Dharvi Jayaprasat // A00034862

TÓMATE a recipes website

GitHub: https://github.com/dharvi-22/tomate.git

Live: https://tomate-tomate.vercel.app/

Purpose

‘Make cooking feel less like a chore and more like an enjoyable experience’

‘Remove the stress and demotivation often associated with solo cooking’

‘Simplify the process of cooking for one’

Target Audience

‘Individuals who frequently cook for themselves’

‘Individuals who struggle with motivation, portion control or finding meal ideas’

Ethics Considerations

GDPR Compliance

‘Tomate does not collect or store personal data.’

Accessibility

‘Tomate uses ARIA labels and semantic tags, as well as colour contrast and button that follow accessibility standards.’

Sustainability and Resource Efficient

‘Tomate encourages single-portion cooking. Promotes reducing food waste.’

After interviewing 6 individuals I discovered...

1

There's a tendency towards looking for **simpler, quicker meals** particularly when people have other responsibilities or need to accommodate specific needs.

2

There seems to be an **emotional connection to cooking,** with some people finding it relaxing and enjoyable, **while others feel it's a task or challenge.**

3

Cooking just for one, can feel like a hassle. **Portion size is a key factor** to motivation when cooking.

Freya

Freelance Animator | Creative, Social, Health-Conscious

Pain Points

- Dietary Restrictions
- Overwhelmed by complex recipes

Needs

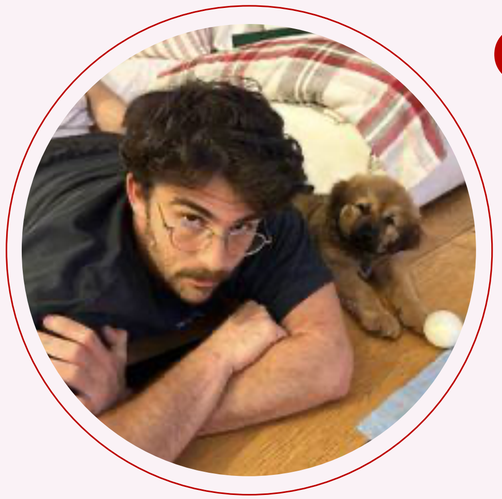
- Sense of confidence in the kitchen.
- Quick and easy recipes that align with her dietary plan and tastebuds

User Journeys - FigmaJam Link

‘I want to be able to find my preferred category without wasting too much time’

‘I want something which is not too difficult to cook’

‘I want to find easy vegan recipes’



Oliver

Software Developer | Practical, Introvert, Goal-oriented

Pain Points

- Time Constraints
- Picky Eater, meals that fit his mood, eating habits and serving portion

Needs

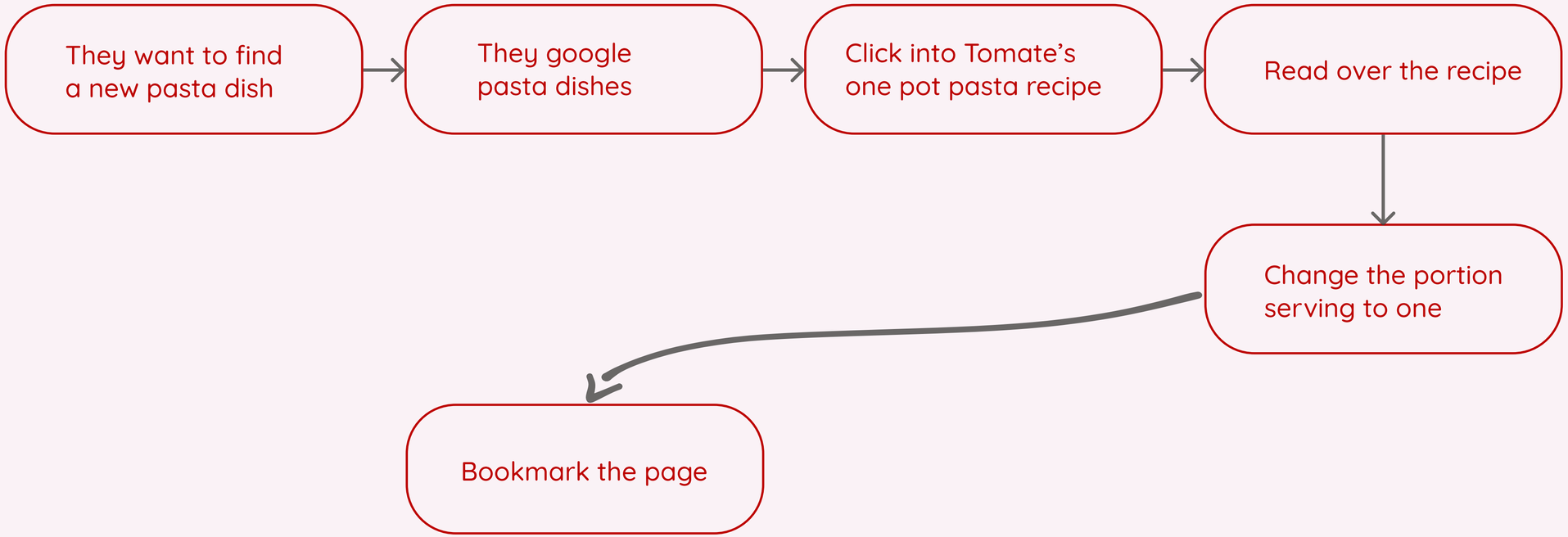
- Meals designed for one serving.
- Cooking meals that have him spend less time in the kitchen.

