Smasko

Arash Amiry, Nils Bengtsson Svanstedt, Gustav Berndtzen, Anton Börås

March 2024

1 Introduction

Today, many of us have problems choosing recipes to cook and remembering recipes we previously enjoyed. Therefore, we created this application to solve these problems. In our app, the user can for example create recipes, search for them, edit them, but also delete them. On the homepage the user is met with the entire list of recipes, allowing for the user can get an overview of all recipes and their rating, while also having the ability to mark a recipe as a favorite. If the user presses the "See ingredients" button the list of ingredients for that particular recipe will be shown in the sidebar. The user can also go to the tab "Favorite recipes" in the header, to view only the recipes marked as favorites. When the user has decided on what recipe to cook they can click on the recipe card to get to the detailed view. They can then choose the amount of servings and then follow the instructions while cooking the dish. While viewing the recipe they can also choose to edit or delete the recipe.

1.1 Link to GitHub repo

`https://github.com/ArashAmiry/Smasko`

2 The use cases

The application consists of the following use cases:

1. The user can easily get an overview of all the recipes
2. The user can easily view all the ingredients for a recipe
3. The user can search for a specific recipe
4. The user can mark recipes as favorites
5. The user can see all the details of a recipe
6. When following a recipe, the user can choose the number of servings → the amount of each ingredient changes accordingly (complex)
7. The user can mark a step as done
8. The user can create a recipe
9. The user can set a rating for the recipe
10. The user can edit a recipe
11. The user can delete a recipe

2.1 Overview of all the recipes

The first thing the user is presented with when entering the application is a page where all the recipes are present, which can be seen below:

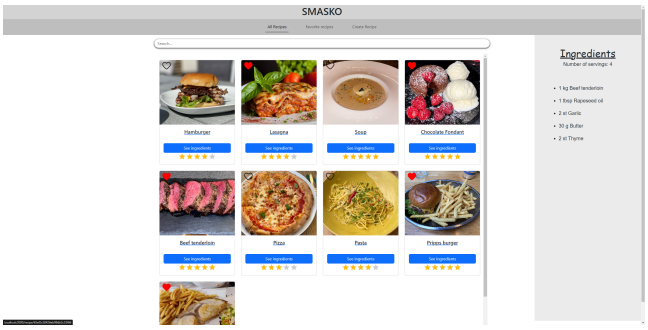

Figure 1: The page for all recipes

Here, the reviewer will be able to see cards for all the recipes, and each card contains the image, name, rating, and the ability to mark the recipe as a favorite, as well as a button for an easy view of the ingredients for that recipe. The rating makes it easy for the user to remember how much they enjoyed the recipe.

When the button "See ingredients" is clicked, it toggles the right sidebar, which makes it easy to see all the ingredients for the recipe, for example, if one wants to write a shopping list.

After a while, the application will contain many recipes, potentially making it hard for the user to find the desired one. This is why we have included the use case of searching for recipes. The search bar on the top of the page makes it easy for the user to search for a specific recipe. As can be seen below, when searching for the Chocolate fondant, only that is displayed:

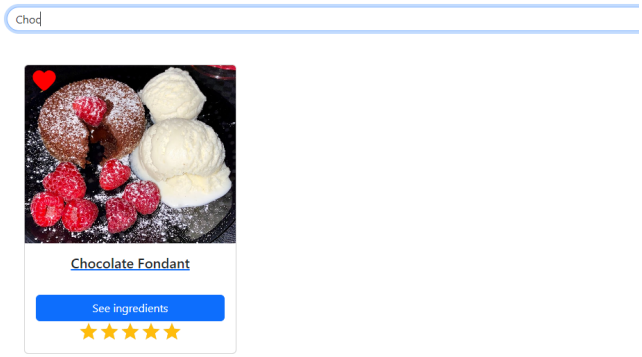

Figure 2: Searching for the recipe for Chocolate Fondant

2.2 Favorite Recipes

After marking a recipe as a favorite, by clicking on the heart on a recipe card, it will be visible on the "Favorite Recipes" page. Here the user keeps track of all their favorite recipes and this page has all the functionality that the "All recipes" page has.

