Cocktails CBR Recommender system

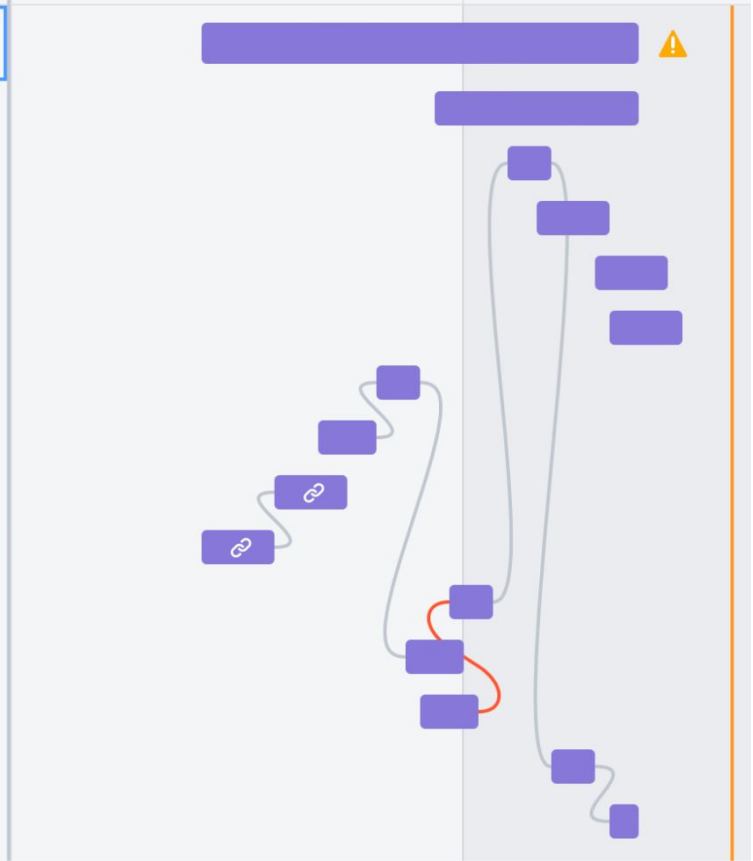
Bejarano Edison
Del Rey Santiago
Reyes Grecia
Zurita Yazmina

TABLE OF CONTENTS

ES SR YZ GR

Status category

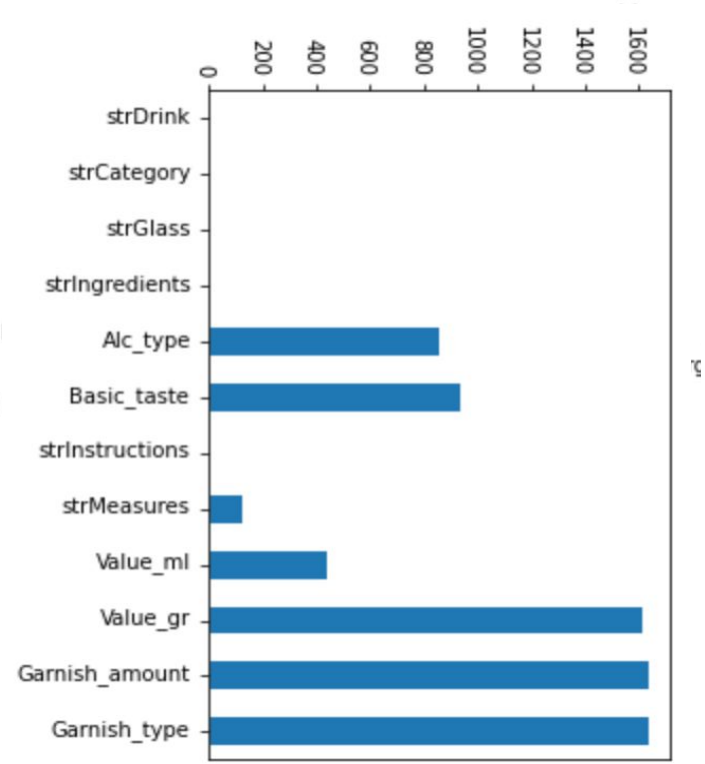
Epic	
SP-3 Implement the CBR system	
~~SP-32~~ Create UI	DONE SR
~~SP-24~~ [EVAL]	DONE ES
~~SP-36~~ Testing if the evaluation is working well	DONE ES
~~SP-37~~ Modify the learning class with the conditional of the user's sc...	DONE ES
~~SP-42~~ Add elements to cocktails	DONE SR
~~SP-9~~ Create case library	DONE SR
~~SP-8~~ Choose case library structure	DONE SR
~~SP-12~~ Choose case structure / prepare data	DONE SR
~~SP-35~~ Choose dataset and tell the teacher	DONE
~~SP-23~~ [ADAPT]	DONE YZ
~~SP-22~~ [RETRIEVE] Search the case library	DONE GR
~~SP-20~~ [RETRIEVE] Implement similarity measure	DONE GR
~~SP-25~~ [LEARN] Evaluate if the case is worth learning	DONE ES
~~SP-26~~ [LEARN] Add case to the case library	DONE ES

