

# Final project app development

**Submission date no later than 8.12.2023 23:59 by e-mail to [kelvin.homann@gmail.com](mailto:kelvin.homann@gmail.com) with the .zip file of the frontend project attached.**

**If the project is submitted as a group, include a list of all participants in the e-mail.**

**If you deliver alone, you only have to deliver the basic functions, the groups have to implement all functions.**

## **Notes on evaluation:**

I will evaluate the charges as follows:

A solution with all the required functions and the correct use of what has been learned from the exercises and the slides results in 100 points and thus a 1.0. The individual submissions must implement the basic functions for the complete points, while the group work must also implement the extended functions for the complete points. have to implement.

I will deduct points for each missing function. Since a function is also theoretically "wrong" in the sense of the exercises and PowerPoint slides can be implemented, I will decide on a case-by-case basis how many points will be deducted.

As there can always be several equally good solutions to a problem in software projects, the evaluation for each submission is always individual.

## Job description

Develop a React app to manage recipes. Uses the provided backend (backend.zip) and implements the functions described on the following pages.

Functions. Applies the learned MVC pattern with mobx where necessary and uses the fetch API to communicate with the backend.

## Basic functions

Everyone must hand over these functions, even if you hand over the task alone.

## Recipes overview

Develops an overview of all recipes that are saved in the backend. This overview is visible directly in the root URL `"/`.

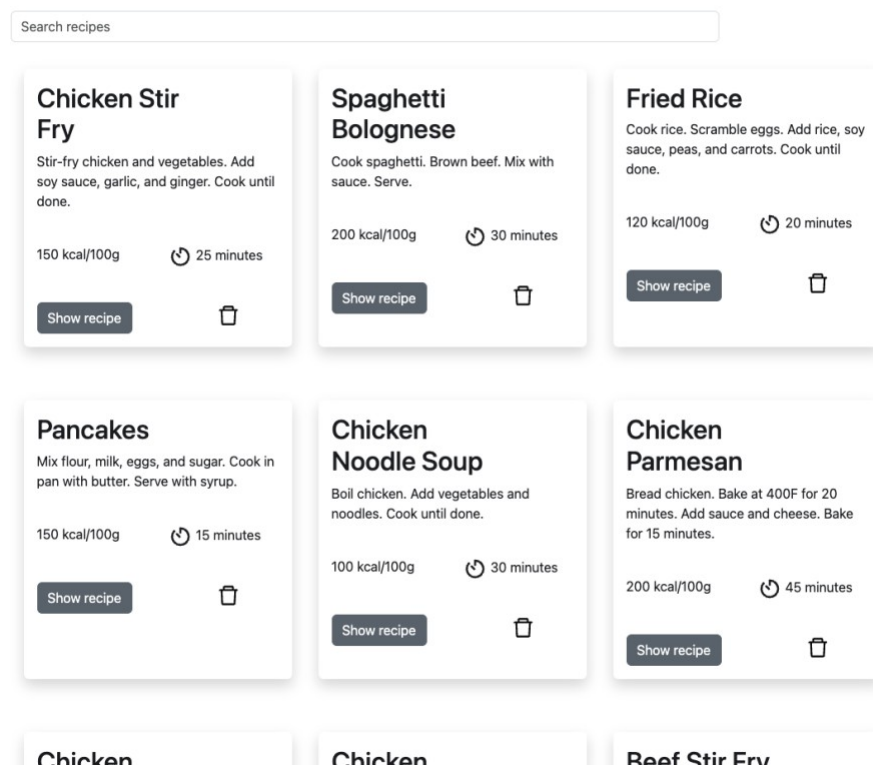
The individual recipes should be rendered with the following information:

- The name of the recipe
- The cooking guide
- The calories per 100g
- The cooking time

Each individual recipe also implements the following functions:

- a button to delete the recipe (with subsequent re-rendering of the list if the recipe has been deleted)
- a link to navigate to the detailed view of the recipe

## My Recipes



## Search recipes

All recipes in the overview should be able to be filtered using a search. This filter searches using the names of the recipes and only displays the recipes that match the filter.

## My Recipes

### Chicken Stir Fry

Stir-fry chicken and vegetables. Add soy sauce, garlic, and ginger. Cook until done.

150 kcal/100g

 25 minutes


Show recipe



### Chicken Noodle Soup

Boil chicken. Add vegetables and noodles. Cook until done.

100 kcal/100g

 30 minutes

Show recipe



### Chicken Parmesan

Bread chicken. Bake at 400F for 20 minutes. Add sauce and cheese. Bake for 15 minutes.

