**CHAPTER 4**

# **UI Pages**

## **4.1 Main Page:**

The main page serves as a visually engaging introduction, capturing the essence of the platform.

Users are greeted with enticing visuals and a clear call-to-action, encouraging them to explore the site.

### **4.1.1. Feature Explanation**

This documentation provides an explanation of the code for the "Deal Checker" feature. The code is written in JavaScript and is used to create the homepage of the Deal Checker application.

### **4.1.2. Importing Required Modules**

The code begins with ​importing the necessary modules, Image and Link, from the next/image and next/link libraries respectively. These modules are used to display images and create links in the application.

### **4.1.3. Home Component**

The Home component is the main component of the homepage. It renders the content and layout of the Deal Checker application.

### **4.1.4. Homepage Layout**

The homepage layout is defined using HTML-like JSX syntax. The layout is divided into different sections to display various elements such as navigation links, a header, a description, and buttons.

### **4.1.5. Navigation Links**

The navigation links section contains a link to the source code of the application on GitHub. Clicking on the link will open the GitHub page in a new tab.

### **4.1.6. Header**

The header section displays the title of the application, "Deal Checker". It also includes a "PRO" label to indicate the pro version of the application.

### **4.1.7. Description**

The description section provides a brief description of the Deal Checker application. It explains that the application is an online shopping assistant that helps users find the best deals.

### **4.1.8. Buttons**

The buttons section includes a "Get Started" button that redirects users to the search page of the application. Clicking on the button will navigate to the specified URL.

### **4.1.9. Additional Links**

The additional links section contains two links: "Docs" and "API Example". Clicking on these links will open the specified URLs in new tabs. The "Docs" link provides in-depth information about the Deal Checker features and API, while the "API Example" link demonstrates how to use the API and mentions that the API returns data in JSON format.

This concludes the documentation for the code. The provided code creates the homepage layout for the Deal Checker application, including navigation links, a header, a description, and buttons. It also includes additional links to the documentation and an API example.

## **4.2 Search Page:**

The search page is designed with user convenience in mind, featuring an intuitive search bar that triggers real-time results.

### **4.2.1. UI Elements**

The component's UI consists of the following elements:

### **4.2.2. Heading and Subheading**

The component displays a heading and a subheading at the top of the page. The heading highlights the website's mission, while the subheading provides additional information about the website's focus.

### **4.2.3. Search Input**

The component includes a search input field where users can enter their search queries. As   
users type, the search state variable is updated accordingly. Users can also press the "Enter"   
key to submit their search.

### **4.2.4. Search Button**

Next to the search input field, there is a search button. Clicking on this button triggers the handelSubmit function, which redirects the user to the search results page based on their search query.

### **4.2.5. Description**

Below the search input field and button, there is a description section that provides an overview of the website's purpose and benefits. It emphasizes the website's ability to help users find the best deals by comparing prices and reviews from various websites.

### **4.2.6. Divider**

A horizontal divider is displayed after the description section to visually separate the content.

### **4.2.7. Additional Description**

Following the divider, there is an additional description section that further explains the benefits of using Deal Checker for online shopping. It highlights the ability to filter search results and view ratings and reviews from other customers.

### **4.2.8. Example Usage**

Here is an example of how you can use this component in your React application:

In this example, the DealCheckerPage component is ​imported and rendered within the App component. You can customize the surrounding elements and styling based on your application's needs.

## **4.3 Item List Page:**

The item list page strikes a balance between information and simplicity, presenting users with a concise overview of available products.

Each product card includes essential details, inviting users to explore further.

### **4.3.1 State Variables**

The component uses the useState hook to define two state variables: data and Loading.

* data is an array that will store the fetched data from the API.
* Loading is a boolean variable that indicates whether the data is currently being loaded.

### **4.3.2. useEffect Hook**

The component uses the useEffect hook to fetch data from the API when the search parameter changes.

* The useEffect hook is called whenever the search parameter changes.
* Inside the useEffect hook, an asynchronous function is defined to fetch data from the API using the axios library.
* The fetched data is then stored in the data state variable using the setData function.
* Finally, the Loading state variable is set to false to indicate that the data has finished loading.

### **4.3.3. Conditional Rendering**

The component uses conditional rendering to display either a loading skeleton or the fetched data based on the value of the Loading state variable.

* If Loading is true, a loading skeleton component is rendered to indicate that the data is being loaded.
* If Loading is false, the fetched data is rendered using the Card component.

### **4.3.4. Rendering Individual Items**

Inside the conditional rendering block, the fetched data is mapped over to render individual items using the Card component.

* The map function is used to iterate over the data array.
* For each item in the data array, an instance of the Card component is rendered with the corresponding item data passed as props.

## **4.4 Full Details Page:**

The full details page is the heart of the platform, offering a deep dive into a selected product.

Price and feature comparisons are presented in a visually appealing format, aiding users in making informed choices.

### **4.4.1. Example**

Here's an example of how you can use this code in a React application:

In this example, the page component is rendered with a params prop that specifies the comparison value as "example". This will trigger the component to fetch data from the API endpoint and render the comparison page.

## **4.5 About Page:**

The "About" page is a personal touch, providing users with insights into the individuals behind the "Deal Checker" project.

It features information about you, the creator, showcasing your background, motivations, and the journey of developing the platform.

### **4.5.1 Creating the Profile Page**

The profile page is created as a functional component named page. It returns JSX code that represents the structure and content of the profile page.

const page = () => {

return (

// JSX code representing the profile page

);

};

### **4.5.2 Structure and Styling**

The profile page is structured using HTML elements and CSS classes. The className attribute is used to apply the specified CSS classes.

// Content for the left side of the profile page

// Content for the right side of the profile page

The flex class is used to create a flex container, allowing the left and right sides of the profile page to be displayed side by side on larger screens (md:flex-row). On smaller screens, they are displayed in a column (flex-col). The items-center and justify-center classes are used to center the content vertically and horizontally.