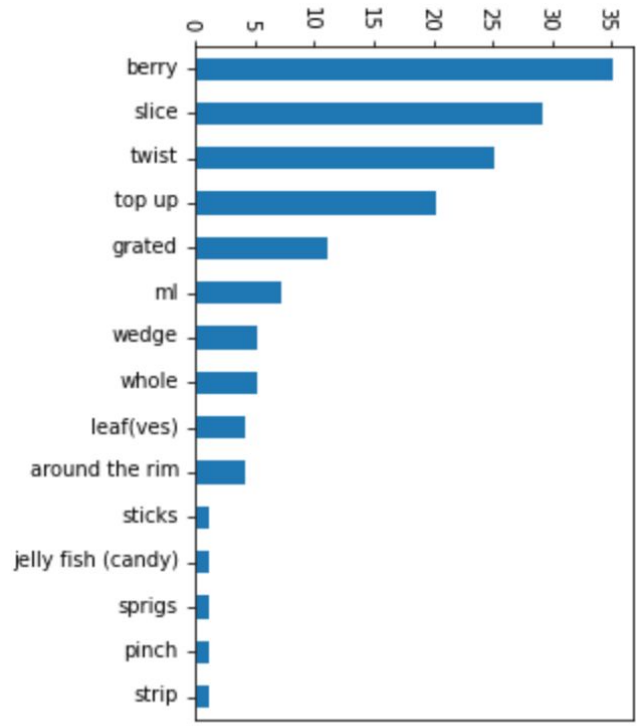


Data domain

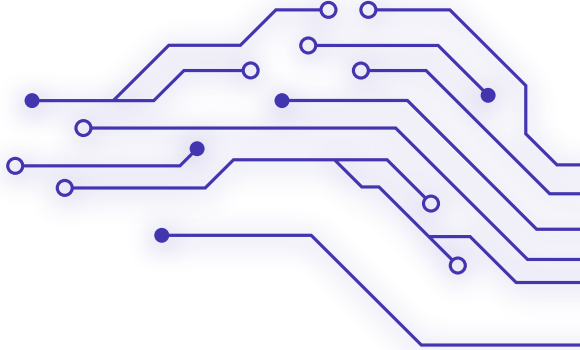
473 unique cocktails with each ingredient labeled and measure defined.

Attributes	No
Ingredients	303
Categories	9
Alcohol Type	25
Basic Taste	9
Glass Type	35
Garnish type	15