‘I want recipes that barely need any prep’

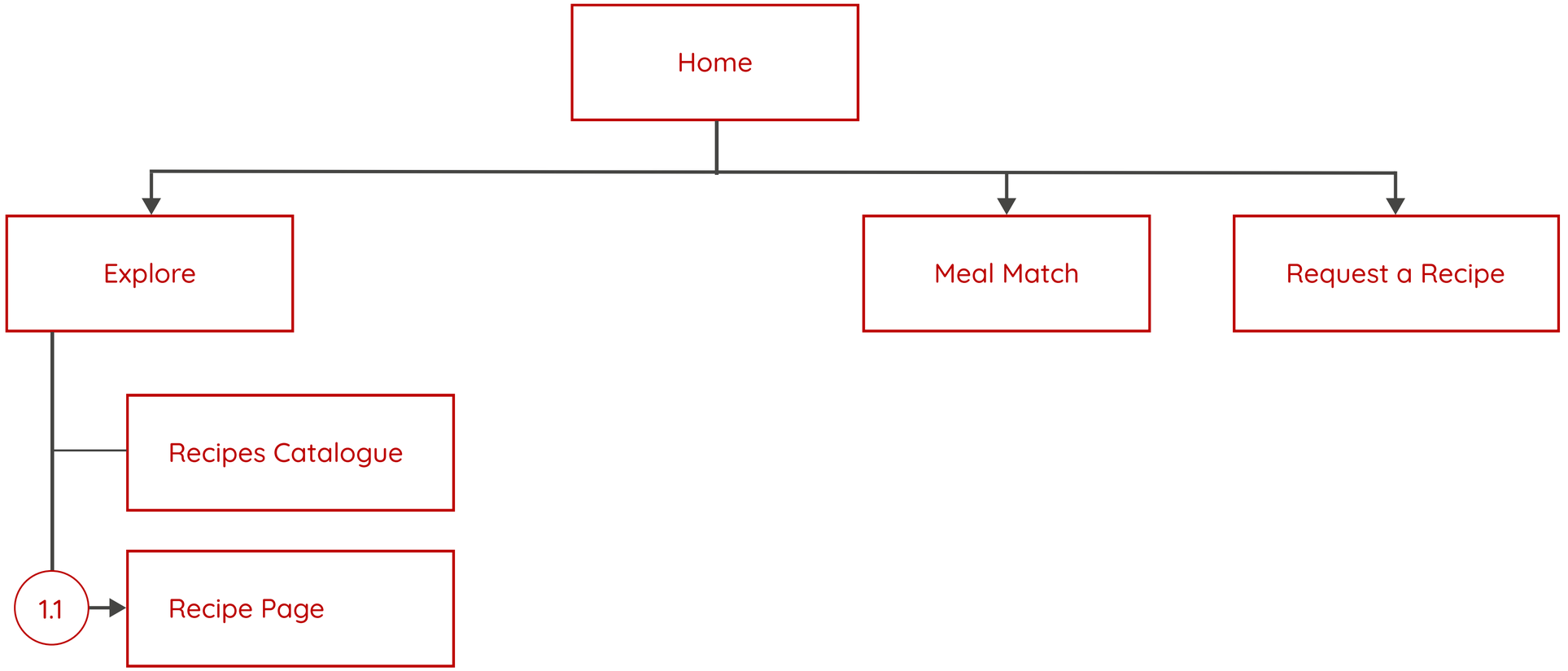
‘I want home cooked recipes portioned just for me’