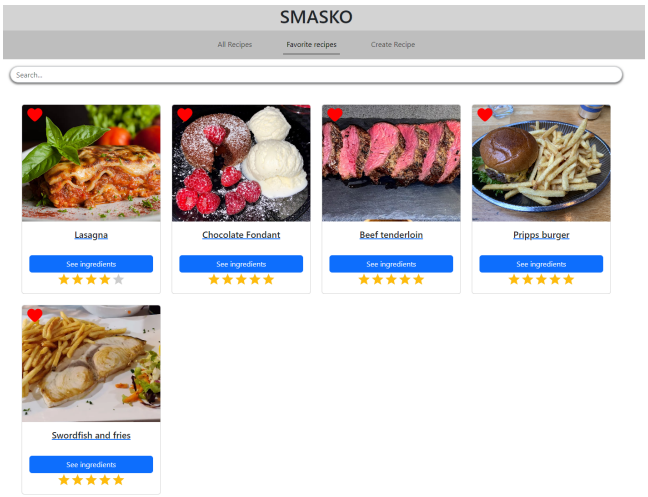

Figure 3: Showing the favorite recipes

2.3 Recipe Details

The user can view the details of a recipe by pressing on the corresponding card on the homepage shown in Figure 1. Here they can see details about the recipe such as the name, image, instructions, number of servings, ingredients, and rating. This detailed view can be seen in Figure 4 below.

We added a complex use case for when the user decides how much food they want to make, they can select the number of servings. Selecting a new number of servings will calculate the amount of ingredients needed. This complex use case can be seen in Figure 5 below.

When following the recipe the different instruction steps can be marked as done making it easier to indicate which steps the user has already performed and keeping track of the process. This can be seen in Figure 6 below.

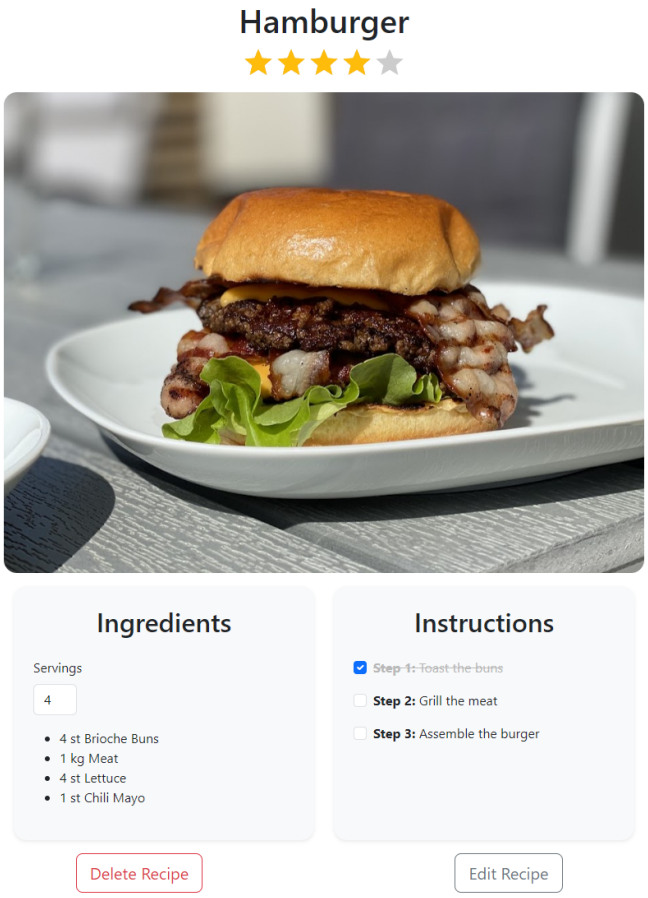

Figure 4: Details for a hamburger

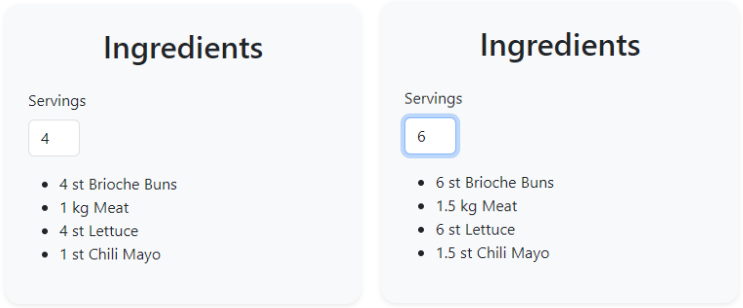

Figure 5: Complex use case where changing the number of servings updates the amounts for the ingredients