Requirements Analysis



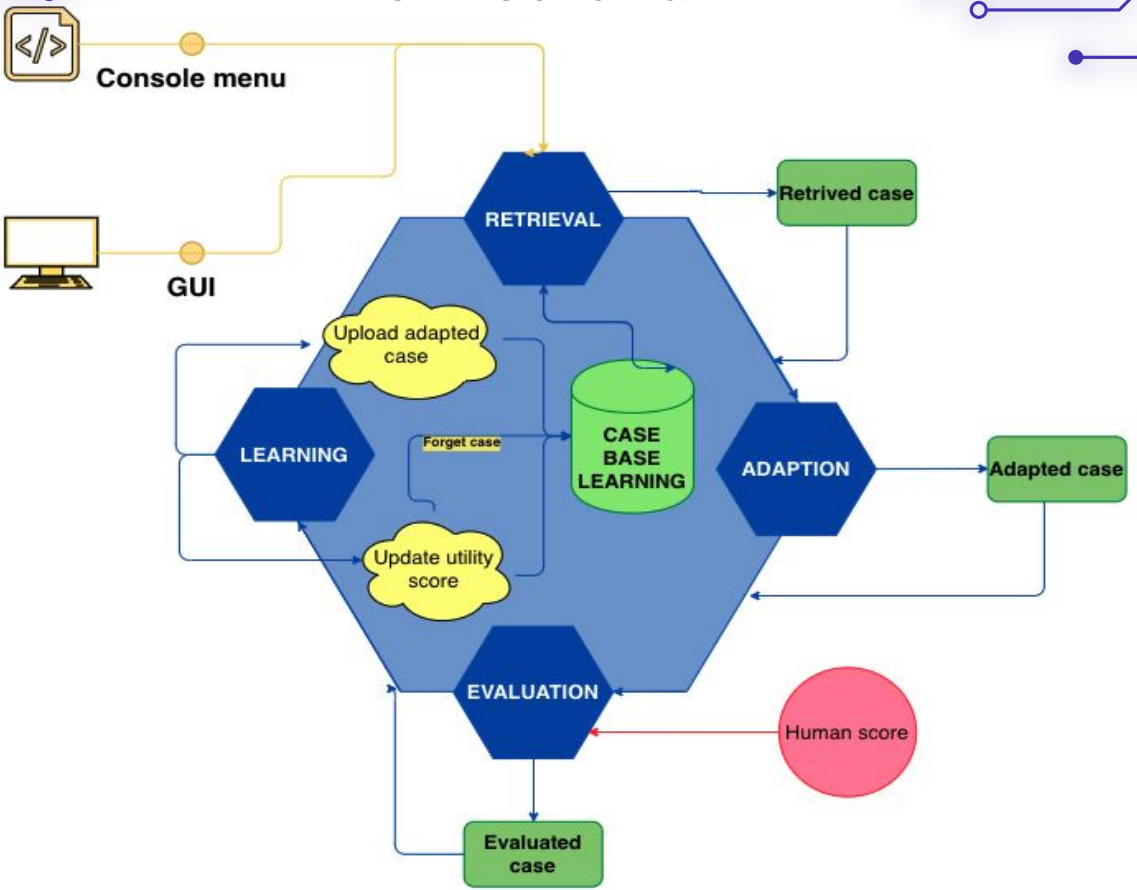
User requirements

- Select drink type.
- Select glass type.
- Include alcohol type.
- Include basic taste.
- Include ingredients.
- Exclude ingredients.
- Evaluate recipe.

