

CHAPTER 1

Introduction

1.1. Background Of Study

A price comparison website acts as a platform or medium between the consumers and the sellers. It allows consumers to see different lists of prices for the product chosen by user and it helps consumers to make an informed decision about which to choose in order to save money. It also acts as a tool to help consumers increase their price consciousness so that they will not feel cheated by the advertisement from the retailers that claimed they are offering the cheapest price but the reality happened to be otherwise.

Unlike other comparison sites, <https://deal-checker.vercel.app/> (the name of this project) will focus on providing list of prices of home groceries products such as onion, chilies, garlic, potatoes, fish, chicken and others. Due to vast increase of people who are online, <https://deal-checker.vercel.app/> will be a great help for those who are stuck with loads of work in the office and don't have much time to check on the current price of the home groceries products. According to research of Social, Digital and Mobile in Malaysia made by We Are Social, the internet penetration for Malaysia is 59% and the average hours Malaysian netizens spend using the internet every week is 19.8 hours. Meanwhile 21% of Malaysian internet users access the web via mobile devices which means they have internet accessibility anywhere with their smart phones. 77% of Malaysian web users have shared their thoughts on a brand via social media during this research was made in the year 2011. The research shows that how Malaysian people are attached to the internet.

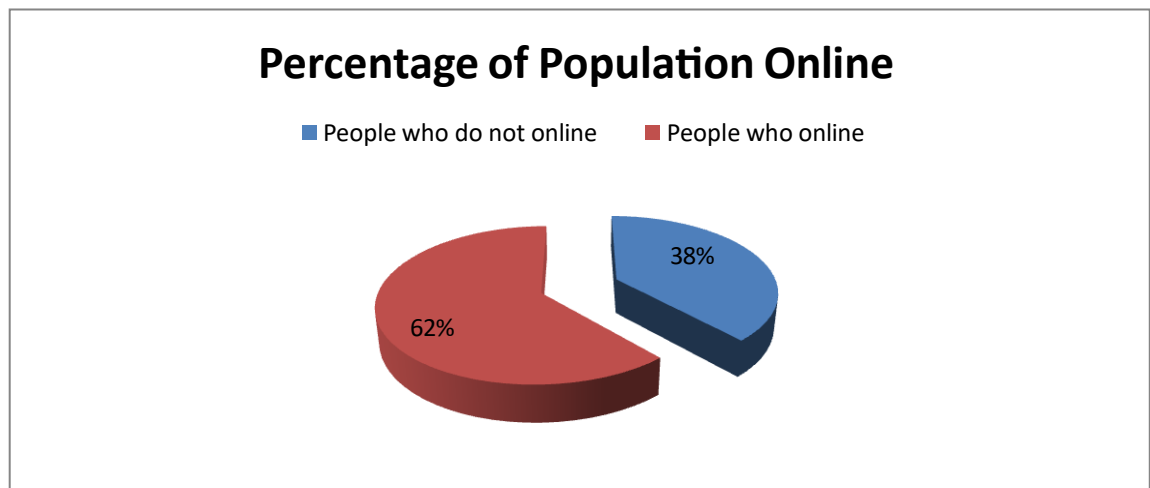


Figure 1: Percentage of Malaysian Population Online
(Source: Malaysiacrunch.com, 2011)

Based on the figure above, the statistic is made according to the year 2011, it has clearly shown that more than half of the population in Malaysia is using internet every day and it is believed that the number is increasing from time to time. A survey has been made by Nielsen towards internet users and the results shows that most of the Malaysian internet users spend 20 hours a week online in average. 53% of the respondents go online everyday meanwhile 35% go online on weekly basis. 63% use internet for information, and 94%, which is majority of the internet users, use internet as shopping guideline.

Based on the research made by FRS (Financial Reporting Standards) in the year 2009, it clearly shows that the usage of price comparison website divided into two types of users; those who really searching for the best deal possible and the other one is consumers that are simply looking for a convenient and time saving way to get a quote. The 'Modern Sophisticate' are serial switchers and more likely to use several comparison websites for research meanwhile the 'Convenience Seekers' are more likely to be loyal to single price comparison site. More people are using price comparison website as their reference to check on the price compared to the users that trusting the website. Please refer the figure below.