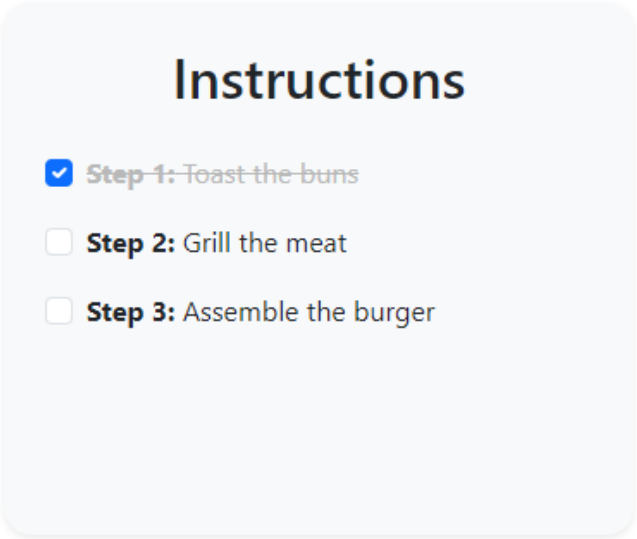

Figure 6: Marking a step as done

2.4 Creating a Recipe

For the application to be useful for the user, there has to be a use case where recipes can be created. To do this, the user navigates to the tab "Create Recipe" in the header, which brings the user to the page shown below:

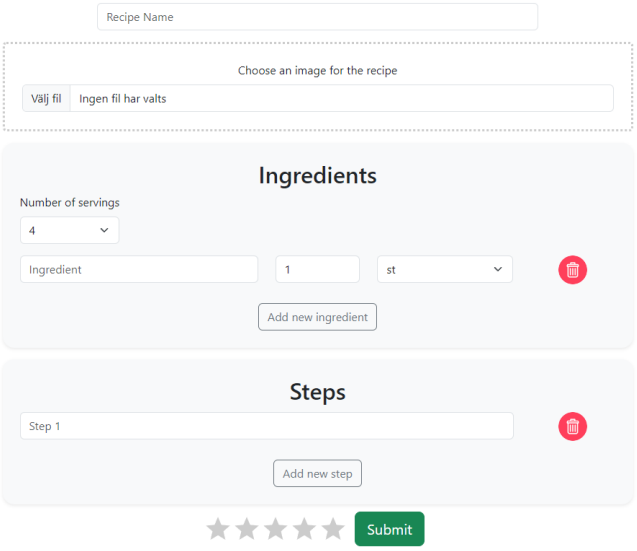

Figure 7: The initial view for creating a recipe

On this page, the user will first give the recipe a name, followed by an image of what the meal will look like after the cooking has been done.

After that, the user will have to enter the number of servings the ingredients are intended for, which by default is set to four servings. Each ingredient will consist of a name, an amount, and a unit. After that, more ingredients can be added, and some ingredients can also be removed if any mistakes have been made. When all the ingredients have been added, the user will write the necessary steps to cook the meal in the recipe. As for the ingredients, a step can also be deleted if any mistakes are to be made. At the end of the create recipe page, the user can give the recipe a rating from one to five, before finally saving the recipe to the database. This will make it easier for the user to remember which recipes they enjoyed when browsing all the recipes.

