Project Description:

The "Dish Delights" web application was conceived to address the culinary challenges faced by international students and busy professionals. As newcomers to foreign countries, we discovered that cooking requires creativity and effort. Hence, we envisioned a cooking website that would allow users to search for recipes based on available ingredients and desired cuisine types. Users can follow step by step instructions to prepare delicious meals. Additionally, we aspire for the platform not only to aggregate existing recipes but also to serve as a platform for users to create and share their own culinary creations.

Persona Analysis:

1. Persona: Mia, the International Student:
   Age: 22
   Sexual Identification/Orientation: Female, straight
   Culture/Language: Originally from China, speaks Mandarin and English
   Familiarity with Technology: High; frequently uses apps and websites for daily needs
   Contextual Factors:
     Being in a hurry: Often searches for quick recipes between study sessions
     Poor data connection: Uses the website in her dorm where WiFi can be unreliable
     Usage Context: Needs to prepare meals in a student kitchen with limited resources
   Preferred Website Usage: Searches for recipes based on the ingredients already in her fridge, appreciates easy to follow steps due to her moderate cooking skills.

2. Persona: John, the TechSavvy Professional:
   Age: 35
   Sexual Identification/Orientation: Male, gay
   Culture/Language: American, speaks English
   Familiarity with Technology: Very high; integrates technology into all aspects of life
   Disabilities: None permanent but occasionally experiences wrist pain from extensive computer use, prefers voice navigation during those times
   Contextual Factors:
     Professional setting: Often plans meals around his work schedule, looking for quick yet healthy options
     Urgency: Not urgent; plans his meals ahead of time and shops accordingly
     Usage Context: Uses the app to find new recipes that fit his dietary preferences and nutritional goals
   Preferred Website Usage: Prefers using advanced search filters to find recipes that can be made within 30 minutes or less, values high quality images and detailed nutritional information.

Persona analysis for Mia and John

Mia	International student	John	Engineer
Age	22		28
Sexual Orientation	Female, straight		Male, gay
Culture/Language	Originally from China, speaks Mandarin and English		American, speaks English
Familiarity with Technology	High; frequently uses apps and websites for daily needs		integrates technology into all aspects of life
Contextual Factors	Often in a hurry, searches for quick recipes		Prefer deciding what to eat and buy materials in advance
Usage Context	Prepares meals in a kitchen with limited resources		Often plans meals around his work schedule
Preferred Website Usage	Searches recipes based on available ingredients; appreciates easy-to-follow steps		Prefers using advanced search filters to find recipes that can be made within 30 minutes or less

User Journey Map:

User1: Mia, a 22yearold international student with a hectic academic schedule, needing to prepare a meal quickly.
Environment: Mia has just returned home after a long day of classes and needs to prepare dinner with

whatever ingredients she has in her fridge.

1. Optimized Web Layout for Tablets:
Opportunity: Mia prefers using her iPad for browsing recipes. A responsive layout that adapts to different screen sizes ensures that the content is appropriately scaled, enhancing readability and interaction on tablets.

2. Simplified Navigation During Cooking:
Opportunity: Recognizing that Mia's hands may be messy while cooking, the design includes an enlarged step view and simple leftright screen taps for navigation between steps, making it easier to use the website without precise touch controls.
Actions:
Chooses a recipe and reviews the step by step instructions.
Begins cooking, using her phone beside her to navigate the steps.
Opportunities:

User2: John, a 35yearsold professional who recently developed an interest in cooking to maintain a healthy lifestyle. He enjoys trying new recipes that align with his dietary preferences.