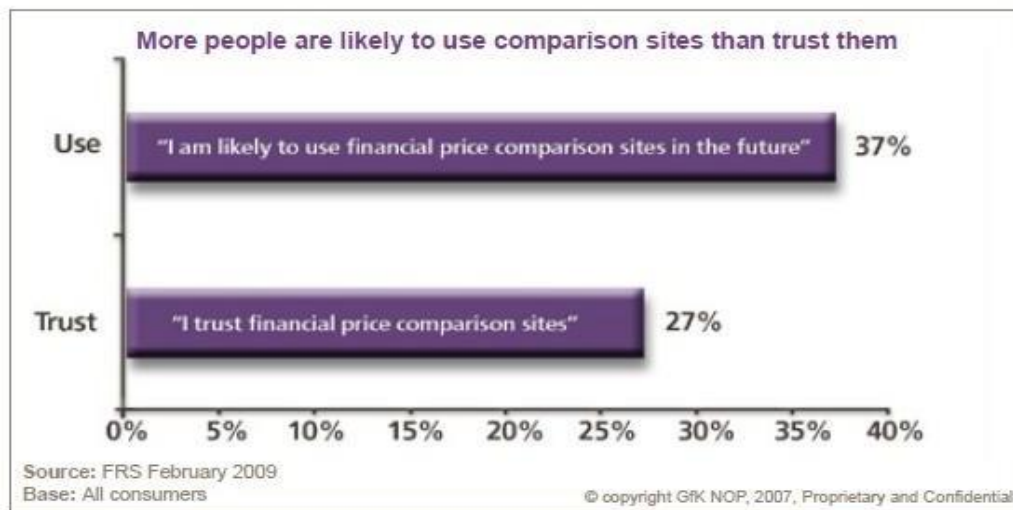


Figure 2: More People Are Likely to Use Comparison Sites than Trust Them
(Source: FRS – February, 2009)

As what has been mention earlier, a price comparison website act as the medium between customers and retailers so customers can make purchase online for certain retailers that provide such services. On the other words, the price comparison website also has the role to promote the retailer/shop/hypermarket/supermarket to the customers. The pressures on time and money especially in the current economic situation where the living cost increases and there's only little time to do some shopping for household, a price comparison website like [Https://deal-checker.vercel.app/](https://deal-checker.vercel.app/) will absolutely become a great help towards consumers. Besides, users nowadays are very comfortable with the internet that it has grown a wider variety of applications from networking and now provide various references for the users.

1.2. Problem Statement

Compared to other countries, in Malaysia there is only few prices comparison website that is accessible until now. Most of them is comparing price for hotel's rate, holiday's package, mobile phone and others.

It is important for a web comparison website to return results with the low prices as what the customers want but accurate results also important so that customers can get what they really want. It also depends on how regular the database is being updated otherwise customers will be confused when they compared it from another site.

Most of the working people do not have time to do shopping for their home groceries. As consumers, they have the right to choose which shop is offering the best price for a certain product that they are interested in. However, to check on price offered by each shop is time consuming and due to limited time that they have, they are not able to compare the prices and end up buying certain product with higher price. Typical mindset of customers nowadays, they see Tesco as the providers for the cheapest product but the reality is, not every product in Tesco offered at the cheapest price. Sometimes, the smaller shop in the neighborhood offers cheaper price.

The other problem that occurs to the retailers/seller side, in order for them to promote their products or if there is any promotion going on, usually they will print out pamphlet to distribute it to the customers. It is costly as they have to produce it in lots of copies and if there is any error in the printed pamphlet, they have to make correction on every copy, which is time consuming, so that customers will not confuse with the pricing. Plus, the catalogue or pamphlets given to customers usually end up being thrown away carelessly and it led to pollution. Therefore, by having catalogue that published online the sellers will be able to save cost and support the green campaign too.

Moreover, for the sellers/retailers who are operating with the small shops, they usually don't have the knowledge to blogging and website to promote their products online. As compared to the big supermarket like Giant or Tesco that have their own website, owners of the small shops found it is hard for them to reach out their customers. They do have the basic knowledge about internet but they do not have the skills to have blog/website for their shops. So, this project will be beneficial for them to get people know about them and their products.

1.3. Objectives

The objective of this project is to develop a price comparison website that will have the following functions:

1. To provide customers with a list of price comparison and highlight the

cheapest price.

2. To increase price consciousness among consumers.
3. To ensure that the price database is updated regularly so that customers will be able to get accurate results.
4. To provide service for users to find the product's price

1.4. Scope Of Study

The scope of study for this [Https://deal-checker.vercel.app/](https://deal-checker.vercel.app/) project will be all internet users and it is narrowed down to those who are using internet for business-related purpose, especially in Malaysia. Humanwebsite.com has provided a statistic for the Malaysian online shopping based on the number of internet users and the internet buyers. Refer to the figure below for the comparison of internet users and the internet buyers.

