable for their skin type to prevent irritation and damage. However, with the fast-growing popularity of the skincare industry and the large amount of products that it releases onto the market every year, it can be difficult to keep track of all the different skincare items and their purposes. The goal of our application, therefore, is to present the user with an overview of products that best fit their needs based on their input. In order to do so, the user is presented with a list of questions. The results of this questionnaire form the base of the application as they are used to retrieve related data from a set of ontologies and datasets. A clear overview of recommended skincare products for the user's skin will be the resulting output. It will inform the user about products' properties which, for instance, include the type of product (moisturizers, cleansers, etc.), the skin type it's suited for, and its brand, price, ranking, and ingredients. In addition to finding recommended products, the application also allows the user to retrieve more information about individual brands and product ingredients. This will help the user understand more of the products that they buy and the brands from which they buy them.

2. Stakeholders

The analysis is intended for skincare companies and people with skin issues. These stakeholders both want the best skincare products. This application can help with their needs by retrieving information, such as information on specific ingredients, products, and price range.

The first stakeholders are skincare companies. Skincare companies want to find products with a good price that will give them maximum profit, however, they also want to give their clients products with high-quality ingredients. This application will give companies an idea of which products would be worth investing in and which products are worth selling to their customers. It is also a huge benefit for the companies because customers will see their products when they are using our application. This is a good way to promote the products they are selling and it will give them a good position in the skincare market. It will also keep companies updated with the current development within the skincare industry. As our application will not be outdated, because it keeps track of the newly developed skincare on the market.

The second stakeholders are people with skin issues. As for people with skin issues, they want to find products that are just right for their skin type. Since skincare products are constantly being improved, it is hard for consumers to stay up to date with the best skincare products. This application will provide these people with an idea on what products to buy depending on their preferences. It will give them recommendations depending on their skin type (oily, dry, normal) which will prepare a list of different brands of the product, the ingredient list, rankings of the product, pricing, and so on.

3. Design

The aim of the design is to lay out the important features that have been selected for the customer, in order to display the essential information to the customer in an efficient manner. As a result, the visualization of the data has been decided to be displayed through a table containing the following features.

a. appropriate skin type or disease for the product (example image)
    i. Extra information about the skin types can be found in the questionnaire that the user is presented with when starting the application. An overview will be given with the properties of different skin types and a small description, making it easier for the user to recognise their own.
b. product name
c. main ingredients of the product
d. usage and effects of the product
e. side effects of the product
f. manufacturer and company of the product
g. customer rating of the product
h. price of the product in euros

The features above have been selected because they are asked and answered in the list of questions that the customer receives. In other words, the results will produce an ontology, which will contain the appropriate information to fill in the data for the questioned features. A table is capable of organizing and arranging complicated data by categorizing them into rows and columns. The arrangement of having certain types of information contained in a specific row or column will allow the customer to retrieve the information that is needed quickly and easily. A visual example of the table can be seen in Table 1. Different visual representations, such as a list would be less suitable, in comparison to a table, due to the fact that it would not be able to organize lengthy pieces of data. Moreover, lists do not contain images, thus when presenting data such as skin type will have to be presented in words. Therefore, the customer most likely will have to search up an example of the skin type to check whether it is the correct skin type that they have. Therefore, to effectively communicate the relevant information to the customers, in this situation a table would be the most appropriate choice.

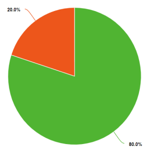
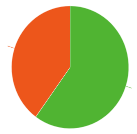
Product Name	**Skin type**	**Ingredients**	**Effects**	**Side effects**	**Company**	**Rating**	**Price in euros**
tea tree cleanser	Greasy, acne	tea tree	reduce acne	dries skin	l'oreal		10.93
coral reef cleanser	Pores	coral reef, sea salt	cleanse pores	burning	ganiay		6.20

Table 1: Example Image of Data Layout Design

4. Identify 2 ontologies

The first ontology we want to use is called the Schema.org[1] ontology. Schema.org contains vocabulary that can be used in applications that publish, discover, and integrate different kinds of data. This ontology has different classes, which include products and services. It is especially useful to online stores with a great diversity of products. We can use this ontology to describe the relevant information about skincare products. By using this we can show the different products with prices, reviews, and quality of the product. Customers will also be able to find related products and see images of each skincare product. Using this ontology will make our application user-friendly because they do not have to search again to find similar products.

The second ontology we want to use is called Product[2] vocabulary. This ontology can be used to represent more information about the skincare products. By using this ontology, the customer will be able to search for products based on brand name, products and packages of products. By combining these two ontologies we will be able to make an application that can help users to find skincare products based on their preferences.