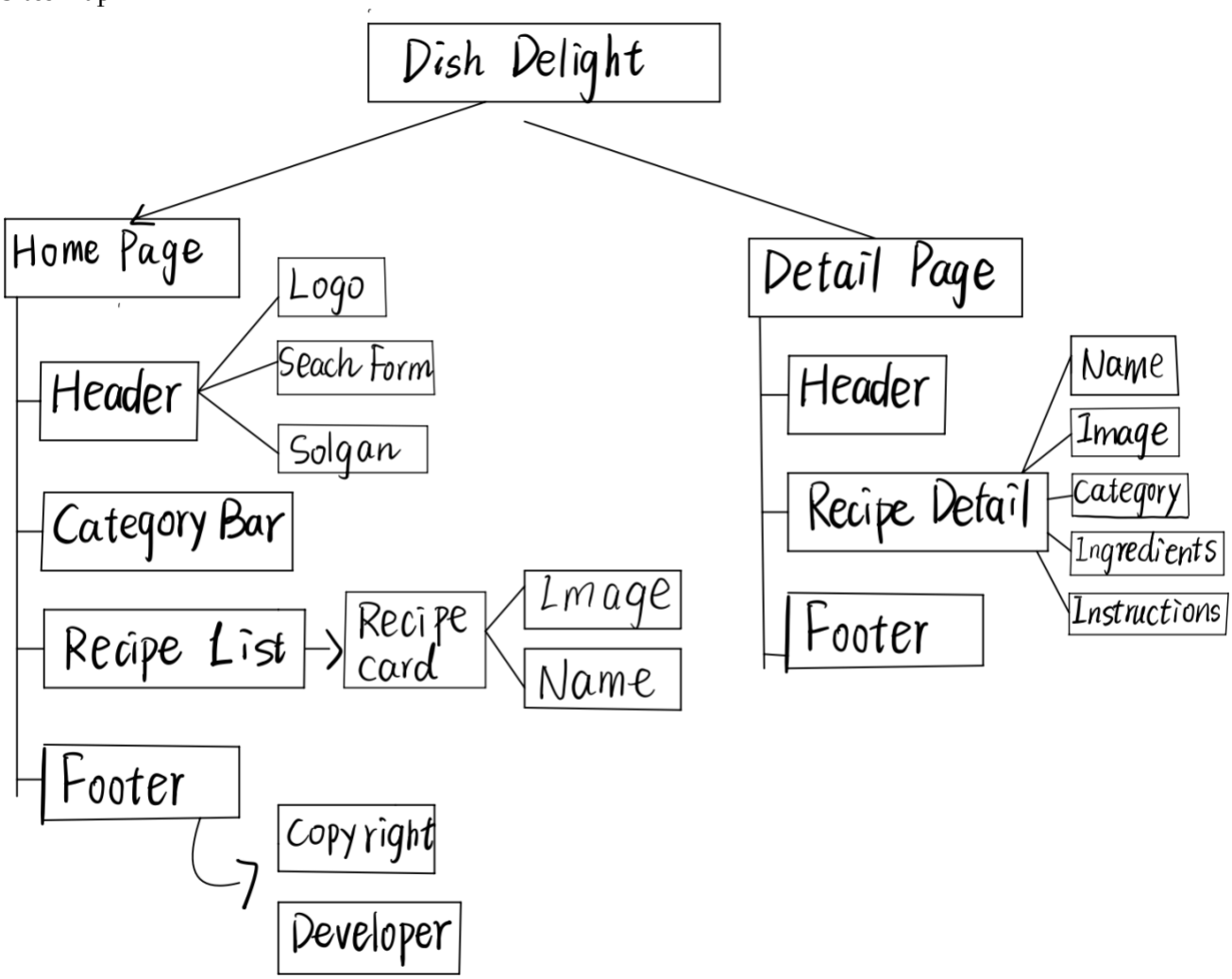
Environment: John plans his weekly meals every Sunday evening to prepare for the upcoming week, using his tablet to browse and select recipes.

1. Exploring Recipes:
Opportunity: Since John is often unsure about which recipes to try, implementing a feature on the homepage that displays random recipes could inspire his meal choices and introduce variety into his diet.

2. Shopping for Ingredients:
Opportunity: To streamline John's shopping experience, develop a checkbox feature for ingredients within each recipe. This would allow him to mark off items he already has or has purchased, facilitating a more organized shopping experience. This feature is planned for future implementation due to current time constraints.