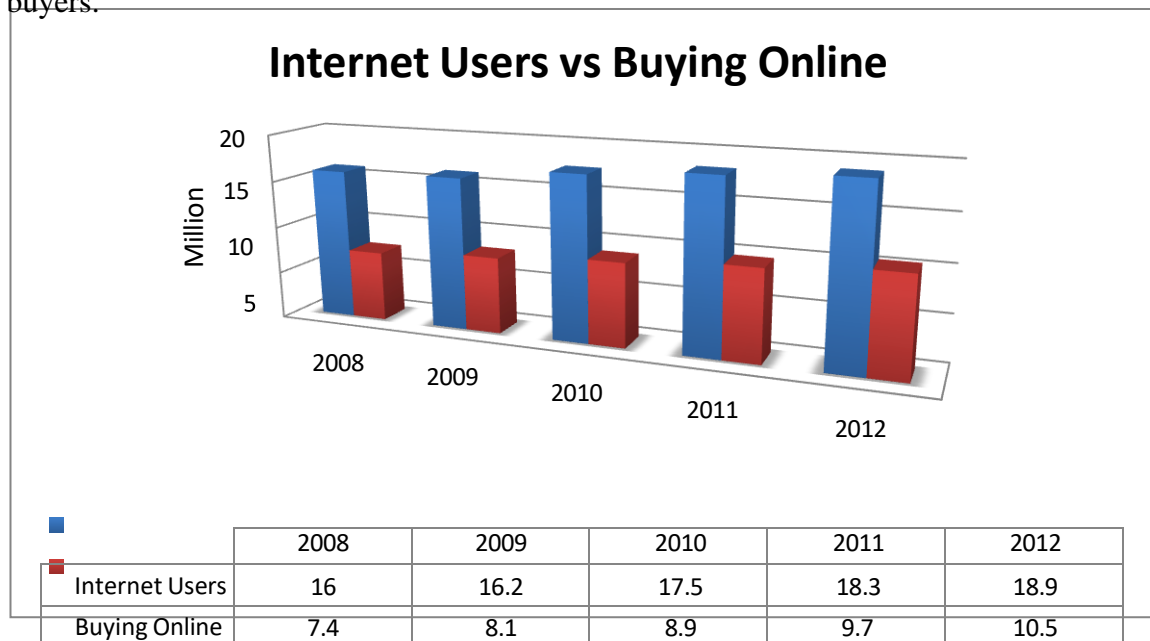


Figure 3: Statistics on Malaysian Online Shopping
(Source: Malaysiacrunch.com, 2012)

This statistic has been published in the local Chinese newspaper, Oriental Daily News. Based on the statistic, it shows that the number of internet users increasing every year as well as the internet buyer which means every single of them is the potential visitor

of <https://deal-checker.vercel.app/>. For a price comparison website like <https://deal-checker.vercel.app/>, it only provides with the useful information to the users and it is up to the users to decide which supermarket or shop they should go.

With internet, a price comparison website is accessible anytime and anywhere. The observation made by the author has resulted that people usually go online when they are on the way back from work especially those who are using public transport. Therefore, visiting <https://deal-checker.vercel.app/> can be one of their choices to fill up their free time. They also can share the link of the promotion through social website such as Facebook and Twitter which has the 90% of the Malaysian internet users according to the research made by, We Are Social in December 2011.

Meanwhile for the sellers/retailers, the target will be focusing more on the business that operating at the shop lot. Based on the interview done by the author with some of the owners of the shops, they admit that it will be useful for them if there is service for them to advertise their products on the web for free because they don't have much time and insufficient skills to maintain website/blog on their own.

1.5. Significance Of The Project

<https://deal-checker.vercel.app/> act as tool to assist consumers make informed decision before purchasing product by providing the list of prices offered by different retailers/supermarket. Users will use this website as their reference to check on the price of groceries products sold and promote if there is any promotion going on. It also able to help sellers to promote new product by sending emails to the subscribers about it.

Instead of taking hours and energy to go to each shop just want to check on the price, <https://deal-checker.vercel.app/> offered better solution by getting all the price and users just need to go online and choose which product they want to know and the list of retailers and the price offered will be shown. Users can check it from anywhere, no matter at home or at work, or even in the train while going back from work, <https://deal-checker.vercel.app/> is accessible anytime as long as there is internet connection.