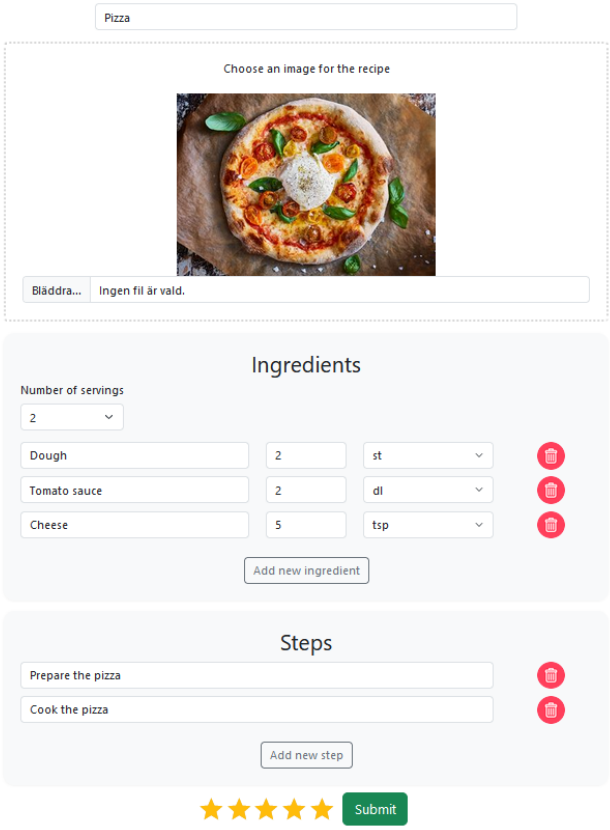

Figure 8: The view right before submitting the recipe

2.5 Editing a Recipe

The user might want to edit a recipe in the future according to their liking, and therefore we implemented an editing functionality. The edit view can be reached from the recipe details page in Figure 4, by pressing "Edit Recipe" in the bottom right corner. Here, the recipe will be shown in a familiar view, similar to the one shown in Figure 8. Here the user can change the recipe and the rating however they want, and save those changes by pressing the save button. In the figure below, the pizza recipe will now be updated with adjusted amounts for the ingredients, and a new step being added.

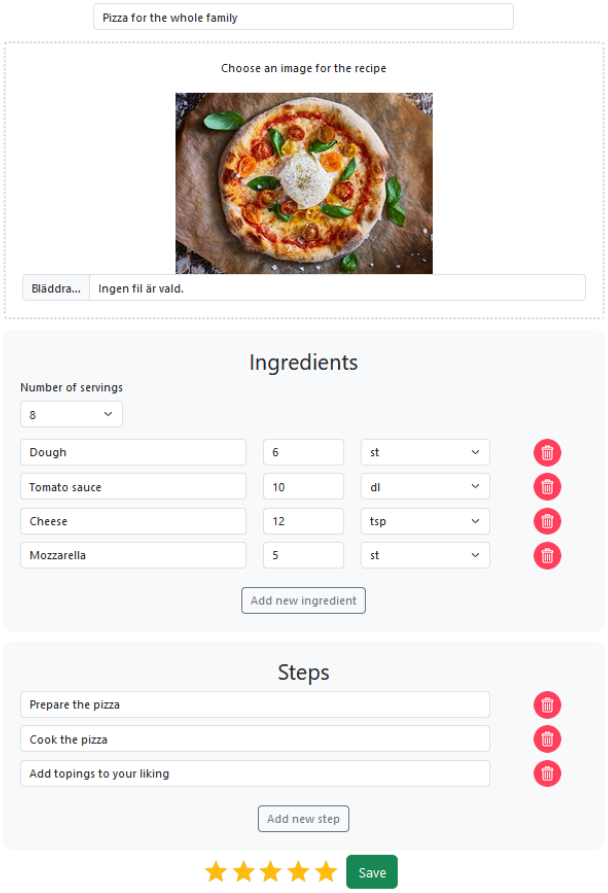

Figure 9: The view for editing the recipe

2.6 Deleting a Recipe

Users might have old recipes that are no longer relevant, and therefore we implemented the possibility to delete those recipes. The user can access the delete button from within the recipe details page, shown in the bottom left of Figure 4. After pressing the button they need to accept that the recipe will be deleted forever.

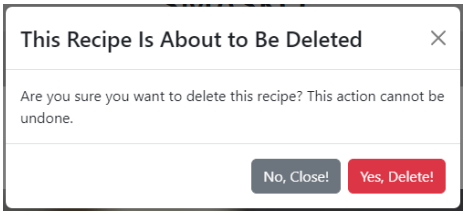

Figure 10: The last step before the recipe is deleted

3 Installation guide for the web application

Clone the repository

Open your command-line interface (CLI) and run the following command to clone the project repository:

- git clone https://github.com/ArashAmiry/Smasko.git

Add the database password

In the directory "Smasko/server", add the file "db_password.ts" (see more details in the submission comment on Canvas).