‘I just want to browse through recipes quick’

Freya & Oliver’s User Journey



Site Map


Home
Explore
Meal Match
Request a Recipe
Recipes Catalogue
1.1
Recipe Page

Wireframes

Figma Link

Explore Recipes

Quick, Easy & Perfectly Portioned for One.

Find tasty recipes that suite your portion size.

Meal Match

Find tasty recipes that suite your portion size.

Search by Category

Find tasty recipes that suite your portion size.

Explore Recipes

I am looking for...

I know the category

Avocado Toast

Avocado Toast

Avocado Toast

Avocado Toast

Avocado Toast

Avocado Toast

view more

< Back

Avocado Toast

Prep Time	5 minutes
Cooking Time	7 minutes
Total Time	12 minutes

Ingredients

Serves

1 2 3 4

- [ ] 1 Egg
- [ ] 1 Avocado
- [ ] 2 Slices of Bread
- [ ] Mayonaise
- [ ] Rocket Leaves

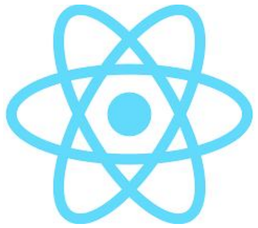
Frontend

React.js

- JSX for building the interface using reusable components, e.g: Header, Footer, Recipe List, Search Bar, Recipe Carousel

React Router Dom

- For client side routing to navigate between pages.



React API Context

- Used to manage and share the recipe data globally (across all the components) without pop drilling.

Styling

SASS (SCSS)

- Modular: each page and component styled using its own scss file, and imported to main scss to maintain consistent and organised styling across the app.

API Integration

Spoonacular API

- External public API for recipes, ingredients and categories.
- Used to fetch data such as recipe name, image and category

Enhancements

Responsive Design

- A mobile first layout, adapts seamlessly across all screen sizes (mobile, tablet, desktop).

Scroll To Top

- Component that ensures each page loads at the top using window.scrollTo() on route changes. Maintaining a consistent user experience.

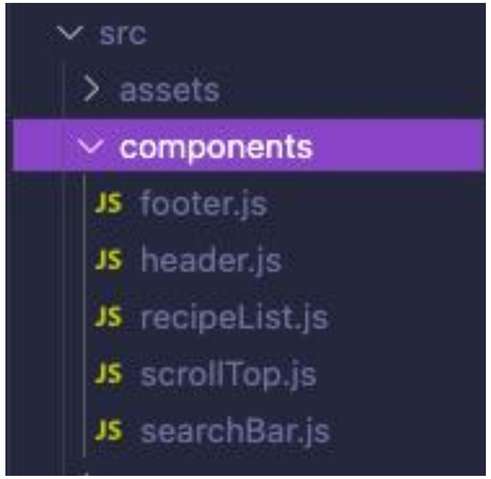
Custom Carousel

- Scrollable recipe catalogue slider with accessible navigation (buttons).

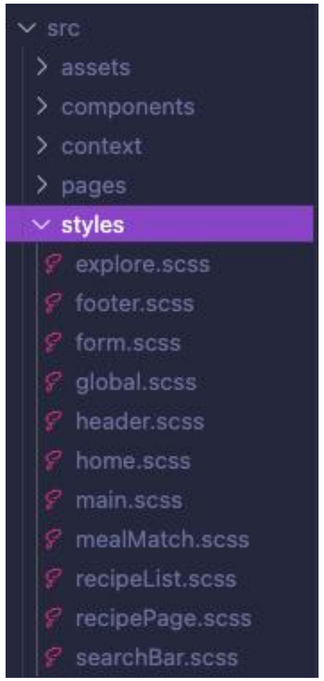
Form Validation

- Ensures users provide recipe name and category before submission. Error message are shown to improve usability