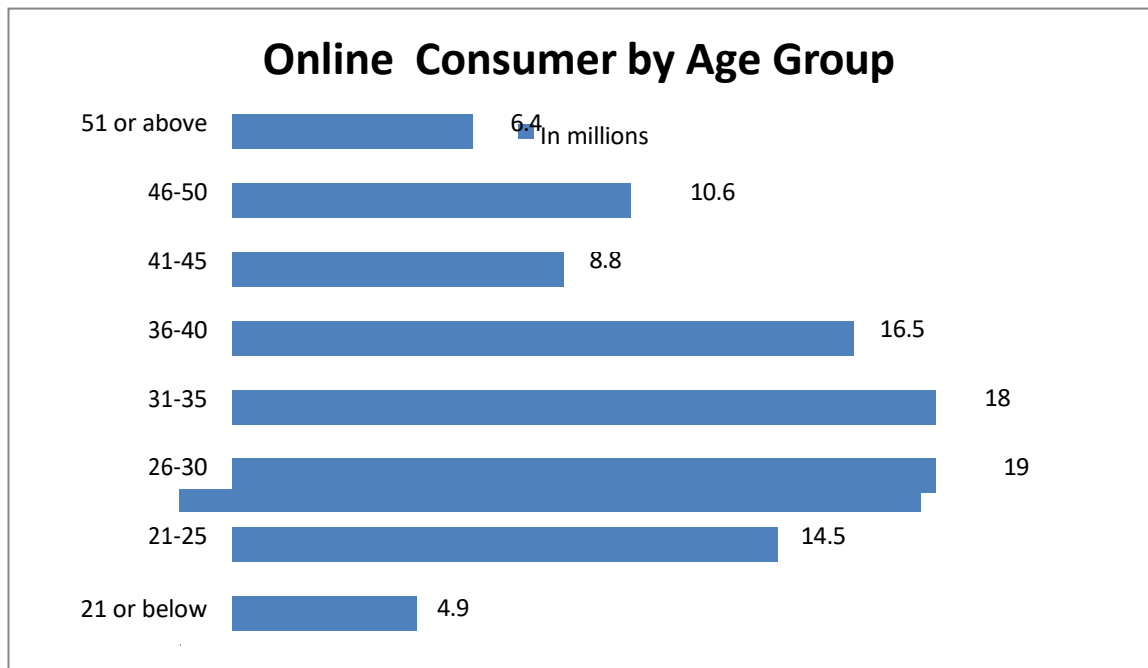


Figure 4: Online Consumer by Age Group
(Source: Malaysiacrunch.com, 2012)

Based on statistics from Malaysia Crunch’s website, most of the consumers who online are contributed by the working adults in the age group of 21 to 40. Meanwhile 60% of online consumers are working as managers or executives and about 13% are students. It shows that most of the working people spend time on the net so a website that provides useful information to help them shopping.

A study was conducted by comScore, co-sponsored by Merchandise Commerce and iProspect, resulted as the following;

- Shoppers usually consulted about four websites for price and feature information (on average)
- Amongst all consumers who are purchasing products offline, roughly two-thirds begin their searches online, using a combination of search website and the retailer’s own web site.
- 70% of shoppers make purchases within one month of their searching from

the internet and one-third of the shoppers buy within one week of starting their searches.

It is clearly shown that online search has clearly become the norm nowadays and most of the customers out there has become more educated and knows how to use the internet. Therefore, it is significance to develop a system that provides information needed for consumers shopping. The development of [Https://deal-checker.vercel.app/](https://deal-checker.vercel.app/) will help consumers to increase their price consciousness, help them making informed decision to save money as well as help the sellers to advertise for free.

1.6. Feasibility Study

Feasibility study is an analysis of the viability idea. The studies provide thorough analysis of the system. The outcome of the feasibility studies will indicate whether can proceed or not to develop the system.

1.6.1 Technical Feasibility

The tools that will use for this system are:

- NEXT js, Axios, Cheerio
- Another language such as tailwind

1.6.2 Operational Feasibility

This feasibility study mainly concerned whether this system will be used if it is developed and implemented. If this system meets the requirements and needs of customers and sellers, it can be proposed to them to be used in the future.

CHAPTER 2

Technologies Used

2.1. Next.js for Dynamic UI:

Description: Next.js, a React framework, is chosen for its ability to create dynamic and server-rendered web applications. Features like server-side rendering (SSR) and efficient client-side routing contribute to a seamless user experience.

2.1.1. Why Next.js:

- **React Framework:** Next.js is a React framework that simplifies the development of React applications by providing a set of conventions and tools.
- **Server-Side Rendering (SSR):** Next.js supports SSR out of the box, enabling faster page loads by rendering pages on the server and sending the fully rendered page to the client.
- **Static Site Generation (SSG):** It also supports SSG, allowing you to pre-render pages at build time, resulting in highly optimized static assets that can be served by a CDN.
- **Automatic Code Splitting:** Next.js automatically splits code into small, optimized chunks, which are loaded only when needed. This helps reduce initial loading times and improves performance.
- **Routing System:** Next.js has a powerful and intuitive routing system that simplifies the creation of dynamic routes and nested routing structures.
- **Zero Configuration:** Getting started with Next.js is easy, thanks to its zero-configuration setup. You can start building a React application without the need for complex configurations.
- **Customizable Head and Document:** Next.js provides a customizable `<Head>` component for managing document head elements and a `_document.js` file for customizing the HTML document.
- **API Routes:** Easily create API routes within your Next.js application, allowing you to handle server-side logic or connect to databases on the server side.