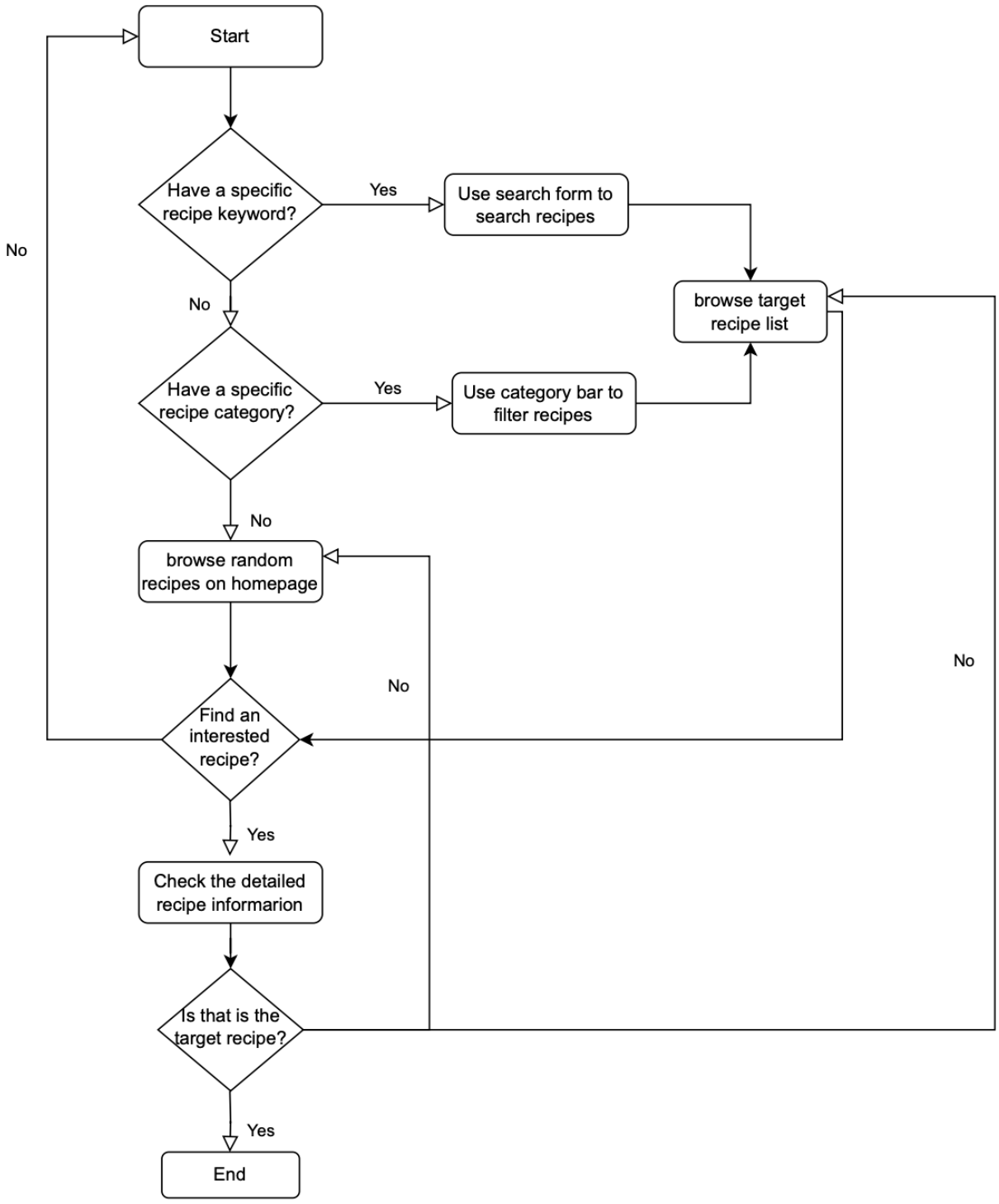
Websites Map:

1. Sites map



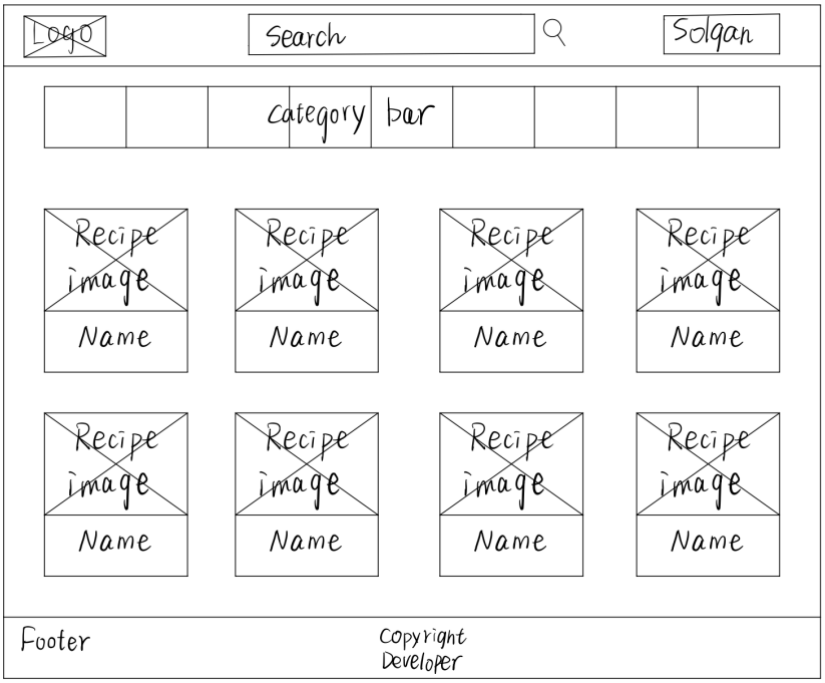
2. Box diagram

For finding a recipe:

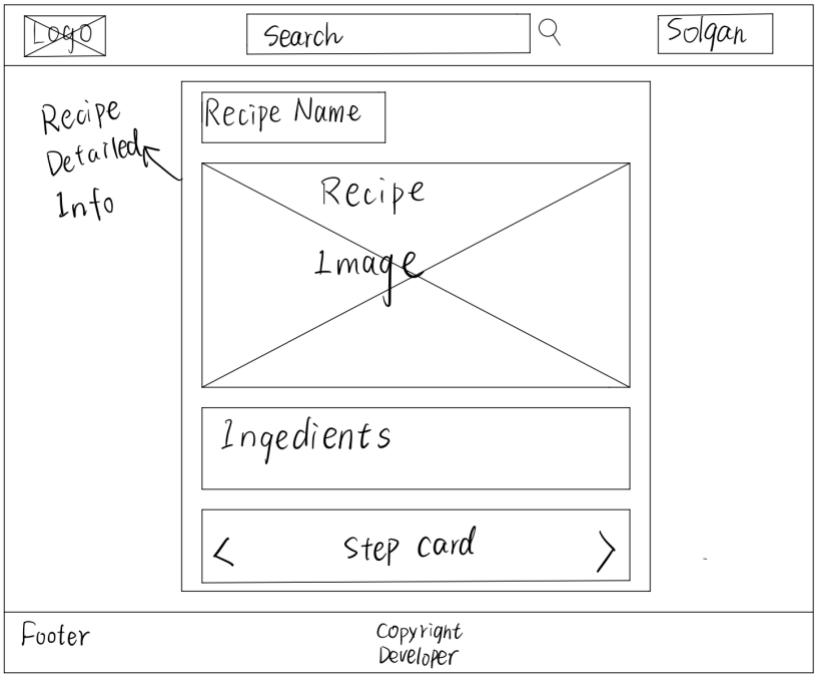

Start
Have a specific recipe keyword?
Yes
Use search form to search recipes
No
browse target recipe list
No
Have a specific recipe category?
Yes
Use category bar to filter recipes
No
browse random recipes on homepage
No
No
Find an interested recipe?
Yes
Check the detailed recipe informarion
Is that is the target recipe?
Yes
End

LowResolution Mockups:

Mockup 1: Homepage layout with search bar, logo, slogan, recipe categories, and recommended recipes.