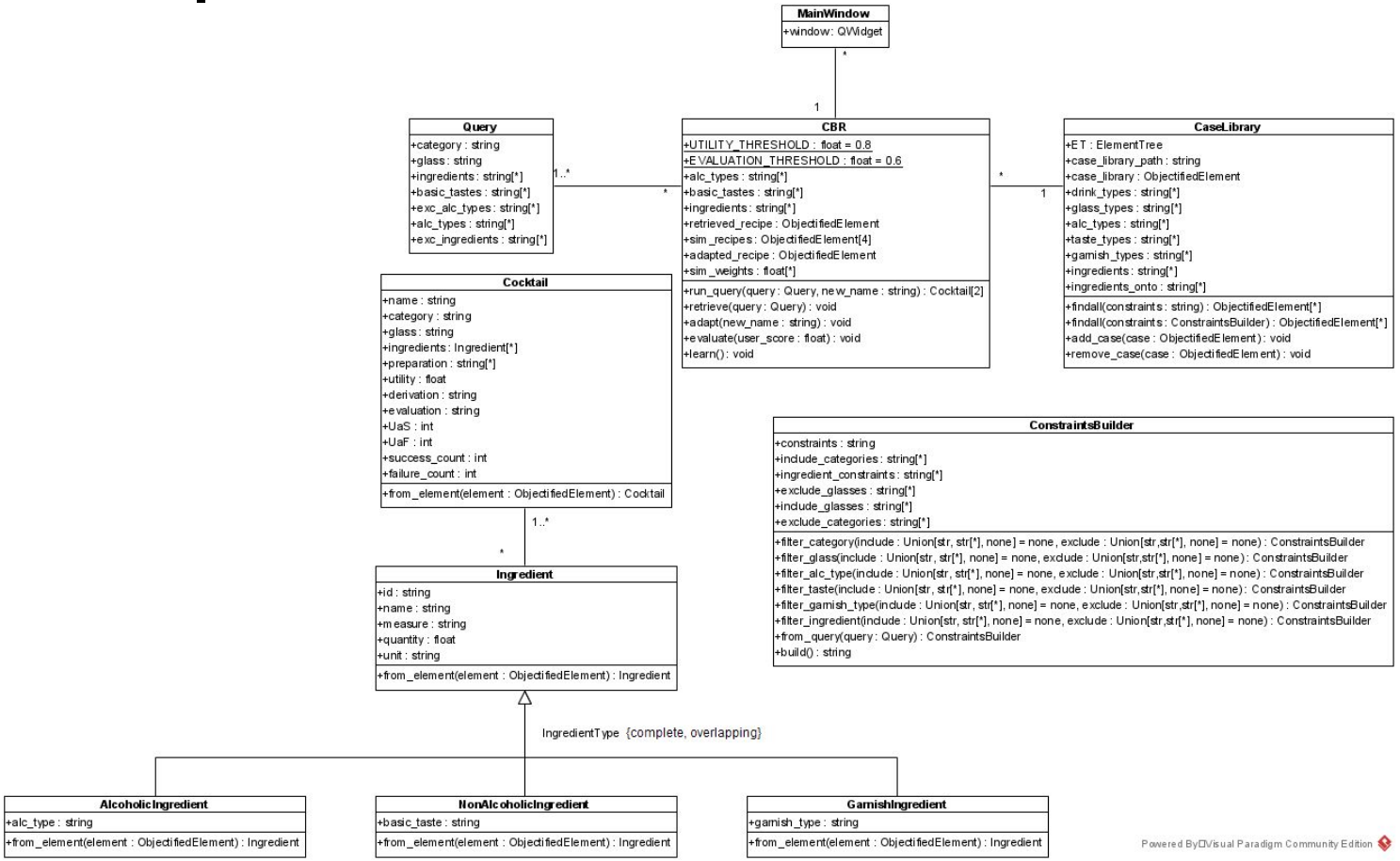
Technical requirements

- Time to process query < 3 seconds.
- Size of the case library < 1 GB.
- Number of queries before failure > 10.
- User friendly.