- **Built-in CSS Support:** Next.js comes with built-in support for styling, including CSS, Sass, and CSS-in-JS solutions. It also supports the latest CSS features like CSS modules.
- **Extensibility:** Next.js is highly extensible, allowing developers to add custom webpack configurations, plugins, and middleware to tailor the development environment to specific needs.
- **Community and Ecosystem:** Being a popular and widely adopted framework, Next.js has a thriving community and a rich ecosystem of plugins, libraries, and tools that can enhance the development experience.
- **Continuous Development:** Next.js is actively developed and maintained by the community and the Vercel team, ensuring that it stays up-to-date with the latest React features and best practices.

2.2. Axios for Efficient Data Retrieval:

Description: Axios, a JavaScript library, excels in making HTTP requests. Its simplicity and flexibility make it ideal for fetching data from external APIs.

2.2.1. Why Axios:

- **Promise-Based:** Axios is a promise-based HTTP client, allowing you to take advantage of JavaScript's native promise syntax for handling asynchronous operations.
- **Browser and Node.js Support:** Axios can be used both in the browser and in Node.js environments, providing a consistent API for making HTTP requests across different platforms.
- **Simple API:** Axios provides a simple and clean API for sending HTTP requests. It supports various HTTP methods like GET, POST, PUT, and DELETE, and you can customize requests with headers, params, and data.
- **Request and Response Interceptors:** Axios allows you to use interceptors to globally handle request and response logic. This can be useful for tasks such as modifying headers, logging, or handling errors.
- **Canceling Requests:** Axios provides a built-in mechanism for canceling requests, which can be useful in scenarios where a user navigates away from a page or when a component unmounts.
- **Automatic JSON Data Transformation:** Axios automatically converts request and response data to and from JSON format, simplifying the process of working with JSON APIs.

- **Cross-Origin Resource Sharing (CORS) Handling:** Axios handles CORS automatically, simplifying the process of making requests to different domains.
- **Error Handling:** Axios provides a convenient way to handle HTTP errors by rejecting promises when the status code indicates an error. This makes it easy to implement error handling logic in your applications.
- **Support for Download and Upload Progress:** Axios supports tracking the progress of download and upload operations, which can be useful for displaying progress bars in your applications.
- **Security Features:** Axios includes built-in protection against common security vulnerabilities, such as Cross Site Scripting (XSS) and Cross Site Request Forgery (CSRF).
- **Community Support:** Axios is widely used and has a large community, making it easy to find resources, tutorials, and solutions to common issues.
- **Integration with Promises:** As a promise-based library, Axios integrates seamlessly with other promise-based JavaScript features, making it easy to work with asynchronous code in a consistent manner.

2.3. Cheerio for Web Scraping:

Description: Cheerio, a lightweight HTML parsing library, is essential for web scraping, enabling efficient data extraction from HTML.

2.3.1. Why Cheerio:

- **jQuery-Like API:** Cheerio provides a jQuery-like API for parsing and manipulating HTML and XML documents. Developers familiar with jQuery will find it easy to use and navigate.
- **Server-Side DOM Manipulation:** Cheerio is designed to work on the server side, making it well-suited for tasks like web scraping and parsing HTML in a Node.js environment.
- **Lightweight:** Cheerio is a lightweight library that focuses on providing a simple and efficient way to traverse and manipulate HTML documents. It does not have the overhead of a full browser environment.
- **Selector Support:** Cheerio supports CSS-style selectors, allowing you to easily target and manipulate specific elements in the HTML document. This is similar to how selectors work in jQuery.

- **Fast and Efficient:** Cheerio is known for its speed and efficiency when it comes to parsing and manipulating HTML. It's a performant choice for tasks that involve dealing with large amounts of HTML data.
- **No External Dependencies:** Cheerio has minimal external dependencies, which makes it easy to integrate into Node.js projects without worrying about a large number of additional dependencies.
- **HTML Parsing:** Cheerio excels at parsing HTML strings and turning them into a traversable DOM structure, allowing developers to extract specific information or modify the document as needed.
- **Attribute Manipulation:** Cheerio provides methods for easily manipulating attributes of HTML elements, making it convenient to modify or extract specific data from the document.
- **Modular:** Cheerio is modular, allowing you to use it in combination with other Node.js libraries for tasks such as making HTTP requests (e.g., with Axios) before parsing the HTML.
- **Well-Documented:** Cheerio has comprehensive documentation and a straightforward API, making it easy for developers to get started and find the information they need.
- **Community Support:** Cheerio has a supportive community, and developers often share tips, tricks, and solutions for common use cases related to web scraping and HTML parsing.
- **Customizable Parsing Options:** Cheerio allows you to customize parsing options, providing flexibility for different use cases and handling various HTML structures.