Mockup 2: Recipe details page layout with recipe name, image, required ingredients, and step by step instructions.



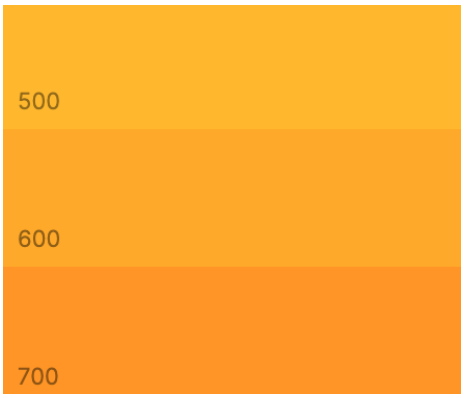
Typography:

1. Choice: Nunito

2. Reasons:
   - Readability: Nunito features rounded characters and even weighting, which ensures good readability even at smaller sizes, making it suitable for prolonged reading.
   - Modern Appearance: The font has a smooth, modern design, making it especially suitable for contemporary and innovative website designs.
   - Broad Compatibility: As part of Google Fonts, Nunito is easy to integrate into various web platforms and maintains good compatibility across multiple devices and browsers.

Theme color:

1. Choice:
   - Amber-500
   - Amber-600
   - Amber-700



2. Reasons:
   - Warm and Inviting: The amber color palette creates a warm and comfortable feeling, which can attract users to explore more recipes. This shade stimulates appetite and is associated with food and cooking.
   - Friendly and Welcoming: Warm tones like amber make the website appear more friendly and welcoming, which helps create a positive user experience and encourages visitors to spend more time on the site.

App's code:

Github repository link: https://github.com/Tingyi-Ruan11/DishDelights.git

Working websites:

https://dish-delights.netlify.app/

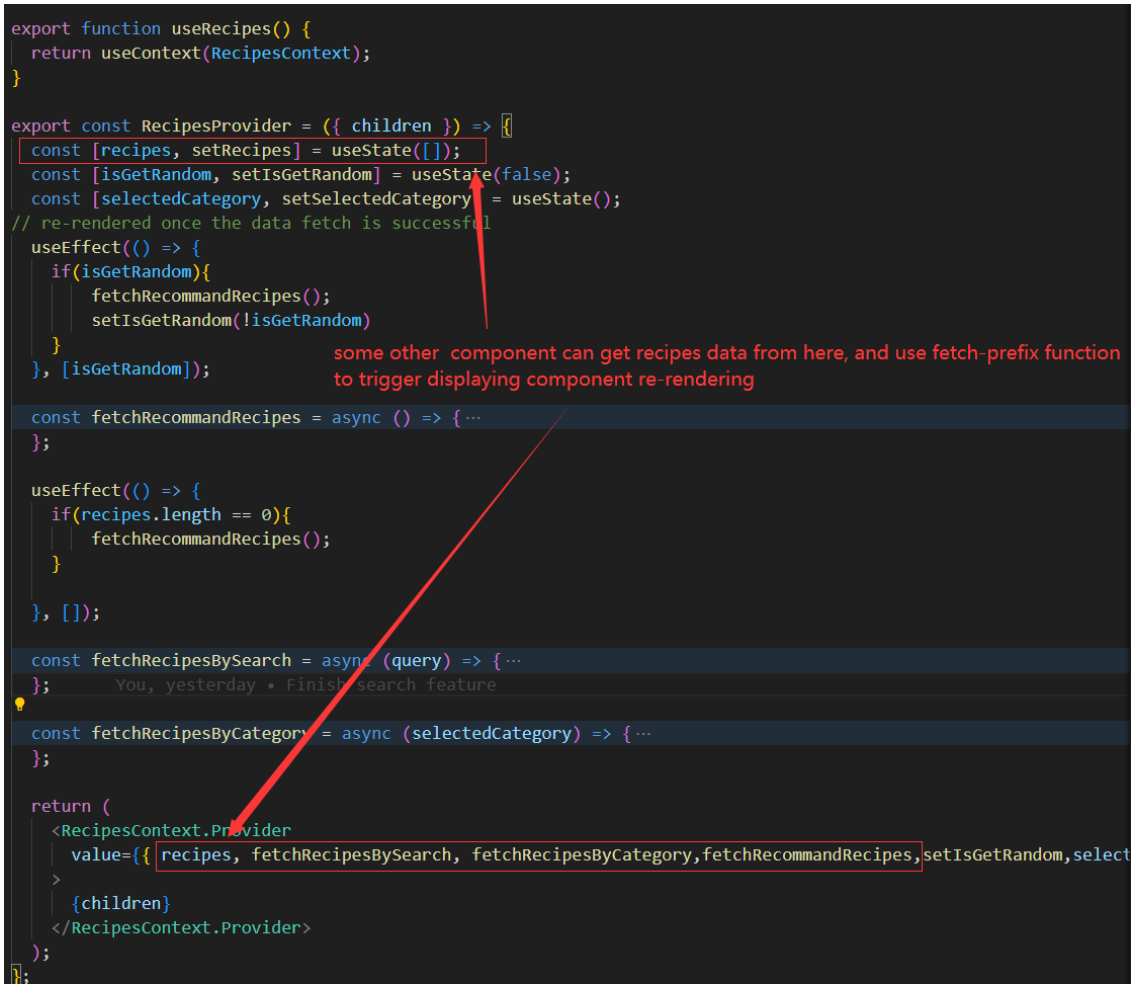
JavaScript Mastery:

0. Techniques:

Custom useContext for state management.
Utilization of getStaticProps and getStaticPaths for efficient data rendering.
Parallel data requests using Promise.all.
Asynchronous Data Rendering with useEffect
Combination of <Link> Component and React Router
Simplified Routing with Next.js File-Based Routing
Responsive Images with Next.js <Image>

1. Custom useContext with recipeContext

The recipeContext is a custom context created using React's useContext to facilitate component communication. In the main page component recipe-list, the recipe data is stored and managed with useState within recipeContext and passed by reference to other components. This ensures that any component wrapped with recipeContext can trigger events and invoke fetch-prefixed methods to update the content displayed in the recipe-list component. These methods include:
fetchRecommendedRecipes
fetchRecipesBySearch
fetchRecipesByCategory
The recipeContext retains the current state of recipes data within the recipe-list component, maintaining the user's last filtered view without re-triggering API requests upon returning to the main page. It also preserves the selected state of CategoryBox in the Category component, remembering the user's last category selection.



“isGetRandom” is for when clicking Logo to back homepage, checking if it’s need to re-render recipes.

```
function HomePage(props) {

  const { recipes,fetchRecommandRecipes } = useRecipes();

  return (
    <div>
      <Head>
        <title>Dish Delights</title>
        <meta
          name="description"
          content="Find a lot of great Dish that allow you to evolve..."
        />
      </Head>
      <Categories onResponseData={handleRecipesDataCategoriesUpdate} />
      <Container>
        {/* <RecipeList items={props.recipes}/> */}
        <div>{recipes ? <RecipeList items={recipes} /> : <p>Loading...</p>}</div>

        <RecipeStepCard />
      </Container>
    </div>
  );
}       You, 2 months ago • Initial commit from Create Next App
```

2. getStaticProps and getStaticPaths in Recipe Details Page

The recipe detail page leverages Next.js's Static Site Generation (SSG) capabilities through getStaticProps and getStaticPaths. Since the content of a specific recipe's detail page rarely changes, caching static pages in advance speeds up response times. It also optimizes SEO by making meaningful data available to search engines instead of serving an empty React framework HTML page.

```
export async function getStaticProps(context) {
  const idMeal = context.params.idMeal;
  const recipe = await getRecipeById(idMeal);

  return {
    props: {
      selectedRecipe: recipe,
    },
    revalidate: 30,
  };
}

export async function getStaticPaths() {
  const recipes = await getFeaturedRecipes();

  const paths = recipes.map((recipe) => ({
    params: { idMeal: recipe.idMeal },
  }));
  return {
    paths: paths,
    fallback: "blocking",
  };
}
```

In the recipe detail page, use next.js provided SSG feature.