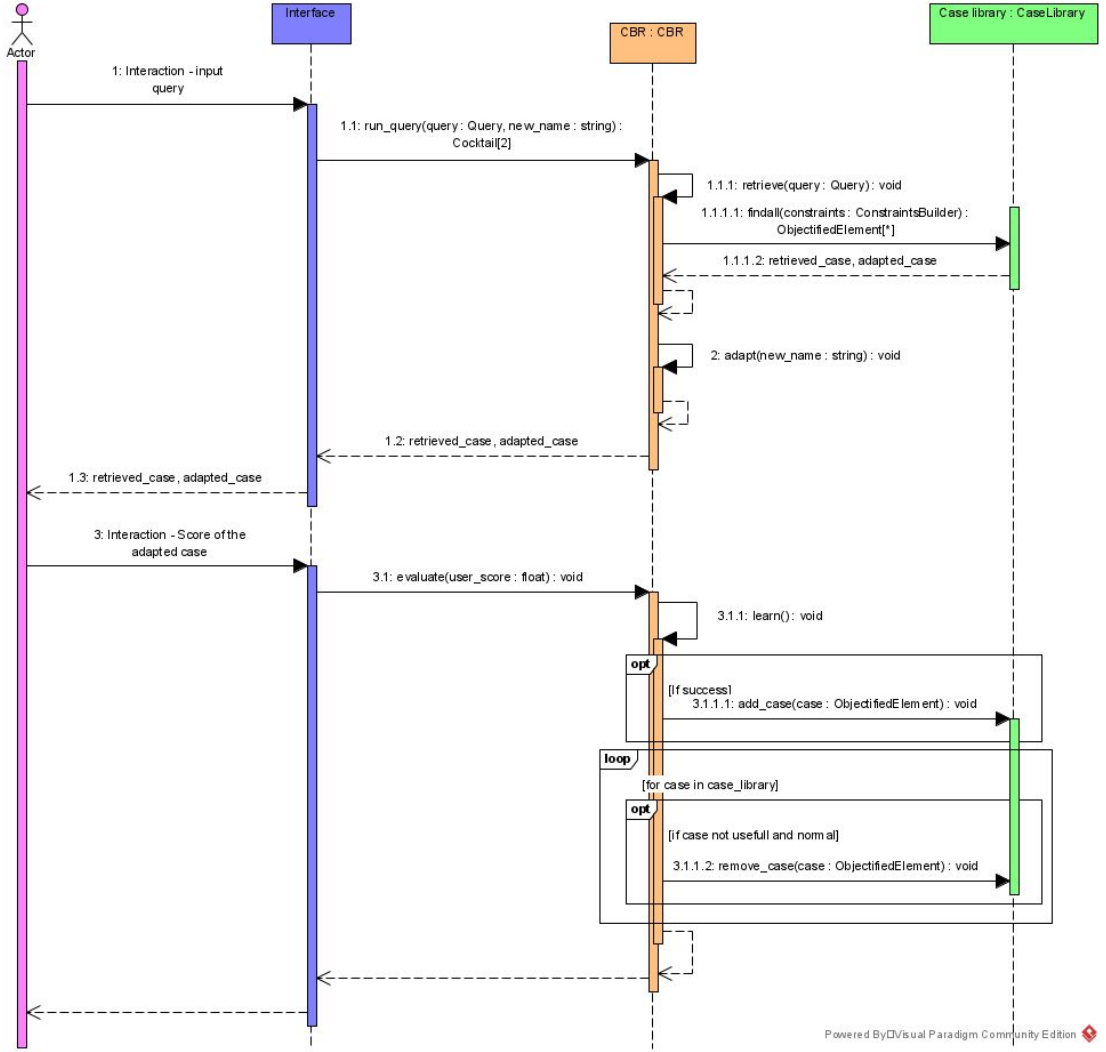
CBR Schema


Console menu
GUI
RETRIEVAL
Retrived case
Upload adapted case
LEARNING
Forget case
CASE BASE LEARNING
ADAPTION
Adapted case
Update utility score
EVALUATION
Human score
Evaluated case

System components



Interaction sequence

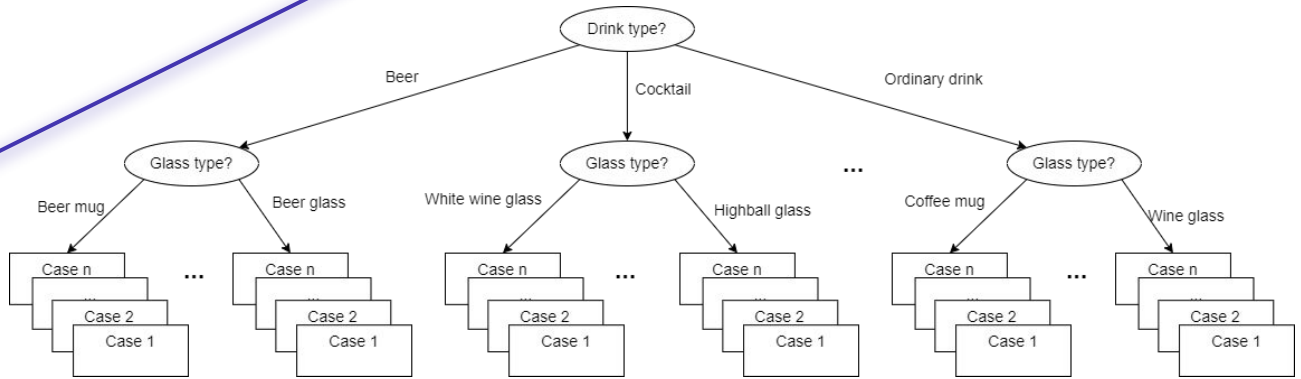


Case representation and Library structure

Case representation

```
<cocktail>
  <name>Snake Bite (UK)</name>
  <category>beer</category>
  <glass>pint glass</glass>
  <ingredients>
    <ingredient id="ingr0" alc_type="beer" basic_taste="" measure="1/2 pint" quantity="236.5" unit="ml" garnish_type="">lager</ingredient>
    <ingredient id="ingr1" alc_type="wine" basic_taste="" measure="1/2 pint dry" quantity="60.0" unit="ml" garnish_type="">red wine</ingredient>
  </ingredients>
  <preparation>
    <step>pour ingredients into a pint glass</step>
    <step>drink</step>
    <step>fall over</step>
  </preparation>
  <utility>1.0</utility>
  <derivation>original</derivation>
  <evaluation>success</evaluation>
  <UaS>0</UaS>
  <UaF>0</UaF>
  <success_count>0</success_count>
  <failure_count>0</failure_count>
</cocktail>
```

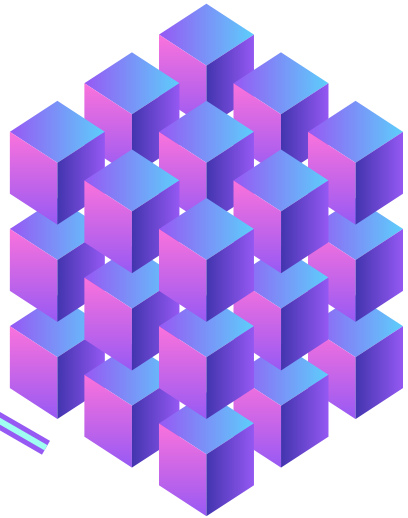
Library structure



CBR System: Retrieval

Searching

- Pre-select the cases that **match with the constraints** given by the user in the new case.
- The constraints: **preferences** and **exclusions** that the user enters in the GUI or in the CL
- We will have **at least 5 cases retrieved** from the case library