[1] https://schema.org/Product
[2] http://ns.inria.fr/provoc

5. External sources of data

To create instances to use in the analysis, external sources of data have to be extracted. The first dataset, the cosmetics dataset[3], provides almost 1,500 products that include the product name, brand, rank, combination, ingredients, skin preference and price. Another potential dataset is the chemical ingredients dataset[4]. These datasets will be capable of providing instances for the classes or subclasses within the ontology. In other words, it will further advance the quality of the recommendation system for the appropriate product and to satisfy the customer by being able to provide further information about the products, if needed.

DBPedia[5] is a database consisting of subsets of datasets in RDF format that offers various information when it comes to skincare such as pages on different brands[6], and on skincare as a subject itself[7]. All this information is useful to our application since it provides a good starting point when searching for suitable and, therefore, recommended products for the user. Because DBPedia has a SPARQL endpoint[8], it will enable the application to go through a wider range of products and make searching for instances based on the given query more efficient. In addition to its easily navigable database, DBPedia is a piece of open-source software. This means it may be used and redistributed without requiring permission. For these reasons, the use of DBPedia is ideal as it can considerably improve the application's searching mechanism, and the quality and size of the output.

[3] https://www.kaggle.com/kingabzpro/cosmetics-datasets
[4] https://www.kaggle.com/krrai77/chemicals-in-cosmetics
[5] https://www.dbpedia.org/
[6] https://dbpedia.org/page/Clinique
[7] https://dbpedia.org/page/Category:Skin_care
[8] https://dbpedia.org/sparql

6. Domain & scope of the ontology

The explored domain is about skin products, in order to reduce the effort of searching for relevant information, when having to purchase skin products from a customer's point of view. As consumers are being bombarded by advertisements and irrelevant information when searching for skin products, it becomes more obvious that there is an urgent need for an application capable of selecting products based on the customer’s needs. The ontology will include data about the product, customer rating and skin conditions. The application will need to retrieve the appropriate information, based on the question sheet or filter filled in by the user, which will then get processed to be displayed for the user to see. Therefore, the goal of this application is to build an ontology to help retrieve and display the correct output.

7. Methodology used in the construction of the ontology

Having a firm sight of what the main function of the application would be, it was decided that reusing two external ontologies would help create the ontology. The classes, subclasses, instances, relations and properties etc would be given to build up the ontology as a base guide line. In addition, creating relationships and restrictions between the classes will help add more accurate detail when giving the correct output. Moreover, the data points about the product (instances) from the external sources would contribute to being able to provide more relevant information and actual outputs. Once the data has gone through the reasoner any errors or inconsistencies were taken care of to check whether the ontology was working properly.

8. Conceptualization

The domain of our ontology will cover skincare products and will consist of 22 classes. These 21 classes contain 6 main classes and 15 subclasses. The properties for our ontology consists of 3 different DataType properties and 5 different ObjectType properties. The classes, the DataType properties and the ObjectType properties can be seen in figure 1 of the appendix.

Classes:

- ProductBrand
- ProductName
- ProductIngredient
- ProductType
    - Cleanser
    - Moisturizer
    - Eye Cream
    - Face Mask
    - Treatment
    - Sun_Protection
- ProductPurpose
    - Moisturizing
    - Soothing
    - UV_Protection
    - Purifying
- SkinType
    - Combination
    - Dry
    - Normal
    - Oily
    - Sensitive

Object Properties:

- hasPurpose
- isProductType
- isForSkinType
- hasIngredient
- isOfBrand (potentially isBrandOf inverse)

Object type property	**Relation**
hasPurpose	ProductType *hasPurpose* ProductPurpose ProductName *hasPurpose* ProductPurpose
hasProductType	ProductName *hasProductType* ProductType
isForSkinType	ProductName *isForSkinType* SkinType ProductType *isForSkinType* SkinType
hasIngredient	ProductName *hasIngredient* ProductIngredient
isOfBrand	ProductName isOfBrand ProductBrand

Table 1.

As can be seen in Table 1, the object properties connect the classes and the specific instances to each other.

Data type properties:

- hasPrice
- hasRating
- hasName

Data type properties	**Relation**
hasPrice	Product hasPrice "..."
hasRating	Product hasRating "..."
hasName	Product hasName "..." ProductType hasName "..."

Table 2.

As can be seen in Table 2, the datatype properties connect the instances of classes to a particular literal, which could be a string or a number in our ontology. The datatype properties are explained below.

- **hasRating** data type property connects the Product to its rating, thus a positive integer. For instance, Crème de la Mer hasRating "8".
- **hasName** connects the ProductType to its name. For instance, moisturizer hasName Crème de la Mer.
- **hasPrice** connects the ProductName to its price in euros. For instance, Crème de la Mer hasPrice €175.

