Fragrance Matcher

The Fragrance Matcher application aims to provide users with the ability to create their own accounts, build perfumes using fragrances from a dataset, and test the compatibility between two fragrances. Additionally, it offers a feature to find the ten most compatible fragrances given another fragrance. The project is divided into two main parts: *Fragrance Recommendation* and *Fragrance Project*.

Fragrance Recommendation

This component is built using Python 3.7 and focuses on processing the dataset, creating and training an AI model using Gensim's Word2Vec algorithm. The steps involved in this part are as follows:

a) Preprocessing the dataset: The fragrance dataset is obtained from Kaggle and undergoes preprocessing to clean the data, remove any inconsistencies, and ensure compatibility with the Word2Vec algorithm.

b) Creating and training the AI model: Gensim's Word2Vec algorithm is used to create an AI model based on the preprocessed fragrance dataset. This model learns the relationships between fragrances and is trained using the clean data.

c) Integration with web app: The trained AI model from the Fragrance Recommendation component is integrated into the Fragrance Project web app for fragrance recommendation and compatibility testing.

Word2Vec is a model for creating word embeddings from a text-based input, making the natural language readable by the computer, and can be trained with one of the following two versions: Skip-grams or Continuous-bag-of-words. It is a modern way of representing words, where each word is represented by a vector using a flat neural network capturing syntactic as well as semantic consistencies, according to Radim Řehůřek on Gensim: Word2Vec Model. Several operations can then be performed on these vectors to search for word similarities. Similar words are placed next to each other and words with different meanings are further apart. The model belongs to unsupervised learning and trains a neural network with a hidden layer. It uses a one-hot encoding of words that goes through a projection layer into the hidden layer and the output layer uses softmax regression, which gives the probability of a target word to be in the context of an input word. Then the projection weights are later interpreted as word embeddings.

For creating Word2Vec word embeddings, the Fragrance Recommendation app used a data set from Kaggle, which consists of perfumes that contain fragrance notes and are each assigned to a brand. In order to create the fragrance similarities, each word from the text dataset is read and then its neighboring words (context) are selected within a certain distance. The words represent the fragrances and the data pairs represent the perfumes. In the image below it can be seen some information about the dataset:

```
Size fragrances dataset (with repetition):      394046
Unique fragrances dataset:          4619
Total # of perfumes                 49341
Total # of brands               8176
Mean deviation of the fragrances frequencies: 85.30980731760121
Standard deviation of the fragrances frequencies:  633.4571654365387
```

An example of how the model works is shown in the next images:

```
Top 15 fragrances:
     ('Musk', 16733)
     ('Sandalwood', 12962)
     ('Jasmine', 12727)
     ('Bergamot', 12647)
     ('Vanilla', 12469)
     ('Amber', 12380)
     ('Patchouli', 11574)
     ('Rose', 10212)
     ('Vetiver', 6440)
     ('Lemon', 6113)
     ('Cedarwood', 5645)
     ('Mandarin orange', 5440)
     ('Lavender', 4826)
     ('Cedar', 4521)
     ('Tonka bean', 4352)
```

```
Tail 15 fragrances:
     ('Indian lemon balm', 1)
     ('Angelica root CO2', 1)
     ('Chocolate biscuit', 1)
     ('Caramelized pineapple', 1)
     ('Indonesian Rosa alba', 1)
     ('Texan leather', 1)
     ('Madagascan ambergris', 1)
     ('Chlorine', 1)
     ('Fiji sandalwood', 1)
     ('Apricot jelly', 1)
     ('Iris milk', 1)
     ('Rum cream', 1)
     ('French currant', 1)
     ('Colombian orange', 1)
     ('Spanish resin', 1)
```

```
Most compatible with rose:
         -iris: 0.872
         -orange blossom: 0.768
         -heliotrope: 0.748
         -lily: 0.748
         -lilac: 0.743
         -plum: 0.742
         -violet: 0.738
         -peach: 0.737
         -sandalwood: 0.727
         -carnation: 0.715
compatibility("red fruits", "water jasmine") =  0.5511524
```

Since Word2Vec is an unsupervised algorithm, the validation methods are not as straightforward as for supervised learning, so the models are most often tested via external validation. Thus it can be tested the performance of training the Word2Vec model. One can see in the table how the training loss, the type of model and the input data affect the training time.

	input_data	compute_loss	sg	train_time_mean	train_time_std
3	25kB	False	1	0.037925	0.004870
0	25kB	True	0	0.061840	0.011606
2	25kB	True	1	0.071983	0.001268
1	25kB	False	0	0.082642	0.014532
4	1MB	True	0	0.591481	0.015548
5	1MB	False	0	0.592811	0.002560
7	1MB	False	1	1.434857	0.067394
6	1MB	True	1	1.483313	0.078832
8	10MB	True	0	5.857182	0.049698
9	10MB	False	0	6.092750	0.354071
10	10MB	True	1	16.000422	0.236671
11	10MB	False	1	16.313952	0.390362
12	50MB	True	0	32.138134	0.841327
13	50MB	False	0	33.290406	0.359520
16	100MB	True	0	94.011636	1.204195
17	100MB	False	0	96.969197	0.994261
14	50MB	True	1	108.802960	6.028130
15	50MB	False	1	130.411238	5.818611
19	100MB	False	1	245.072844	41.719803
18	100MB	True	1	325.314851	20.048555