2.4. Tailwind for Stylish Design:

Description: Tailwind CSS, a utility-first framework, offers pre-defined classes for consistent and visually appealing design.

2.4.1. Why Tailwind:

- **Utility-First Approach:** Tailwind CSS follows a utility-first approach, providing a set of low-level utility classes that can be composed to build designs directly in your markup. This approach eliminates the need for writing custom CSS styles for every component.

- **Flexibility:** Tailwind is highly flexible, allowing developers to create unique designs without being restricted by pre-designed components. It provides a wide range of utility classes for styling elements, spacing, typography, and more.
- **Responsive Design:** Tailwind makes it easy to create responsive designs with utility classes that allow you to specify styles based on screen sizes. This helps in building websites that work well on various devices.
- **Customization:** Tailwind is highly customizable, and you can configure it to include only the styles you need for your project. This helps in keeping the final CSS file size minimal.
- **Easy to Learn:** With its utility-first approach, Tailwind is easy to learn, especially for developers familiar with HTML and CSS. The class names are intuitive, making it simple to understand and use.
- **Rapid Prototyping:** Tailwind enables rapid prototyping by allowing developers to quickly build and iterate on designs using utility classes. This is especially beneficial during the early stages of development.
- **Community Plugins:** Tailwind has a thriving community that contributes various plugins, extensions, and tools. These can enhance the functionality and utility of Tailwind for specific use cases.
- **Dark Mode Support:** Tailwind includes built-in support for implementing dark mode in your designs, making it easier to create websites that adapt to user preferences or system settings.
- **PurgeCSS Integration:** Tailwind can be integrated with tools like PurgeCSS to automatically remove unused styles, resulting in smaller CSS file sizes for production, despite the extensive utility classes available.
- **Theming:** Tailwind allows you to define custom themes, making it easy to maintain a consistent design system across your project. You can customize colors, fonts, spacing, and more.
- **Active Development:** Tailwind is actively developed and maintained, ensuring

CHAPTER 3

API Endpoints

3.1. Dynamic API Endpoint:

Description: The dynamic API endpoint (https://deal-checker.vercel.app/api/item/*) dynamically retrieves detailed product information based on user queries.

3.1.1. Why Dynamic API Endpoint:

Flexibility: Accommodates a wide range of product queries for adaptability.

Real-Time Data: Provides up-to-date information for an enhanced user experience.

3.1.2. Feature Explanation:

This documentation provides a comprehensive explanation of the code used to scrape data from a website using Axios and Cheerio in a Next.js application.

3.1.3. Importing Required Modules:

To begin with, the code imports the necessary modules: `NextResponse` from the "next/server" package, `axios`, and `cheerio`. These modules are required for making HTTP requests and parsing HTML respectively.

3.1.4. Setting Up URLs:

The code defines two URL variables: `main_url` and `base_url`. These variables store the main URL and the base URL for the search query on Google Shopping.

3.1.5. Scraper Function:

The `scraper` function is an asynchronous function that takes a URL as an input parameter. This function is responsible for making an HTTP GET request to the specified URL and scraping data from the response.

Within the `scraper` function, the following steps are performed:

1. Make an HTTP GET request to the specified URL using Axios and retrieve the response.

2. Load the response data into a Cheerio instance to parse and manipulate the HTML.
3. Iterate over each element with the class "sh-dgr__content" and extract the desired data using Cheerio selectors.
4. Construct an object with the extracted data and push it to the `data` array.
5. Return the `data` array.

3.1.6. GET Request Handler:

The code exports an async function named `GET` that handles the GET requests to the server. This function takes two parameters: `request` and `{ params }`.

Within the `GET` function, the following steps are performed:

1. Extract the value of the `item` parameter from the `params` object.
2. Construct the search URL by appending the `item` value to the `base_url`.
3. Call the `scraper` function with the constructed URL and await the scraped data.
4. Return the scraped data as a JSON response using `NextResponse.json()`.

Example Usage:

This code can be used to fetch data from a website using Axios and Cheerio in a Next.js application. The scraped data is returned as a JSON response, which can be further processed or displayed as needed.