9. Ontology, class restrictions and other existing ontologies

For the ontology we used the classes and properties we described in the conceptualization. We used these classes to map and link these classes and properties to others in external reusable ontologies. The two external ontologies we reused are the schema.org[9] ontology and the product vocabulary. The schema.org ontology contains relevant classes such as ProductName and ProductBrand. To reuse only the classes we need, we copied the classes we need for our skincare ontology to a seperate ontology. After doing this, we imported this piece of the schema.org ontology into our skincare ontology. The second vocabulary we reused is the Product[10] Vocabulary. We used the classes ProductBrand and package from this vocabulary. The class ProductBrand will be useful to connect the skincare products to their brand. The 3 class restrictions we made are:

1. ProductName hasIngredient some ProductIngredient
2. ProductName isOfBrand some ProductBrand
3. ProductName isForSkinType some SkinType

[9] https://schema.org/Product

[10] http://ns.inria.fr/provoc/v1/provoc_v1.html

10. Integrated external datasets

For the integration of external data, it is important to first note what form the data that is used is in. The application uses various datasets of different formats. The first dataset retrieved from Kaggle is a CSV file[11], while the second dataset from DBPedia is in RDF-format[12].

The CSV file contains a large range of products together with their specific properties which are the type of product (label), the skin type it is suited for, and its brand, name, price, rating, and ingredients. Each row contains all the information about one product, where all of the previously mentioned properties are displayed in its own column.

As the CSV file is in tabular form, it has to be converted to TTL format before the useful information can be integrated in the application's ontology. In order to do so, GraphDB is used alongside its OntoRefine tool. The cosmetics dataset from Kaggle contains columns of which all are relevant to the application and so all columns are maintained. Every value is then added to the ontology as an instance of the corresponding class. Take as an example the class ProductBrand in the ontology which contains all instances from the table column Brand, so *Brand a ProductBrand*. In addition to classes, some properties (e.g. price and rating) will also be converted to literals to simplify the later process of finding matching instances or data. Furthermore, mappings will be created between the data in the table using the object properties defined in the ontology. An example of this would be *Name (a ProductName) isOfBrand Brand (a ProductBrand).* As for the third potential chemical ingredients dataset[13], the only useful column will be the one containing the names of chemicals. These chemicals can be added as

[11] https://www.kaggle.com/kingabzpro/cosmetics-datasets
[12] https://www.dbpedia.org/
[13] https://www.kaggle.com/krrai77/chemicals-in-cosmetics

instances of the class *ProductIngredient* after which we can retrieve useful information about them from DBPedia that can then be presented to the user.

All DBPedia information will be acquired through multiple SPARQL queries using the DBPedia SPARQL endpoint in GraphDB. All necessary additional information about brands, individual products, types of products, or the ingredients used in products can be retrieved by creating SPARQL queries based on the instances that were added before. This additional information will be added to the ontology, in the form of triples, using an INSERT query.

11. Meaningful inferences in Protégé

the ontology should produce meaningful **inferences** that are essential for the application. This should be evidenced by **screenshots** of, e.g., Protégé reasoning results. Describe the inferences (100-500 words) (**NB**: For the final report: inferences should be on the external data)

This will be added later and (most likely) asked about in a QnA session.

12. Complex SPARQL queries

For our skincare application, we will design multiple SPARQL queries that can retrieve results from the ontology and integrated data. These have not been finalised yet but for now, the following form a representation of the ideas for SPARQL queries (external ontologies will be used in the final versions).

SPARQL query 1:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

PREFIX sc: <http://www.semanticweb.org/skincare/>

PREFIX dbp: <http://dbpedia.org/property/>

PREFIX owl: <http://www.w3.org/2002/07/owl#>


SELECT DISTINCT ?brand ?name  ?comment ?products WHERE{

        ?brand a sc:ProductBrand;

          sc:hasName ?name

   BIND(CONCAT(SUBSTR(?name, 1, 1), lcase(SUBSTR(?name, 2))) as ?brandname)

   BIND(STRLANG(?brandname,"en") as ?tag)

   SERVICE <https://dbpedia.org/sparql> {

        ?dbpbrand rdfs:label ?tag.

        ?dbpbrand rdfs:comment ?comment.

        ?dbpbrand dbp:products ?products.

        }
}
```

This SPARQL query retrieves information from DBPedia about different brands that were added as instances from the Kaggle csv file.

SPARQL query 2:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sc: <http://www.semanticweb.org/skincare/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>


SELECT DISTINCT * WHERE{
      ?product a sc:ProductIngredient;
     sc:hasName ?label.


  SERVICE <https://dbpedia.org/sparql> {
      ?dbptype rdfs:label ?label.
    ?dbptype rdfs:comment ?comment.
      }
}
```

QUERY to obtain more info about the ingredients that can be found in products.