Component Architecture



1 My project follows a modular structure, breaking down UI into reusable components.



2 Each component is self-contained with its own SCSS styling, which enhances maintainability and scalability.

3 Common logic (fetching recipes) is primary via React Context API to make data accessible across all components.

Component: Recipe List

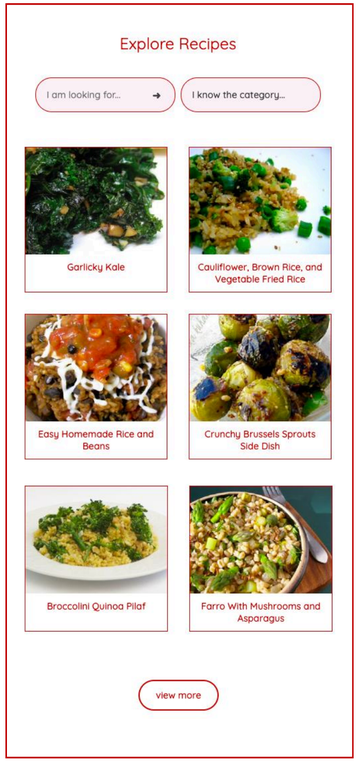
Recipe list to dynamically alter number of columns within a row based on screen size.

//recipe list code

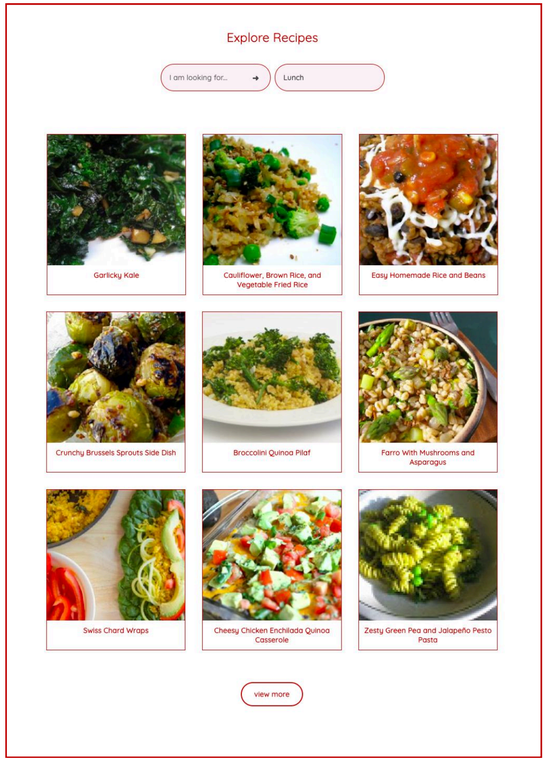
src > components > recipeList.js > RecipeList > useEffect() callback > updateVisibleCount

```
9   const RecipeList = () =>{
35
36      //load more recipes adds 1 row
37      const handleLoadMore = () => {
38          if (window.innerWidth >= 1024) {
39          //desktop load 4 more
40              setVisibleCount((prev) => prev + 4);
41          //tablet load 3 more
42          } else if (window.innerWidth >= 768) {
43              setVisibleCount((prev) => prev + 3);
44          //mobile load 2 more
45          } else {
46              setVisibleCount((prev) => prev + 2);
47          }
48      };
49
50      return (
51          <div className="recipe-grid">
52              {recipes.slice(0, visibleCount).map((recipe) => (
53                  <Link key={recipe.id} to={`/recipe/${recipe.id}`} className="recipe-card">
54                      <img src={recipe.image} alt={recipe.title}/>
55                      <p>{recipe.title}</p>
56                  </Link>
57              ))}
58              <div className="load-more-container">
59                  {visibleCount <= recipes.length && (
60                      <button className= "load-more" onClick={handleLoadMore}>
61                          view more
62                      </button>
63                  )}
64              </div>
65          </div>
66      );
67  };
68
69  export default RecipeList;
```

//mobile



//tablet



Spoonacular + React Contex API

//Shared and managed globally
//Stored in **localStorage** for persistence between refresh

```
const API_KEY = process.env.REACT_APP_API_KEY;
```