Fragrance Project

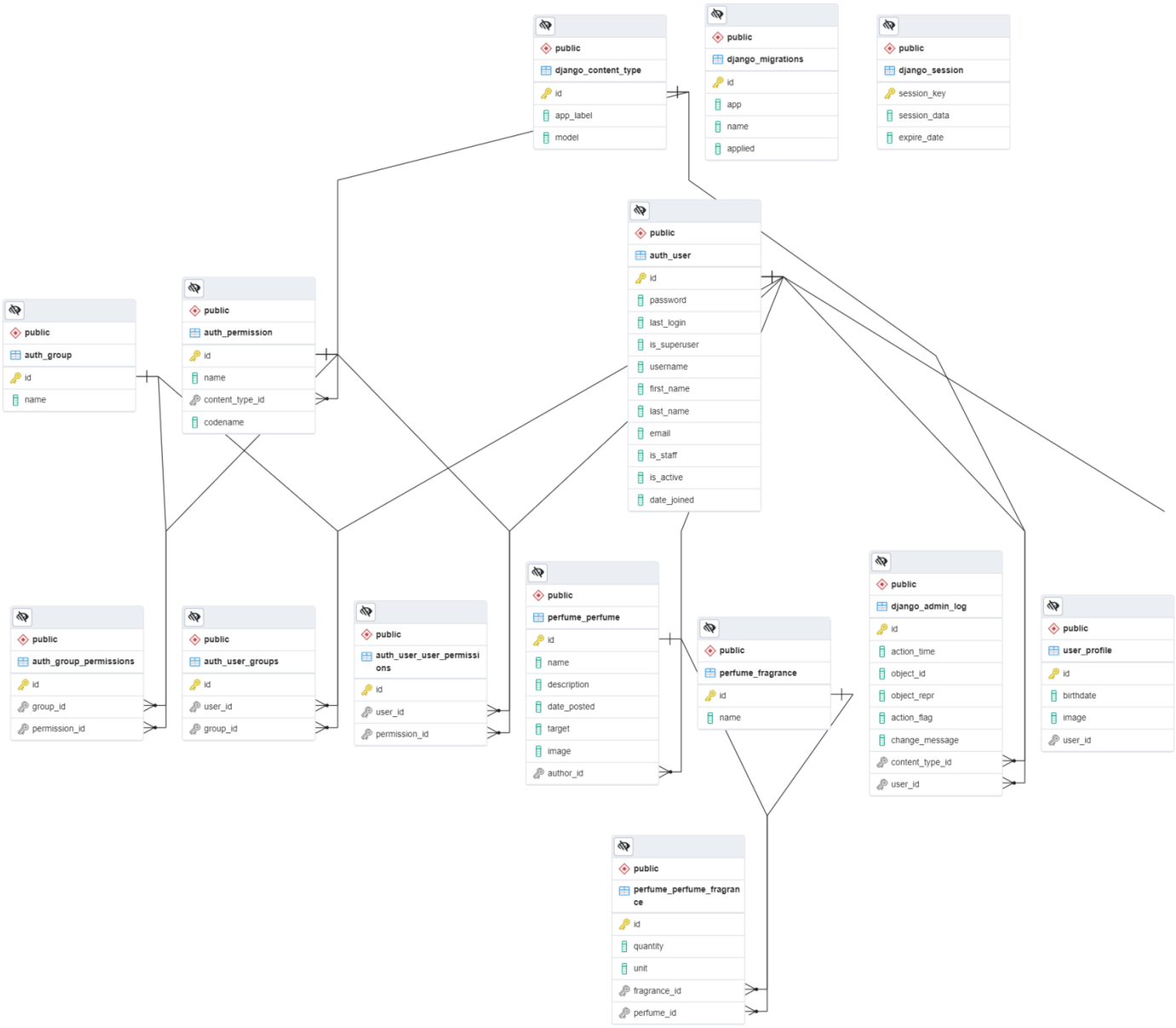
The Fragrance Project web app is built using Django framework and comprises two major components: user and fragrance. Django provides a robust and scalable environment for building the Fragrance Project web app, enabling efficient user management, perfume creation, and fragrance compatibility testing.

a) User component: The user component handles the logic for creating and updating user accounts, as well as login and logout functionalities. It ensures the security and privacy of user accounts.

b) Fragrance component: The fragrance component is responsible for the CRUD (Create, Read, Update, Delete) operations related to perfumes. Users can create perfumes and select fragrances from the dataset. The component also facilitates fragrance compatibility testing, allowing users to check the compatibility between two fragrances. Furthermore, it provides a feature to find the ten most compatible fragrances given a particular fragrance.

Django is a web framework, which is a Python framework that implements an architecture called MVT (Model View Template) to better organize the main components of the application, their structure and the way they communicate with each other. The Model works like an image of the data which is stored in the database and is responsible for maintaining the data, facilitating operations with it. Views handle all of the business logic behind the web application and they are the connection between models and templates, whilst receiving the user's request and providing the right data for the templates to answer it. Templates are accountable for the way in which the data is presented to the user that visits the website.

It is also worth noting, that when no user is logged in, only the "about" page is accessible to the user. Additionally, only the author of a perfume can update or delete that perfume, ensuring data integrity and user-specific operations. Moreover, for the data retention and integrity is has been used PostgreSQL and the table schema can be seen below:



As result, the Fragrance Matcher web app demonstrates successful implementation of the desired functionalities. Users can create accounts, build personalized perfumes using available fragrances, and test the compatibility between fragrances. The fragrance recommendation feature also suggests ten most compatible fragrances based on user input. The integration of the Fragrance Recommendation component ensures accurate recommendations, leveraging the trained AI model. User access restrictions ensure data privacy and integrity.

In conclusion the Fragrance Matcher app provides a valuable platform for perfume enthusiasts. Users can explore and experiment with different fragrances, creating customized perfumes and verifying compatibility. The successful integration of Python's Fragrance Recommendation with Django's Fragrance Project ensures accurate recommendation results and a user-friendly interface. Future enhancements may include expanding the dataset for broader fragrance options and exploring additional AI algorithms for improved fragrance recommendations.