A logo of a chef

Description automatically generated

# Programmer Manual

A18

Dor Shabat - 316575620  
Yuval Rozner - 207756552  
Niv Dahan - 308215086  
Nitzan Ezra - 208697334

Table of Contents

[1. Introduction 3](#_Toc162716925)

[1.1 System Architecture Overview 3](#_Toc162716926)

[1.1.1 Key Components 3](#_Toc162716927)

[1.1.2 Key Libraries and Frameworks 3](#_Toc162716928)

[2. Key Files and Folders: 4](#_Toc162716929)

[2.1 Index.html 4](#_Toc162716930)

[2.2 recipes.json 4](#_Toc162716931)

[2.3 public folder 4](#_Toc162716932)

[2.4 src folder 4](#_Toc162716933)

[2.5 package.json 4](#_Toc162716934)

[3. Key Components 5](#_Toc162716935)

[3.1 App.jsx 5](#_Toc162716936)

[3.2 Always loaded components 5](#_Toc162716937)

[3.2.1 Head.jsx 5](#_Toc162716938)

[3.2.2 NavigationBar.jsx 5](#_Toc162716939)

[3.2.3 Tail.jsx 5](#_Toc162716940)

[3.3 Specific page components 5](#_Toc162716941)

[3.3.1 RecipesList.jsx 6](#_Toc162716942)

[3.3.2 Recipe.jsx 6](#_Toc162716943)

[3.3.3 CookWithMe.jsx 6](#_Toc162716944)

[4. Key functions 7](#_Toc162716945)

[4.1 Dynamic Routing 7](#_Toc162716946)

[4.2 State Management for Recipe Interaction 7](#_Toc162716947)

[4.3 Ingredient Scaling 7](#_Toc162716948)

[4.4 Accessibility and Navigation 7](#_Toc162716949)

[4.5 Recipe Selection 8](#_Toc162716950)

[4.6 Persistent UI Elements 8](#_Toc162716951)

## Introduction

This manual serves as a guide for developers working on the Icook – Interactive Cooking Assistant Web Application stored on GitHub repository and hosted on GitHub Pages. It details the Web architecture, code organization, interaction between the components of React, and how to use and modify the React components with tailwind for changing the UI.

### System Architecture Overview

The application integrates ReactJS for web coding, a json file as a database for storing and retrieving data, and tailwind as a UI design framework.

#### Key Components

* **ReactJS:** the website coded using React framework and uses the components architecture.
* **Database:** Json file used as a data file storing all the recipes the websites can offer.
* **UI**: the UI design by using tailwind framework.

#### Key Libraries and Frameworks

* **Vite**: For creating web application project.
* **ReactJS:** For developing web components for UI elements.
* **Tailwind:** For designing the UI elements.
* **GitHub Pages:** For hosting the website UI.

## Key Files and Folders:

We utilize Vite to create our project, which offers a template for a web project using the React framework.

### Index.html

* **Single page website:** The single html page of our project, which determine the website meta data and defining a root used for the React UI components.

### recipes.json

* **Database:** Recipes data and information stored as a dictionary in a Json file and being imported in the JS code.
* **Structure (dictionary structure):**

recipes : {  
 id : {  
 title,   
 allergies,  
 difficulty,  
 ingredients[],  
 amounts[],  
 instructions[],  
 durations[]  
 },   
 id : {  
 title,   
 allergies,  
 difficulty,  
 ingredients[],  
 amounts[],  
 instructions[],  
 durations[]

}, ….  
}

### public folder

* **Shared Files:** The public folder stored the website images, the recipes and dishes pictures and the favicon icon file, the logo image file, and more.

### src folder

* **Components:**  The src folder, holds all the React UI components of our website. (full components descriptions in the next section).

### package.json

* **Scripts:** Hold the project scripts like: build, deploy and etc..
* **Dependencies**: Hold the project dependencies like: react version, react-dom, react-router-dom, and tailwind framework version.

## Key Components

The Icook website uses React framework for rendering UI components written in JS on a single html page. Those components designed by using Tailwind framework for adding css class during the JSX code.

We implement some components as a reusable component like the head, tail, navigation bar and more.

**Descriptions of some of those main components:**

### App.jsx

* Root component of the Icook application.
* Utilizes `react-router-dom` for handling routing.
* Imports recipe data and initializes state for titles derived from the data.
* Defines routes for home, individual recipes, and CookWithMe view.
* Ensures `Head` and `Tail` components are always rendered across different routes.