API Integration

- **I** used a **fetch function** to make API calls within the RecipeContext provider.
- And I stored my API key in a **.env file** for security and accessed via: **process.env.REACT_APP_API_KEY**

```
//if a search term is provided, add it to the url
if (query) {
    url += `&query=${query}`;
}
```

Dynamic Querying

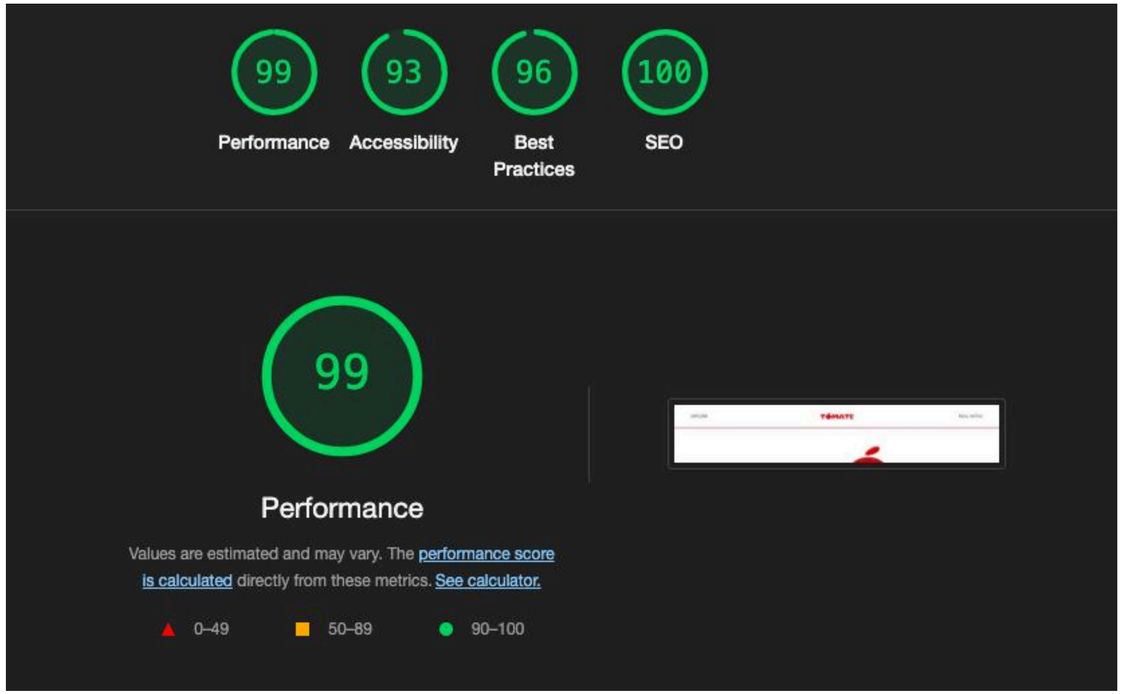
- Users can search recipes by name and category using a search bar, as those inputs are passed as query parameters to the API endpoint.

```
  //store the fetched recipes in local storage so they persist after refresh
  localStorage.setItem("recipes", JSON.stringify(data.results || []));
catch (error){
  console.error("Error fetching recipes:", error);
```

Loading and Error Handling

- I have a **loading state** (loading) in place to show feedback during data fetch.
- Also **error handling** is in place to catch and log failed API requests through console.