3. Parallel Data Requests using Promise.all

Due to the integrated API's lack of a fuzzy search interface, the application attempts to use the search input as parameters for Name, Category, Category and Id, and then merges and deduplicates the arrays returned from these requests. The application executes parallel requests for the search input content, reducing page load times with Promise.all([fetchByName, fetchById, fetchByCategory, fetchByIngredient]);.

```
export async function getRecipeBySearch(searchQuery) {      You, yesterday • Finish search feature
  const fetchByName = getRecipeByName(searchQuery);
  const fetchById = getRecipeById(searchQuery);
  const fetchByCategory = getRecipesByCategory(searchQuery);
  const fetchByIngredient = getRecipeByIngredient(searchQuery);

  try {
      // executes parallel requests
      const results = await Promise.all([fetchByName, fetchById, fetchByCategory, fetchByIngredient]);

      // merging
      const mergedResults = results.flat().filter(item => item != null);
      console.log("flatArray",mergedResults)

      // deduplication
      const uniqueRecipes = mergedResults.reduce((acc, current) => {
          if (!acc.map(recipe => recipe.idMeal).includes(current.idMeal)) {
              acc.push(current);
          }
          return acc;
      }, []);

      return uniqueRecipes;
```

4. Asynchronous Data Rendering with useEffect

To address the asynchronous nature of API requests, the useEffect hook ensures that components initially rendered without data are re-rendered once the data fetch is successful. This re-rendering occurs only once per data retrieval.

```
useEffect(() => {      You, yesterday • Finish s
    if(recipes.length == 0){
        fetchRecommandRecipes();
    }

}, []);
```

5. Combination of <Link> Component and React Router

The <Link> component from Next.js is used to prevent the default browser request behavior and render components based on the React router configuration. A custom <Button> component is wrapped with <Link> to provide reusable routing functionality upon button clicks.

```
const  Logo= () => {
    const { setIsGetRandom } = useRecipes();
    const handleClick = () => {
      setIsGetRandom(true);
     };

    return (
      <div className="flex flex-row items-center justify-between
      <Image href= "/" alt="Logo" className="hidden md:block cur
      <Link href="/" className="font-bold">Dish Delights</Link>
      </div>
    );
}
```

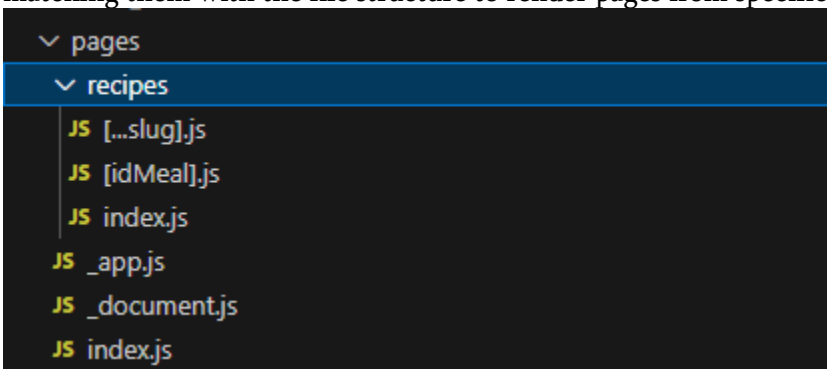
in Logo components, we can click it and re-render Homepage contents, and trigger random recommendation.

```
const Search = () => {

  const [inputValue, setInputValue] = useState('');
  const { fetchRecipesBySearch } = useRecipes();
  const router = useRouter();

   const handleChange = (event) => {
    setInputValue(event.target.value);
  };

        You, 23 hours ago • update css
  const handleSearch = () => {
    fetchRecipesBySearch(inputValue);
    router.push('/');
  };

  const handleKeyPress = (event) => {
    if (event.key === 'Enter') {
      handleSearch();
    }
  };
```

In search components, we have to handle “Enter” input searching, so we use react-router directly.

