Cravr API Docs

Ben Gottfried, Elias Talcott, and Justin Hyndman

Last Revised: April 7, 2021

Contents

1 Backend Function Index

1.1 Authentication

Logs in existing users and creates new users. Located in `authentication_utils.py`.

1.1.1 authenticate_user

Description

Log in an existing user. Fails if username/password do not match a database entry or if database connection cannot be established.

Arguments

username	String containing username. Must be a valid email address.
password	String containing password.

Return Value

True if authentication is successful. False otherwise.

1.1.2 register_user

Description

Create a new user in the database and log in. Fails if username is already associated with an account or if database connection cannot be established.

Arguments

username	String containing username. Must be a valid email address.
password	String containing password.

Return Value

True if registration is successful. False otherwise.

1.2 Database

Executes reads and writes to the database. Located in `database_utils.py`.

1.2.1 setup

Description

Class method to create a MySQL instance at the beginning of app execution. Uses a default configuration from a DatabaseConfig object and allows custom configuration parameters. Once configured, create one connection and add it to the connection pool.

Arguments

app	Flask app instance.	
**kwargs	Custom configuration parameters.	
	socket	Tuple containing hostname and port of the MySQL database. Defaults are "MYSQL_DATABASE_HOST" environment variable and port 3306.
	credentials	Tuple containing user and password to access database.
	database	Name of database to use. Default is "cravr".
	charset	Character set used by database. Default is "utf8".

Return Value

None.

1.2.2 execute_query

Description

Gets a connection from the connection pool, executes a query on the database, and gives the connection back. Fails if database connection cannot be established.

Arguments

query	String containing MySQL query to be executed.

Return value

String containing the first result of the query. -1 if database connection failed.

1.3 Recommender

Suggests restaurants based on user and restaurant data. Located in `recommender.py`

1.3.1 get_restaurant

Description

Arguments

Return value

1.3.2 cache_restaurant

Description

Arguments

Return value

1.4 Routing

Interacts directly with the frontend for authentication, restaurant recommendations, and user feedback. Provides restaurant/review data and routing information. Located in `app.py`.

1.4.1 login

Description

Uses authentication and database functionality to check if user's credentials are valid. Fails if username not found in database, hashed password does not match, or database connection cannot be established.

Return value

{"result": "/"} if authentication successful. {"result": "/Login"} otherwise.

1.4.2 register

Description

Uses authentication and database functionality to register a new user. Fails if username already exists in database or database connection cannot be established.

Return value

{"result": "/Login"} if registration successful. {"result": "/Register"} otherwise.

1.4.3 get_restaurant

Description

Parses a user's search parameters and gives a restaurant recommendation.

Return value

{"result": RESTAURANT_OBJECT} where RESTAURANT_OBJECT is the return value of the Recommender class's get_restaurant method.

1.4.4 rate_suggestion

Description

Take a restaurant ID along with a user's rating of "Yummy", "Maybe later", or "Yuck" to update their review list and train the recommender.

Return value

None.

1.4.5 get_reviews

Description

Fetch a list of restaurants that the user needs to review.

Return value

{”result”: REVIEW_LIST} where REVIEW_LIST is the return value of the User class’s get_reviews method.

1.4.6 submit_review

Description

Remove a restaurant from a user’s review list and use review data to train their recommender.

Return value

None.

1.5 User Utilities

Manage users’ restaurant caches, review lists, and recommender training data. Located in `user.py` and `user_data_utils.py`

1.5.1 User

Description

Class that maintains permanent user data such as the restaurants they need to review and their recommendation model.

Fields

str name	The user’s username or email address.
bool is_dirty	Boolean for whether or not the User object has been modified since it was last written to the database.
list reviews	List of Yelp business IDs (strings) that the user has clicked ”Yummy” on and must review.
Model model	Model object that serves as the user’s individual recommendation model.

Methods

add_review **Description**
Append a Yelp ID to the User’s reviews list.
Return value
None.

get_reviews **Description**
Get list of restaurants that need to be reviewed by the User.
Return value
List of dictionaries for each ID in the reviews list containing the following:

restaurant	Yelp object returned by looking up the ID’s business details.

review	Dictionary containing the default review settings that are changed in the frontend.

submit_review **Description**

The User reviewed a restaurant in the frontend and it must be removed from the User's reviews list passed to the model for training

Return value

None.

handle_review **Description**

Train the User's model based on the review for this restaurant.

Return value

None.

disliked **Description**

Adjust model weights to discourage similar restaurants from being suggested when the "Yuck" button is clicked.

Return value

None.

1.5.2 UserList

Description

Class that caches a dictionary in memory of unique usernames mapped to User objects. This is a subset of all of the user data stored in the database. When the UserList is deleted (e.g. when the app goes down), it writes all modified user data to the database in its destructor. Its default methods have been overridden so that it behaves similar to a standard dictionary in Python.

Fields

dictionary users	Mapping of unique usernames to their User objects.
bool is_prod	Boolean that when false, disables the destructor in order to pass unit tests.

1.5.3 read_user_data

Description

Read the user's data from the database

Arguments

str username	String for the User's username

Return value

A dictionary containing the user's reviews list and model if the user exists in the database, otherwise None.

1.5.4 write_user_data

Description

Modify the user's data in the database

Arguments

str username	String for the User's username
User data	User object containing data that will be replace the value at the key 'username' in the database if it exists, otherwise a new entry in the database will be created.

Return value

True if write successful, False otherwise

1.6 Yelp Fusion API

Gets restaurant data from Yelp Fusion API. Located in `yelp_api_utils.py`

1.6.1 business_search

Description

Arguments

Return value

1.6.2 business_details

Description

Arguments

Return value

1.6.3 business_reviews

Description

Arguments

Return value

2 Code Samples

2.1