200 kcal/100g

 45 minutes


Show recipe



### Chicken Tacos

Cook chicken. Chop vegetables. Assemble tacos. Serve with sour cream.

150 kcal/100g

 30 minutes


Show recipe



### Chicken Salad

Cook chicken. Chop vegetables. Assemble salad. Serve with dressing.

100 kcal/100g

 30 minutes

Show recipe



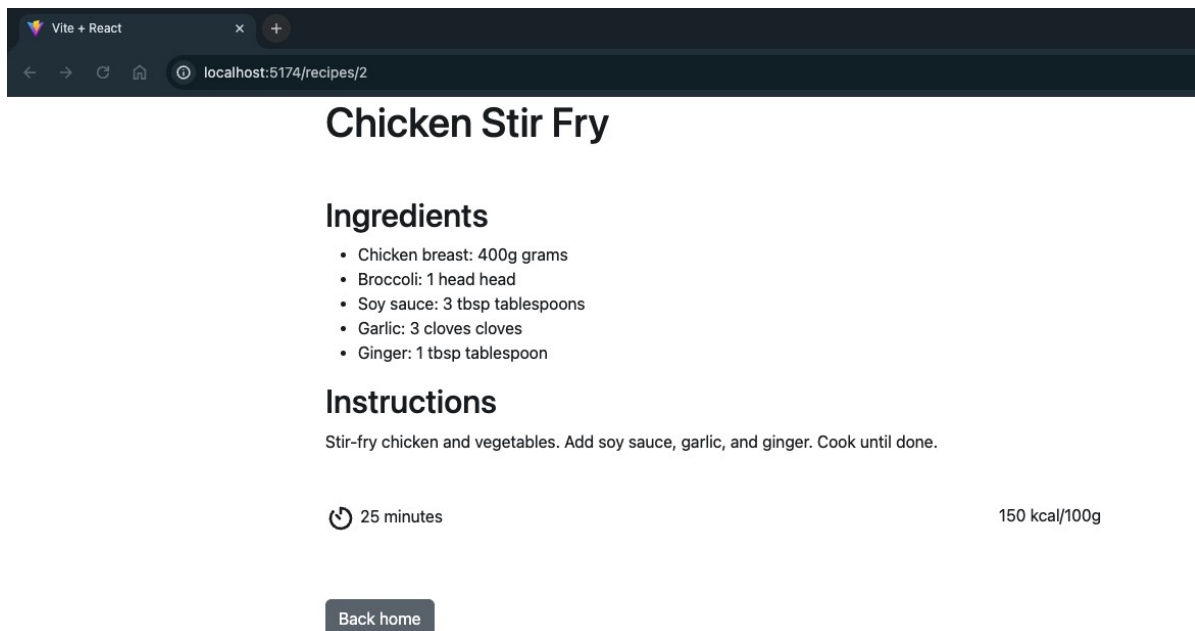
## Recipe detailed view

Develops a detailed view for a single recipe. This detailed view can be accessed via the URL **"/recipes/:recipeId"** can be viewed. To access this detailed view, click on the "Show recipe" button on the overview page.

The overview page contains the following information:

- The name of the recipe
- A list of ingredients
- The cooking guide
- The calories per 100g
- The cooking time

At the bottom of the page there should be a link back to the main page.



## Styling with CSS

Styling is not the main focus and is not part of the evaluation. The basic functions are important. But if you want to implement styling, feel free to use Bootstrap.