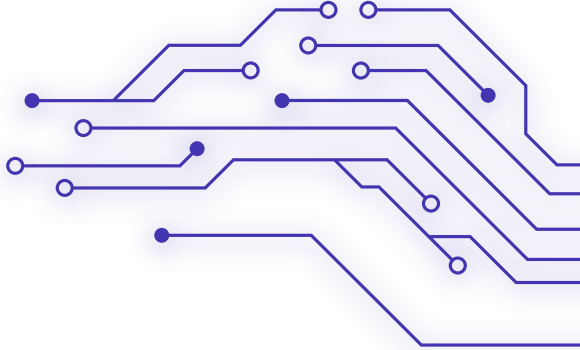


Selecting

- Select the **most relevant case among the subset retrieved** in the previous step.
- To calculate the similarity we will **define weights $w_k \in [0, 1]$**.

(+) Constraints included: 1, 0.85 y 0.5
(-) Constraints excluded: -1 y -0.5

CBR System: Adaptation



Main methods applied on the adapted recipe:

`delete_ingredient:`

1. Remove it from the ingredient list.
2. Remove it from the steps:
   - If there is more than one ingredient in the step → placeholder [IGNORE].
   - If it is the only ingredient in the step, delete the step.

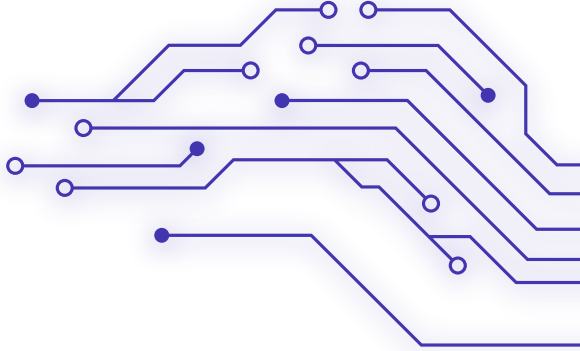
`replace_ingredient:`

1. Replace text of one ingredient by the text of another.

`include_ingredient:`

1. Add it to the ingredients list with `measure = M` and `id = ingrN`.
2. Insert step in the second position:
   - If `M` → "add `ingrN`".
   - Otherwise, "add `ingrN` to taste".

CBR System: Adaptation



Input: query of the user, adapted recipe, similar recipes.

1. Exclude ingredients:
   - If it is not an alcohol → find ingredient with the same `basic_taste` and call `replace_ingredient` (if failure → call `delete_ingredient`).
   - If it is an alcohol → call `delete_ingredient`.

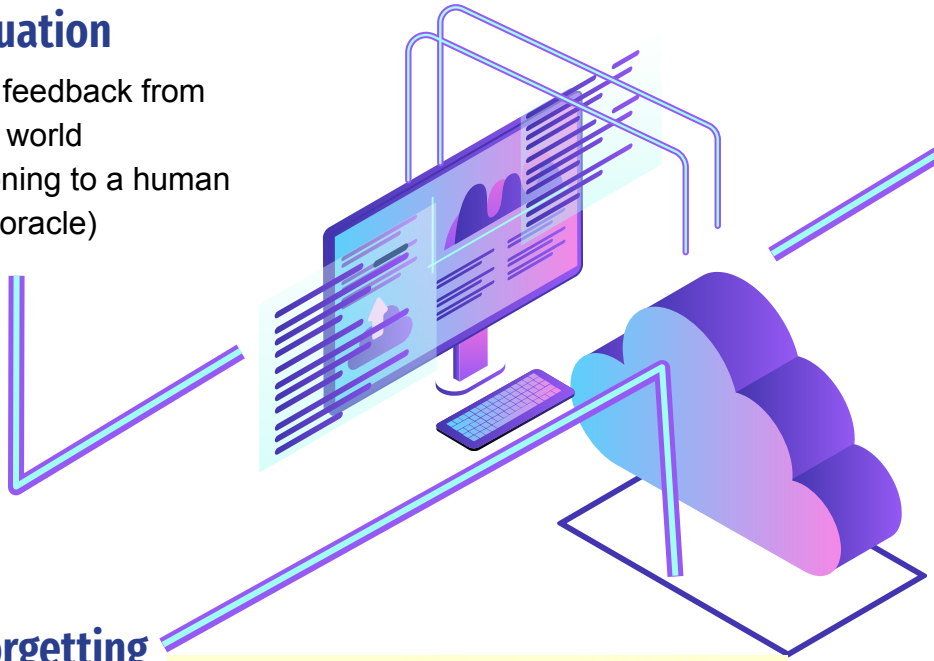
2. Include ingredients:
   - Search for the measure and call `include_ingredient`.