6. Simplified Routing with Next.js File-Based Routing

Next.js's file-based routing feature simplifies React router configuration by parsing URLs and matching them with the file structure to render pages from specific folders.



7. Responsive Images with Next.js <Image>

The <Image> component from Next.js is utilized to address image compression and responsive layout requirements, automatically cropping images based on the center for different screen sizes.

```jsx
function RecipeItem(props) {
  const data = props.data;
      You, 2 days ago • fix data bug
  const exploreLink = `/recipes/${data.idMeal}`;

  return (
    <li className={classes.item}>
      <Image src={data.strMealThumb} alt={data.strMeal} width={250} height={160} />
      <div className={classes.content}>
        <div className={classes.summary}>
          <h2 className="font-bold ">{data.strMeal}</h2>
        </div>
        <div className={classes.actions}>
          <Button link={exploreLink}>
            <span>Explore</span>
            <span className={classes.icon}>
              <ArrowRightIcon />
            </span>
          </Button>
        </div>
      </div>
    </li>
  );
}
```

ReciptItem is a card-like component, can show recipe image and basic info.

API Integration:

Search Functionality API

Endpoint Used:

By name: www.themealdb.com/api/json/v1/1/search.php?s=Arrabiata

By ID: www.themealdb.com/api/json/v1/1/lookup.php?i=52772

By area or ingredient: www.themealdb.com/api/json/v1/1/filter.php?a=Canadian or www.themealdb.com/api/json/v1/1/filter.php?i=chicken_breast

Purpose:

The Search API endpoint is used for implementing the search functionality within the application, allowing users to find specific recipes by name or ID.

Category Filtering API

Endpoint Used:

By category: www.themealdb.com/api/json/v1/1/filter.php?c=Seafood

Purpose:

This API endpoint is used for filtering recipes according to the category, area, or ingredients, which aids users in displaying recipes belonging to selected criteria.

Random Recipe Selection API

Endpoint Used:

www.themealdb.com/api/json/v1/1/randomselection.php

Purpose:

The random selection API fetches 10 random recipes, intended to provide homepage recommendations for users uncertain about what to cook.

Categories Listing API

Endpoint Used:

www.themealdb.com/api/json/v1/1/categories.php

Purpose:

Used to display all recipe categories for quick user filtering and selection.

Recipe Detail API

Endpoint Used:

www.themealdb.com/api/json/v1/1/lookup.php with the recipe ID as a parameter.

Purpose:

This endpoint is crucial for displaying detailed content upon clicking a recipe card.

Functional Goals

The homepage recipe recommendation aims to suggest options to users unsure of their culinary choices. Displaying all categories for quick filtering assists users with specific preferences to swiftly find recipes. The search functionality is designed to help users quickly locate particular dishes, and the detailed recipe display function presents comprehensive content to the user.

API Considered But Not Used

Ingredients Listing API:

Endpoint:

www.themealdb.com/api/json/v1/1/list.php?i=list

Reason for Exclusion:

While this endpoint provides a list of all recipe ingredients, it only offers names and descriptions without images. The extensive variety makes it less suitable for categorical selection like the categories feature. Moreover, if users need to find recipes containing specific ingredients, the search function already serves this need, making this API superfluous for the project.

The APIs selected and integrated into this culinary project have been carefully chosen to align with the application's primary functions: recommending, filtering, searching, and presenting detailed recipe information. These chosen endpoints ensure an enhanced user experience while keeping the interface and functionality streamlined and efficient.

SEO mastery and Accessibility mastery:

SEO

These checks ensure that your page is following basic search engine optimization advice. There are many additional factors Lighthouse does not score here that may affect your search ranking, including performance on Core Web Vitals. Learn more about Google Search Essentials.

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.

Future of the Project and Team's Learnings:

- The team plans to continue developing the Dish Delights platform, adding the user login/register module, social community and features for user generated content.
- Throughout the project, the team learned the importance of user centered design, responsive layouts, and effective communication in collaborative development efforts.