SPARQL query 3:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sc: <http://www.semanticweb.org/skincare/>

PREFIX dbp: <http://dbpedia.org/property/>

PREFIX owl: <http://www.w3.org/2002/07/owl#>


SELECT DISTINCT * WHERE{

        ?productName sc:hasIngredient sc:ProductIngredient;

        sc:isProductType sc:Moisturise;

        sc:isForSkinType sc:Combination;

        sc:isOfBrand sc:ProductBrand

        sc:hasName ?name


}
```

Simple representation of a query the recommendation engine of our application would use. The properties on which it searched would be based on the user's input in the questionnaire.

13. Inferences of the SPARQL queries

a description and evidence that running the SPARQL queries against the ontology and data produces **inferences** (**screenshots** reasoning on/off). Discuss the inferences. (200-300 words) []

This will be added later and (most likely) asked about in a Q&A session.

14. Have a working prototype of the Jupyter notebook, which uses Pandas dataframes (nothing to hand in)

Appendix

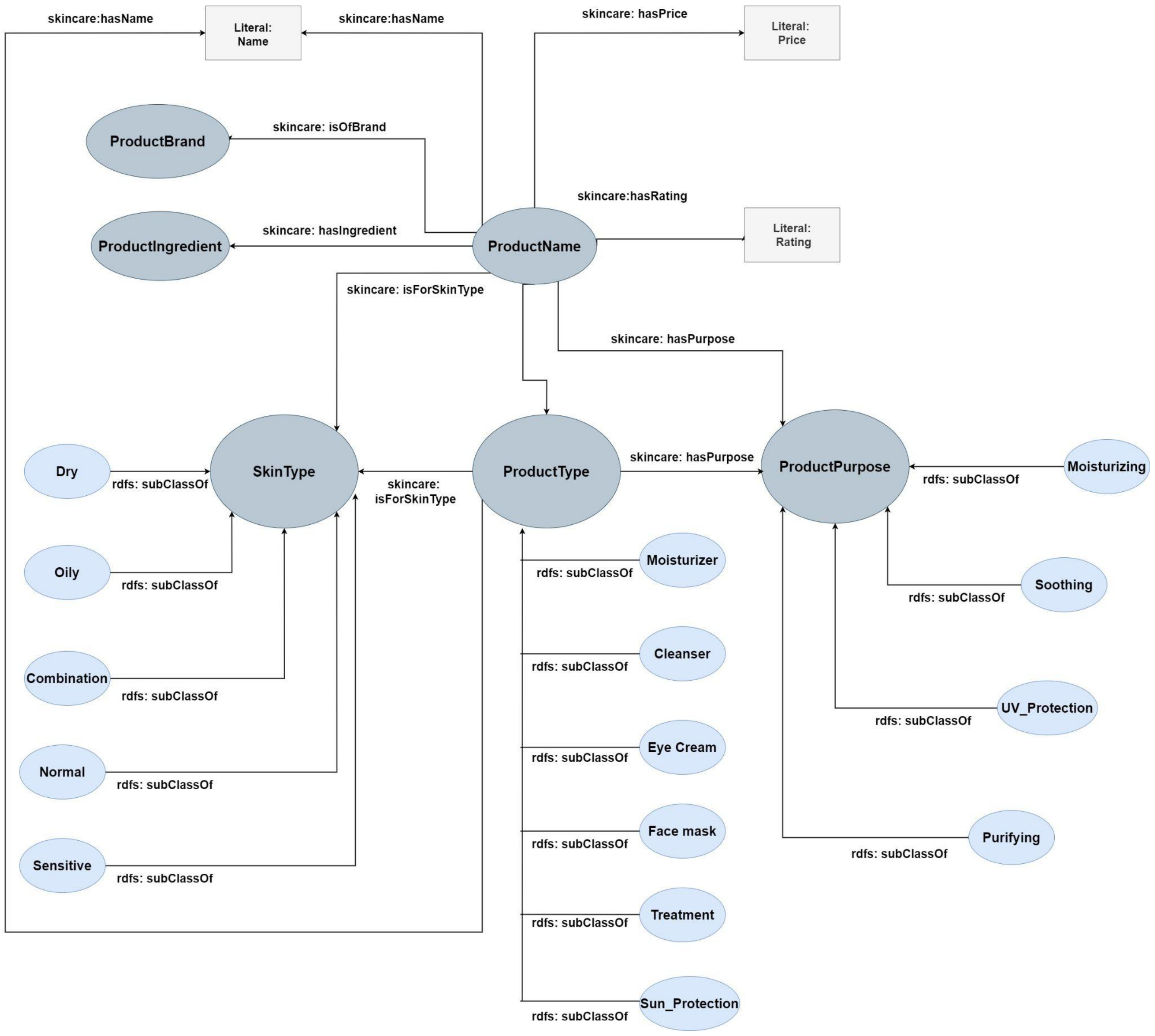


Figure 1. The graph of the created skincare ontology