3. Adapt alcohol types and basic tastes:
   - Search for an ingredient with the `basic_taste` or `alc_type` desired and call `include_ingredient`.

Evaluation - Learning - Forgetting

Evaluation

- Getting feedback from the real world
- Questioning to a human expert (oracle)



Learning from experience

Learning from success

- Storing a new successful case
- Update the utility measure of all retrieved cases

Learning from failure

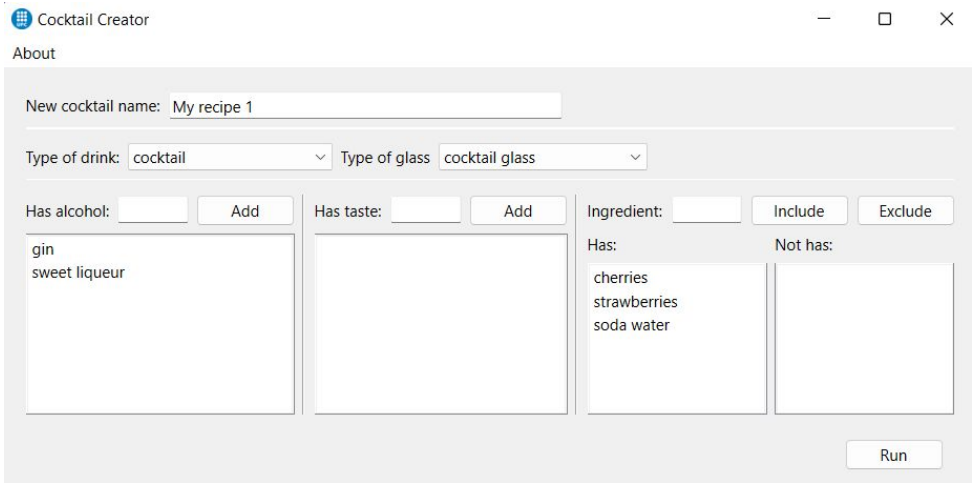
- Storing a new failed case
- Update the utility measure of all retrieved cases

Forgetting

- Utility measure for forgetting
- Normalized utility

$$UM(C) = \frac{\frac{\#UaS}{\#S} - \frac{\#UaF}{\#F} + 1}{2}$$

Evaluation and Results: Manual



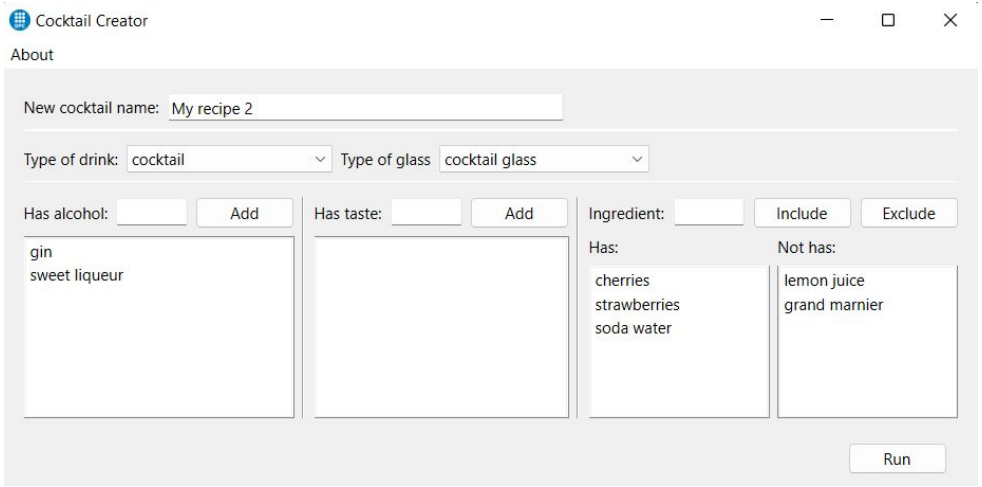
Retrieved case:

My recipe 1
Type of drink: cocktail
Glass: cocktail glass
Ingredients:
1/8 shot of grenadine, 1 3/4 shot of gin, 1/4 shot of lemon juice, 1 shot of grand marnier some cherries, fill to top with of soda water, 1 part of galliano
Preparation:
0. pour all ingredients into a cocktail shaker, mix and serve over ice into a chilled glass
1. add 1 part of galliano
2. add fill to top with of soda water
3. add some cherries to taste