Navigate to the project directory:

Change your current working directory to the project folder by entering the following in your CLI:

- cd Smasko/client

Start the development server

Execute the following commands in the following order to start the development server through your CLI:

- npm install
- npm run start

Open a new terminal tab and ensure that you now are in the Smasko/server folder, and then execute the following commands:

- npm install
- npm run dev

3.1 Image of homepage

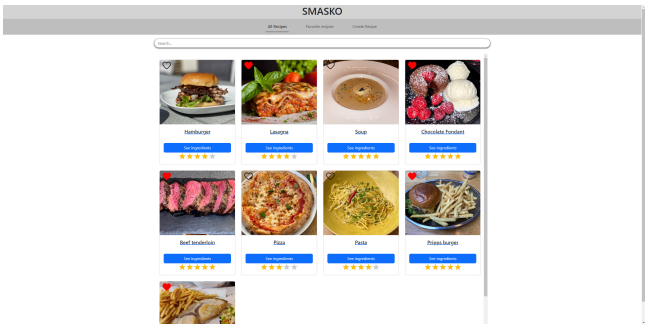

Figure 11: The homepage of the application

4 Design

The project consists of numerous libraries, including Mongoose which is used as an interface between the server and the NoSQL database MongoDB that we also utilize. We have also used the Express library in combination with the framework Node. Node is a framework that makes it possible to create server-side applications and execute JavaScript code on the server side, while Express is a web application framework that runs on top of Node, providing a more streamlined and structured approach to server-side web development. Express provides powerful functions, such as routing, which facilitates the process of managing HTTP requests.

The frontend of our project used other libraries, for instance, the Axios library, to manage the HTTP requests that are sent to the server. The library used for constructing a responsive design was the renowned library Bootstrap, offering access to pre-made responsive components. To enable users to rate their recipes with stars, the library react-simple-star-rating was imported. Moreover, to mark a recipe as a "favorite", the react-sandbox/heart library was utilized. While waiting for the recipe to be displayed on the screen, a loading screen appears, with animations provided by the React Spinners library. However, the most fundamental technology, used throughout the frontend code was the JavaScript library React. The React library ensures an interactive UI and a smooth user experience, partly through the Single Page Application (SPA) technology.

The programming language used for both the frontend and backend was TypeScript, which is built on JavaScript but is based on object-oriented programming. To test the code written in both the frontend and backend, tests were constructed using the JavaScript testing framework Jest. Additionally, the SuperTest library was used to test the backend API routes and endpoints.

4.1 API Specification

Get All Recipes

- URL: /
- Method: GET
- Description: Retrieves all recipes.
- Response:
  - 200 OK with an array of Recipe objects.
  - 500 Internal Server Error if there's an issue with the server.

Get Favorite Recipes

- URL: /favorites
- Method: GET
- Description: Retrieves all favorite recipes.
- Response:
  - 200 OK with an array of Recipe objects.
  - 500 Internal Server Error if there's an issue with the server.

Get Recipe by ID

- URL: /:id
- Method: GET
- Description: Retrieves a recipe by its ID.
- Parameters:
  - id: Recipe ID (path parameter).
- Response:
  - 200 OK with the Recipe object if found.
  - 400 Bad Request if the recipe with the given ID doesn't exist.
  - 500 Internal Server Error if there's an issue with the server.

Add New Recipe

- URL: /
- Method: POST
- Description: Adds a new recipe.
- Request Body: Recipe object (excluding the ID).
- Response:
  - 201 Created with the newly added Recipe object.
  - 400 Bad Request if the request body is invalid.
  - 500 Internal Server Error if there's an issue with the server.

Delete Recipe by ID

- URL: /:id
- Method: DELETE
- Description: Deletes a recipe by its ID.
- Parameters:
  - id: Recipe ID (path parameter).

  Response:
  - 204 No Content if the recipe is successfully deleted.
  - 400 Bad Request if the recipe with the given ID doesn't exist.
  - 500 Internal Server Error if there's an issue with the server.

Update Recipe by ID

- URL: /:id
- Method: PUT
- Description: Updates a recipe by its ID.
- Parameters:
  - id: Recipe ID (path parameter).
- Request Body: Updated Recipe object.
- Response:
  - 200 OK with the updated Recipe object.
  - 400 Bad Request if the ID in the request body doesn't match the ID in the path or if the request body is invalid.
  - 500 Internal Server Error if there's an issue with the server.

Update Recipe Like Status

- URL: /:id
- Method: PATCH
- Description: Updates the like status of a recipe by its ID.
- Parameters:
  - id: Recipe ID (path parameter).
- Request Body: liked: boolean .
- Response:
  - 200 OK with a boolean indicating whether the recipe was successfully updated.
  - 400 Bad Request if the request body is invalid.
  - 500 Internal Server Error if there's an issue with the server.

5 Responsibilities

We have developed almost everything with pair programming, so no one has had any specific responsibilities.