**Hungerger**

**Use-Case UC5: Edit Recipe**

| **REVISIONS** | | | |
| --- | --- | --- | --- |
| **Rev. No** | **Description** | **Date** | **Person** |
| 0.1 | Creating the document | 23.11.2023 | Aslı |
| 0.2 | Typos | 25.11.2023 | Aslı |

**Scope:** Hungerger Application

**Level:** User Goal

**Primary Actor:** User

**Stakeholders and Interests:**

* User: Wants accurate, fast entry and up-to-date prices as possible.
* Marketplace: Represents the online external platform from which the application collects

the prices of various ingredients. Wants to be requested API call (See Open Issues).

**Precondition**

* The User shall have an account on the system and be logged in.
* The User shall have at least one recipe in their account.
* A collaboration must be made with an online marketplace (See Open Issues) to collect up-to-date prices of the ingredients for accurate cost calculation.
* The database system must be operational.
* There is an active network connection.

**Success Guarantee (or Postconditions):**

* The recipe is updated, or the process is canceled.
* The ingredients' prices and the recipe's total price are correctly calculated.
* The database is updated with the changes.
* The changes are reflected in the User interface of the application.

**Main Success Scenario (or Basic Flow):**

1. The use case begins when the User selects a recipe and starts to edit it.

2. The system displays a page where the User can edit recipe information (See Supplementary Requirement WC-1 for recipe information).

3. The User edits recipe information (See Supplementary Requirement WC-2 for editing types).

4. The system validates the information entered by the User.

5. The User submits the updated recipe.

6. The system fetches the current prices of the ingredients from the marketplace and calculates the total cost of the recipe.

7. The system updates the recipe information on the database.

8. The system shows a message regarding a successful operation.

9. The use case ends successfully.

**Extensions (or Alternative Flows):**

\*a. At any time prior to step 5 in the basic flow, the User requests to cancel the operation:

1. The system shall display a message indicating that the operation was canceled.

2. The use case ends.

\*b. At any time, the System fails:

1. The system shall display a message indicating the type of the failure

2. The system updates its logs.

3. The use case ends.

4.a. Invalid Image

If the User cannot upload the image due to its size, then

1. The system displays a message that indicates the accepted size (See Supplementary Requirement WC-3 for the accepted image size) and asks the User to upload another image according to that size.
2. The use case resumes at step 3.

4.b. Missing entry

If a User input is missing, then

1. The system shows a message that indicates the name of the missing entry and asks the User to complete it.
2. The use case resumes at step 3.

4. c. Invalid amount

If the User enters an amount that cannot be created (See Supplementary Requirement WC-4 for valid amounts), then

1. The system shall display a message indicating the accepted amount interval.
2. The use case resumes at step 3.

**Supplementary Requirements:**

[SpReq: WC-1]: The recipe information contains the recipe name, description, image, dietary type (See Supplementary Requirement WC-5), ingredients, and amounts of the ingredients

[SpReq: WC-2]: The recipe information can be edited either by selecting (dietary type, ingredients), uploading (image), or typing (recipe name, description, and amounts of the ingredients).

[SpReq: WC-3]: The accepted image size is 50 Mb.

[SpReq: WC-4]: The valid amounts of the ingredients are in kg, and they are between 0,10-2.

[SpReq: WC-5]: The existing dietary types are regular, vegan, vegetarian, gluten-free, and keto.

**Open Issues**

* Dealing with a marketplace part is under discussion. We will use a database instead in the first release.

**Frequency of Occurrence:** Could be nearly continuous.