Adapted case:

Type of drink: cocktail
Glass: cocktail glass
Ingredients:
1/8 shot of grenadine, 1 3/4 shot of gin, 1/4 shot of lemon juice, 1 shot of grand marnier some cherries, fill to top with of soda water, 1 part of galliano, some strawberries
Preparation:
0. pour all ingredients into a cocktail shaker, mix and serve over ice into a chilled glass
1. add some strawberries to taste
2. add 1 part of galliano
3. add fill to top with of soda water
4. add some cherries to taste

Evaluation and Results: Manual



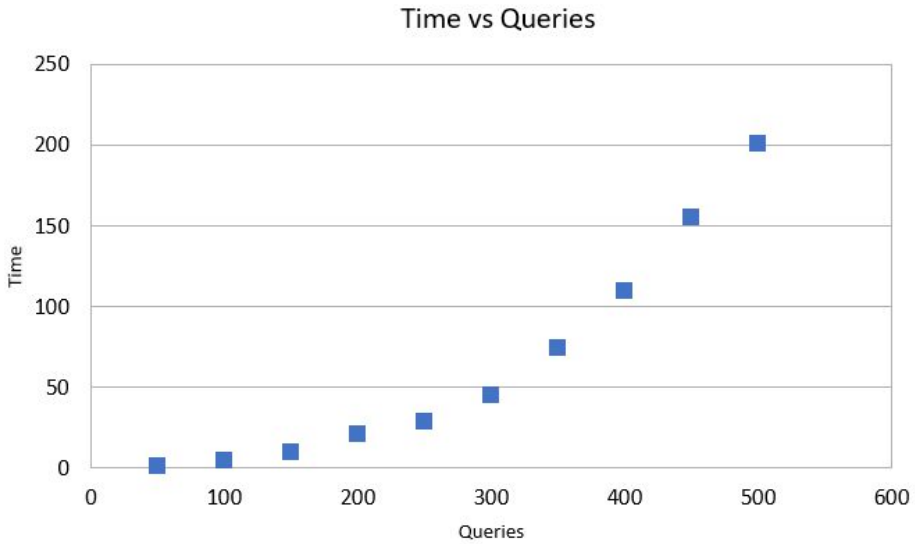
Retrieved case:

Addington
Type of drink: cocktail
Glass: cocktail glass
Ingredients:
1 shot of dry vermouth, top up with of soda water, 2 shots of sweet vermouth
Preparation:
0. mix both the 1 shot of dry vermouth's in a shaker and strain into a cold glass
1. top up with a squirt of top up with of soda water

Adapted case:

My recipe 2
Type of drink: cocktail
Glass: cocktail glass
Ingredients:
1 shot of dry vermouth, top up with of soda water, 2 shots of sweet vermouth, some cherries some strawberries, 1 3/4 shot of gin, 1 part of galliano
Preparation:
0. mix both the 1 shot of dry vermouth's in a shaker and strain into a cold glass
1. add 1 part of galliano
2. add 1 3/4 shot of gin
3. add some strawberries to taste
4. add some cherries to taste

Evaluation and Results: Automatic



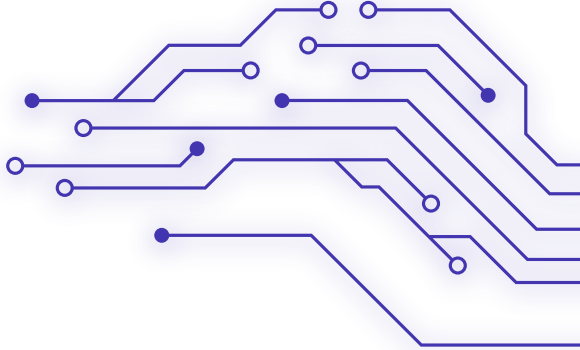
(*)11 samples - average

- test.py script was developed to perform the necessary tests.
- Generates a number of queries randomly.
- Save: the retrieved case, the adapted recipe generated, the total number of queries and processing time.

➔ 500 queries → 200 sec / 0.4 sec (system should respond less than 3 sec)

➔ Case library 522 kB → executing the tests 3.22 MB

Conclusions and future work



- We have implemented a CBR system able to generate cocktail recipes based on user preferences.
- The generated recipes satisfy the user constraints.
- The system complies with the requirements specified.
- Although we have achieved our goal, the solution could be further improved:
  - Apply NLP techniques to better clean the source dataset.
  - Explore other methods to learn and forget.
  - Introduce a measure on how well the user constraints are being met.