### Always loaded components

These components are foundational to the Icook website and persist across all views, providing a consistent user experience:

#### Head.jsx

* Displays the website's logo.
* Acts as a navigational link to the home page.
* Responsive design that adapts to various screen sizes.

#### NavigationBar.jsx

* Provides back and home navigation buttons.
* Displays the current page title.
* Uses `useHistory` hook for back navigation.
* Responsive layout with hover effects for buttons.

#### Tail.jsx

* Serves as the website's footer.
* Provides copyright information.
* Lists the names of the creators.
* Simple, responsive design for text alignment and visibility.

### Specific page components

These components are specific to certain routes or pages within the Icook website and are rendered according to user interaction and navigation:

#### RecipesList.jsx

* Displays recipes in a responsive grid layout.
* Handles click events to navigate to recipe details.
* Each recipe item includes an image and a label.
* Implements shadow and hover effects for interactivity.

#### Recipe.jsx

* Detailed view of a selected recipe.
* Shows allergies, difficulty level, ingredients, amounts, and instructions.
* Features a dish count selector for ingredient amount scaling.
* Responsive design with elements and text scaling for various devices.

#### CookWithMe.jsx

* Interactive step-by-step cooking guide.
* Displays instructions, estimated duration, and images for each step.
* Includes a progress bar indicating cooking progress.
* Navigation buttons to move between steps.
* Images and layout responsively adjust to the screen size.

## Key functions

In this section, we outline several important functionalities that are integral to the Icook website, enabling interactive user experiences and the site's core features.

### Dynamic Routing

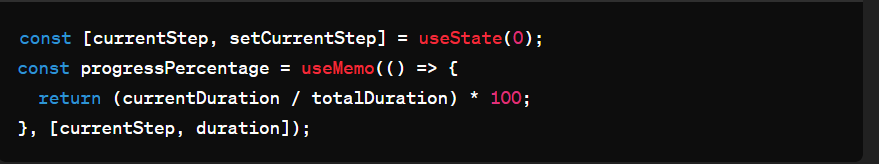
* Utilizes react-router-dom to handle navigation between different components without reloading the page.
* Implements parameterized routes to display recipe details dynamically based on user selection.

A computer code on a black background

Description automatically generated

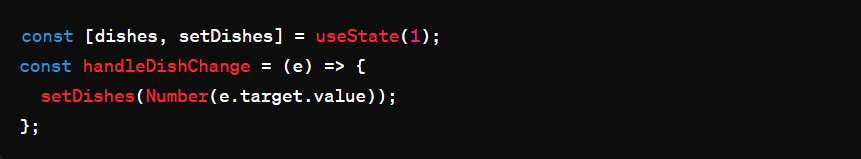
### State Management for Recipe Interaction

* Manages the state of the current recipe step within CookWithMe.jsx, allowing users to progress through the cooking instructions.
* Utilizes useState for state management and useMemo to calculate the progress percentage based on completed steps.



### Ingredient Scaling

* Provides functionality within Recipe.jsx to adjust the quantities of ingredients based on the number of dishes the user wishes to prepare.
* The handleDishChange function updates the state based on the user's selection from the dropdown.



### Accessibility and Navigation

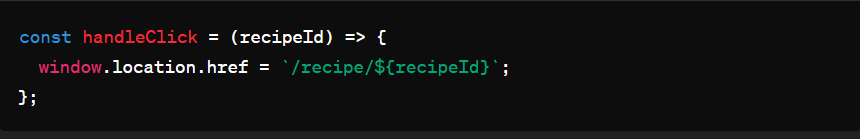
* The NavigationBar.jsx component employs the useHistory hook to programmatically navigate users back to the previous page, enhancing accessibility and user experience.

A black screen with a black background

Description automatically generated

### Recipe Selection

* In RecipesList.jsx, the handleClick function is responsible for redirecting users to the detailed view of the clicked recipe, showcasing the use of imperative navigation.



### Persistent UI Elements

* The always-loaded components Head and Tail provide a persistent UI experience, with Head.jsx showcasing logo rendering and navigation home functionality.

A computer code on a black background

Description automatically generated

These functionalities represent a cross-section of the dynamic and interactive nature of the Icook website, demonstrating both frontend routing and state management capabilities, as well as user interaction design. The code snippets provide insight into how these features are implemented within the React framework, leveraging the utility of hooks and the Tailwind CSS framework for styling.