3.2. Comparison API Endpoint:

Description: The comparison API endpoint (https://deal-checker.vercel.app/api/comparison/*) offers a comprehensive view of product pricing and delivery details across various online retailers.

3.2.1. Why Comparison API Endpoint:

Informed Decision-Making: Enables users to make decisions based on pricing, delivery, and brand reputation.

User Empowerment: Provides the ability to choose the best option based on individual preferences.

3.2.2. Usage

To use the web scraper, you need to import the required modules and call the `scraper` function with the URL of the product page on Google Shopping as the argument. The `scraper` function returns an array of objects containing the scraped data.

Here's an example of how to use the web scraper:

In the example above, the GET function is an API endpoint that receives a request and extracts the comp parameter from the request URL. It then constructs the URL for the product page on Google Shopping and calls the scraper function to retrieve the data. The scraped data is returned as a JSON response using the NextResponse.json method.

3.2.3. Scraper Function

The scraper function is the main function responsible for scraping the data from the provided URL. It uses Axios to make HTTP requests and Cheerio to parse the HTML response.

The function follows the following steps:

1. It initializes the url2 variable and sets the baseUrl to "https://www.google.com".
2. It makes an HTTP GET request to the specified URL using Axios and sets the User-Agent header to mimic a web browser.
3. It loads the HTML response into the Cheerio library.
4. It extracts the URL of the first search result from the loaded HTML using the CSS selector ".EJbZzc".
5. If the extracted URL is "https://www.google.comundefined", it sets url2 to the original URL.
6. It initializes the mainData and content arrays to store the scraped data.
7. It makes another HTTP GET request to the extracted URL using Axios.
8. It loads the HTML response into Cheerio.
9. It extracts the image link, title, review, and review out of the product from the loaded HTML using CSS selectors.
10. It iterates over each offer row in the loaded HTML using the CSS selector ".sh-osd__offer-row".
11. For each offer row, it extracts the company brand, company link, delivery details, item price, and total price using CSS selectors.
12. It creates an object with the extracted data and adds it to the content array.
13. It creates an object with the main data, including the title, review, review out of, image link, and content array, and adds it to the mainData array.
14. It returns the mainData array containing all the scraped data.

3.2.4. Error Handling

The code includes error handling to catch any exceptions that may occur during the scraping process. If an error occurs, it will be logged to the console along with the message "Scraper Error".

Please note that web scraping may be subject to legal restrictions and terms of service of the websites being scraped. Make sure to comply with all applicable laws and obtain proper authorization before scraping any website.

That's it! You now have a basic understanding of the web scraper code and how to use it to extract data from Google Shopping product pages. Happy scraping!

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 `handleSubmit` 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.

CHAPTER 5

Deployment

The deployment on Vercel (<https://deal-checker.vercel.app/>) ensures a reliable and scalable platform for users. Vercel's automatic deployment and CDN support contribute to a seamless user experience, with minimal downtime and optimal performance.

5.1. Setting up Vercel

- Creating a Vercel account
- Linking the project repository

5.2. Configuring Deployment Settings

- Choosing the deployment type
- Specifying the project root directory
- Selecting the deployment branch

5.3. Deploying the Project

- Triggering the initial deployment
- Monitoring the deployment process
- Verifying the deployment status

5.4. Testing the Deployment

- Accessing the deployed project URL
- Checking for any errors or issues
- Performing functional testing

5.6. Customizing the Deployment

- Configuring environment variables
- Setting up custom domains
- Enabling automatic deployments

CHAPTER 6

Conclusion

In conclusion, the "Deal Checker" project represents a cutting-edge and user-centric approach to online shopping, leveraging a carefully selected set of technologies to address the challenges users commonly encounter in the e-commerce landscape.

6.1. User-Centric Design:

The implementation of Next.js for dynamic UI ensures that users experience a seamless and responsive platform. The utilization of server-side rendering enhances the initial loading speed, while efficient client-side routing contributes to a smooth and enjoyable browsing experience. This user-centric design is further complemented by Tailwind CSS, which not only facilitates a stylish and visually appealing layout but also ensures consistency and ease of maintenance.

6.2. Efficient Data Retrieval and Parsing:

Axios, chosen for its simplicity and flexibility, facilitates efficient data retrieval from external APIs, ensuring that users have access to real-time and up-to-date product information. The integration of Cheerio for web scraping plays a pivotal role in extracting relevant data from HTML, enabling the platform to dynamically update product details and maintain accuracy.

6.3. Empowering Users through Comparison:

The implementation of dynamic and comparison API endpoints empowers users with the ability to make informed decisions. The dynamic API endpoint provides users with detailed and real-time product information, adapting to evolving market offerings.