Lighthouse Report: Performance



Performance Optimisation

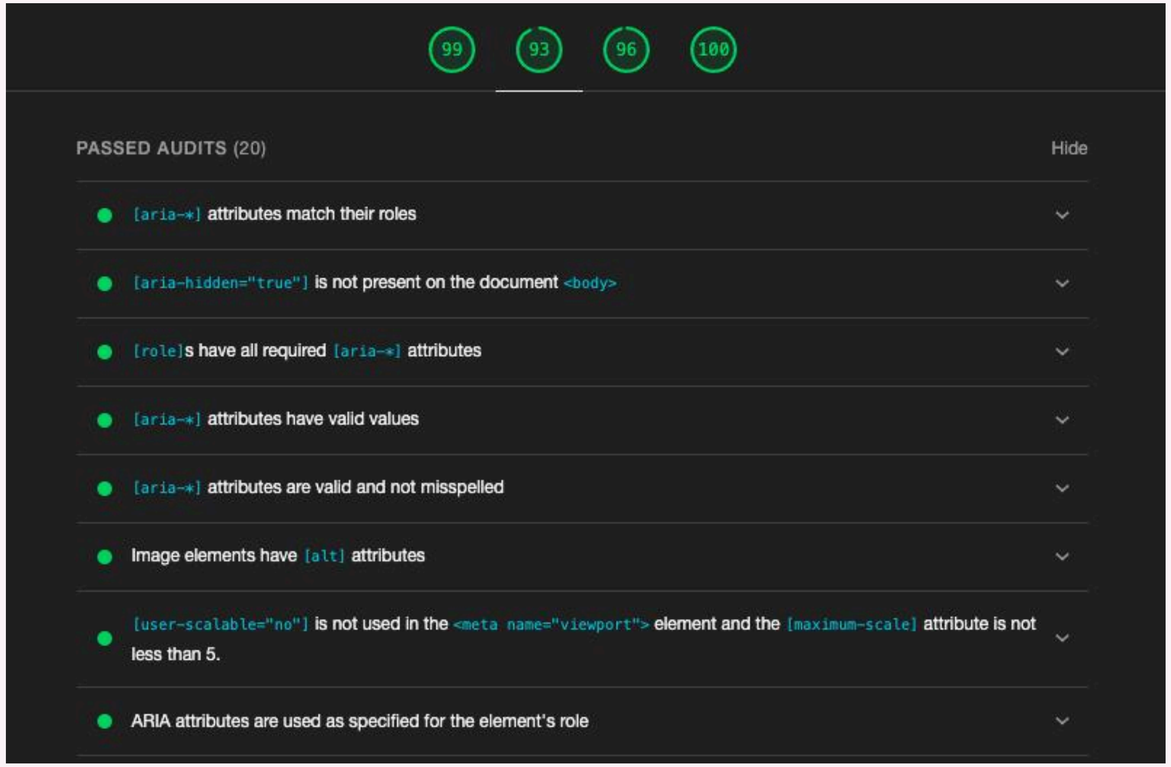
Mobile-first design: Ensured fast rendering and responsive interactions across devices.

Semantic HTML tags for all pages, for better accessibility and SEO.

Optimised Image alt tags for better indexing.

Context API: distributes data efficiently, avoiding unnecessary renders.

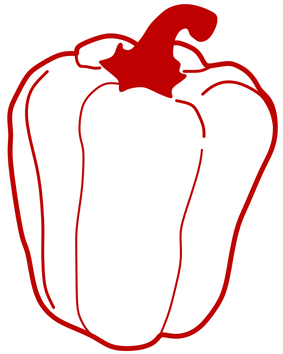
Lighthouse Report: Accessibility



Accessibility

Semantic HTML and **ARIA labels** labels for all pages, ensuring screen reader support and improved accessibility for keyboard users.

Interactive (scroll) and keyboard (buttons) navigation for carousels.



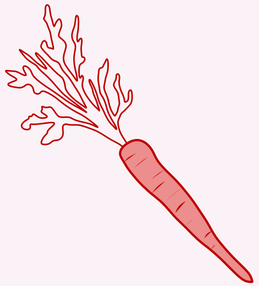
Challenges + Solutions

1 **Challenge:** Using React, SASS and Context API for the first time. Understanding how to make the most of components, structuring them to be reused.

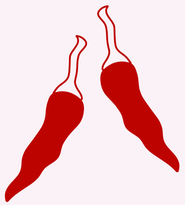
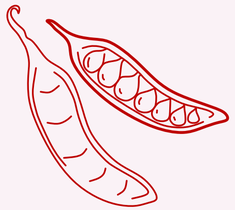
Solution: Researching standard file structuring order, breaking UI into modular components for clarity. Consistent testing and debugging.

2 **Challenge:** The Spoonacular API imposes a daily request limit, which initially created challenges in developing and testing the recipe-specific pages.

Solution: I implemented local storage to cache fetched recipe data. This reduced the number of repeated API calls, improved performance, and ensured data persistence across sessions.





Live Demo

Thank you!