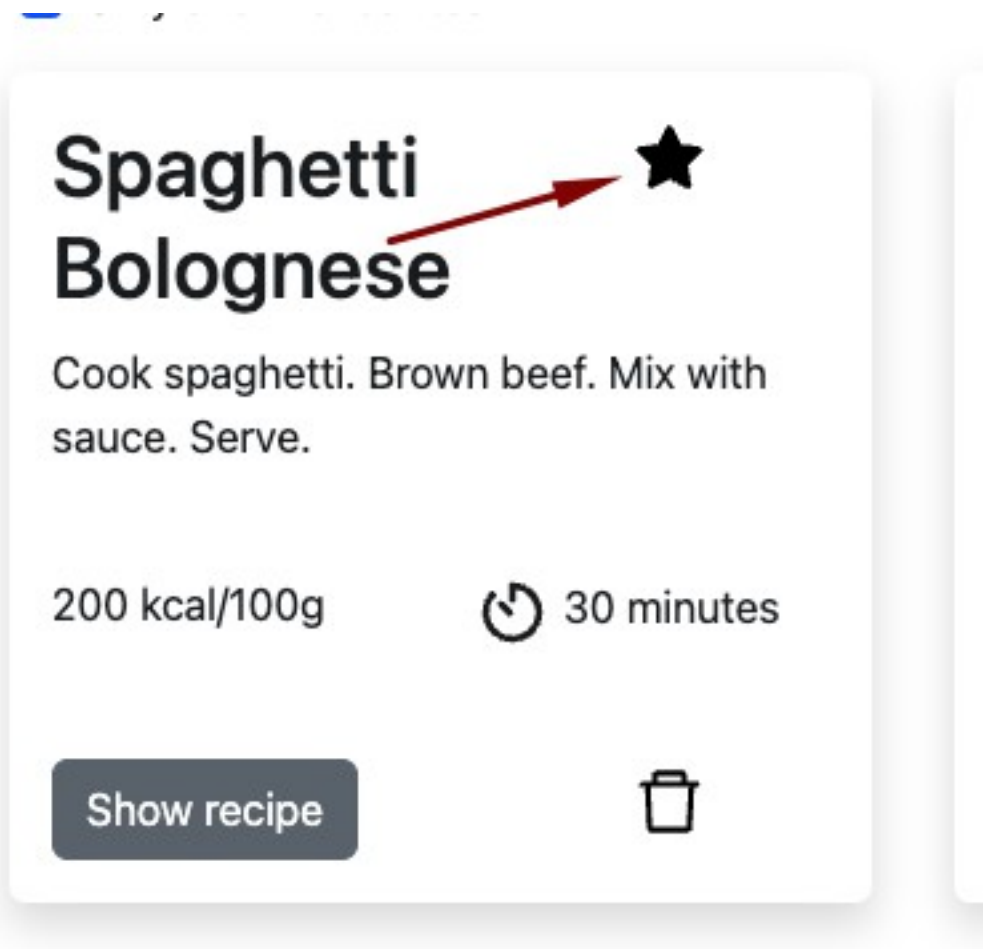
## Functions for group delivery

These functions only need to be developed if you are in a group.

### Mark favorites

Adds the function to mark each recipe as a favorite. To do this, add a button which, when clicked, selects or deselects a recipe as a favorite.

Use the PUT route `/recipes/:id/favorite` in the backend.



## Filter for favorites

Expands the overview with a checkbox to filter by favorites. If the checkbox is ticked, only favorites are displayed in the list. If the checkbox is unchecked, all recipes are displayed.

## Filter off

# My Recipes

☐ Only show favourites

**Spaghetti  
Bolognese**



**Pancakes**

Mix flour, milk, eggs, and sugar. Cook in pan with butter. Serve with syrup.

## Filter on

# My Recipes

[New recipe +](#)☒ Only show favourites

**Spaghetti  
Bolognese**



Cook spaghetti. Brown beef. Mix with sauce. Serve.

200 kcal/100g

30 minutes

[Show recipe](#)

**Pancakes**



Mix flour, milk, eggs, and sugar. Cook in pan with butter. Serve with syrup.

150 kcal/100g

15 minutes

[Show recipe](#)

**Chicken  
Tacos**



Cook chicken. Chop vegetables. Assemble tacos. Serve with sour cream.

150 kcal/100g

30 minutes

[Show recipe](#)

## Create new recipes

Extends the application with a page for creating new recipes. This can be viewed via the URL **"/recipes/new"**.

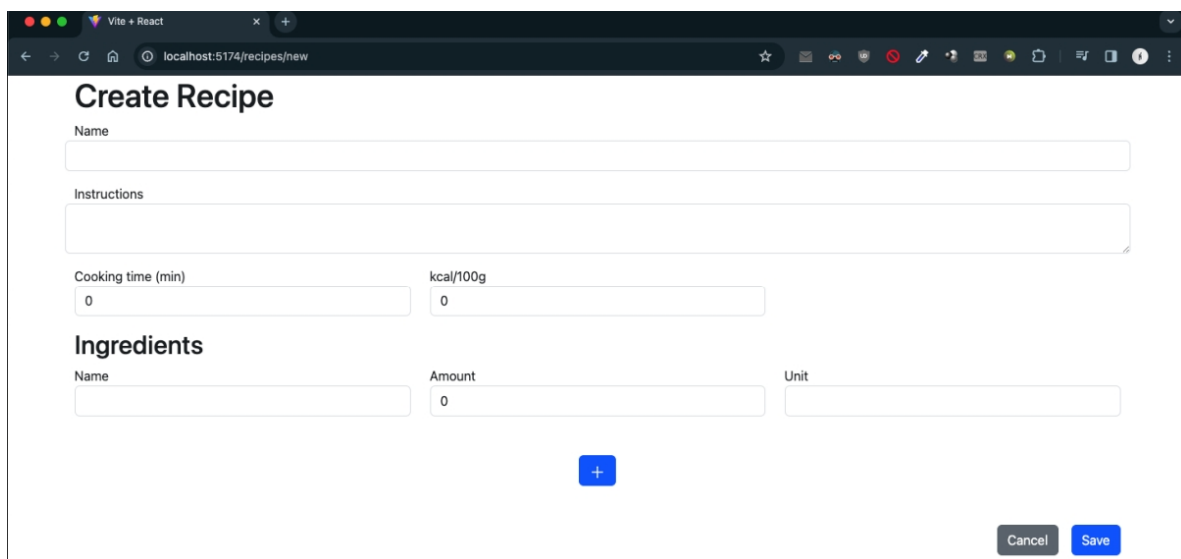
It contains a title and a form with the following fields:

- The name of the recipe
- The cooking guide
- The calories per 100g
- The cooking time
- The list of ingredients
  - Name of the ingredient
  - Amount of ingredient
  - Unit of quantity

Below the ingredients there is a "+" button which can be used to add further ingredients.

There are two buttons at the bottom of the page:

- Cancel button: Navigates back to the overview page
- Save button: This button saves the recipe and navigates to the detailed view of the recipe



The screenshot shows a web browser window with the address bar displaying 'localhost:5174/recipes/new'. The page title is 'Create Recipe'. The form contains the following fields:

- Name**: A text input field.
- Instructions**: A text area.
- Cooking time (min)**: A text input field with the value '0'.
- kcal/100g**: A text input field with the value '0'.
- Ingredients**: A section with three input fields: **Name**, **Amount** (with value '0'), and **Unit**.

Below the ingredients section is a blue button with a white '+' sign. At the bottom right of the form are two buttons: 'Cancel' and 'Save'.

## Start backend instructions

Unzip the attached backend.zip into a new folder "recipes".

Then open the recipes folder in the terminal. Now execute the following commands in the terminal.

**cd backend**

**npm install**

**npm run start**

The backend now runs on <http://localhost:3000> and provides the following routes available. Under the GET routes you will already find some recipes that are available at the start of the server.

HTTP method	Route	Description
GET	<a href="http://localhost:3000/recipes">http://localhost:3000/recipes</a>	Reads out all recipes
GET	<a href="http://localhost:3000/recipes/:id">http://localhost:3000/recipes/:id</a>	Reads a recipe with a specific ID from
POST	<a href="http://localhost:3000/recipes/:id">http://localhost:3000/recipes/:id</a>	Creates a new recipe
PUT	<a href="http://localhost:3000/recipes/:id/favorite">http://localhost:3000/recipes/:id/favorite</a>	Sets the "isFavorite" flag of a new recipe
DELETE	<a href="http://localhost:3000/recipes/:id">http://localhost:3000/recipes/:id</a>	Deletes a recipe with a specific ID

The data structure for a recipe object is as follows:

```
{
  "id": "6",
  "name": "Chicken Parmesan",
  "ingredients": [
    { "name": "Chicken breast", "quantity": "400g", "unit": "grams" },
    { "name": "Tomato sauce", "quantity": "400g", "unit": "grams" },
    { "name": "Mozzarella", "quantity": "200g", "unit": "grams" },
    { "name": "Bread crumbs", "quantity": "1 cup", "unit": "cup" },
    { "name": "Parmesan", "quantity": "1 cup", "unit": "cup" }
  ],
  "totalCookingTimeMinutes": 45,
  "calories100g": 200,
  "instructions": "Bread chicken. Bake at 400F for 20 minutes. Add sauce and cheese. Bake for 15 minutes.",
  "isFavorite": false
}
```



You can also find a presentation of the finished project in the recording of day 5.

## **Helpful literature and tools:**

**Read parameters from URL in React-router:**

<https://reactrouter.com/en/main/hooks/use-params>

**For the icons:**

<https://reactsvgicons.com/react-svg-icons-guide>

**For optional styling: Bootstrap**

<https://getbootstrap.com/docs/4.0/components/input-group/>

<https://getbootstrap.com/docs/4.0/utilities/spacing/>

<https://getbootstrap.com/docs/4.1/utilities/shadows/>

<https://getbootstrap.com/docs/4.0/layout/grid/>

**Mobx documentation:**

<https://mobx.js.org/README.html>