In essence, "Deal Checker" is not merely a product comparison platform; it's a comprehensive solution designed to enhance the entire online shopping journey. By combining advanced technologies, a user-friendly interface, and a commitment to transparency, the platform stands as a testament to the continuous pursuit of excellence in the realm of digital commerce.

As the project evolves, there is a commitment to ongoing improvement and adaptation to user needs. The technologies selected and the thoughtful design choices underscore a dedication to providing users with a reliable, efficient, and enjoyable online shopping experience.

References

[1] Next.js Documentation:

The official documentation of Next.js served as a crucial resource in implementing dynamic UI components, server-side rendering, and client-side routing. [Next.js Documentation](#)

[2] Tailwind CSS Documentation:

The Tailwind CSS documentation provided extensive guidance on leveraging utility-first CSS for a consistent and visually appealing design. [Tailwind CSS Documentation](#)

[3] Cheerio Documentation:

The documentation for Cheerio was a valuable reference in implementing web scraping for efficient data extraction from HTML. [Cheerio Documentation](#)

[4] Axios Documentation:

The Axios documentation was consulted for best practices in making asynchronous HTTP requests, ensuring efficient data retrieval from external APIs. [Axios Documentation](#)

[5] Flowbit Documentation:

The Flowbit documentation provided insights into effective data parsing strategies, contributing to the overall efficiency of the web scraping process. [Flowbit Documentation](#)

[6] npm (Node Package Manager):

The npm website was used as a central hub for discovering, installing, and managing project dependencies, ensuring a streamlined development process. [npm Documentation](#)


[7] Vercel Documentation:

The Vercel documentation offered guidance on deploying and hosting the "Deal Checker" project, ensuring a reliable and scalable platform. [Vercel Documentation](#)

These resources played a vital role in the successful development, deployment, and maintenance of the "Deal Checker" project, contributing to its overall functionality, design, and user experience.

Appendix

[Source Code](#)

Deploy By  Vercel

Deal Checker PRO

The smart way to shop online.

Find the best deals with Deal Checker, the ultimate online shopping assistant.

[Get Started →](#)

[Docs →](#)

Find in-depth information about Deal Checker features and API.

[API Example →](#)

The API returns JSON format.

Deal Checker

[Home](#) [About](#)

Deal Checker

[Home](#) [About](#)

We invest in the world's potential

Here at Deal Checker we focus on markets where technology, innovation, and capital can unlock long-term value and drive economic growth.

[Search](#)

Looking for a great deal on your next purchase? Check out Deal Checker, the website that helps you compare prices and reviews from hundreds of websites. You'll never miss a bargain again!

Online shopping can be convenient and fun, but it can also be overwhelming and expensive. How do you know if you are getting the best price and quality for your products and services? That's where Deal Checker comes in. Deal Checker is a website that helps you find the best deals online by comparing prices and reviews from hundreds of websites. You can search for anything you want, from electronics to fashion to travel, and see the best offers in seconds. You can also filter your results by various criteria, such as market, language, safe search, freshness, aspect ratio, and more. Deal Checker also shows you the ratings and reviews from other customers, so you can make an informed decision. Deal Checker is the ultimate online shopping assistant that helps you save money, save time, and save hassle. Try it today and see for yourself!

Samsung S23 Ultra

Search

Deal Checker

HomeAbout

Search...

Samsung Galaxy S23 Ultra 5G
(12GB RAM, 256GB, Green)

★ 4.7 / 5 · 26,431 product reviews.

Amazon.in

Free delivery

₹1,21,999.00

→

Samsung Galaxy S23 Ultra 5G
(Phantom Black, 12GB, 256GB Storage)

★ 4.7 / 5 · 45,488 product reviews.

Ovantica.com

Free delivery by 19 Nov and free 7-day returns

₹87,900.00

→

Samsung Galaxy S23 Ultra 5G
(Cream, 12GB, 256GB Storage)

★ 4.7 / 5 · 45,488 product reviews.

Zebars

Free delivery

₹1,21,999.00

→

Samsung Galaxy S23 Ultra 5G
(Green, 12GB, 256GB Storage)

★ 4.7 / 5 · 45,488 product reviews.

Amazon.in

Free delivery

₹1,01,420.00

→

S23+ultra 4g/5g Smartphone
7.3"16gb Unlocked Android Mobile

★ No Data Available

eBay

+Delivery

₹14,556.68 + tax

→

Samsung Galaxy S23 Ultra 5G
SM-S918B Dual Sim 512GB Sky Blue (12GB RAM) - Dual ...

★ 4.7 / 5 · 44,341 product reviews.

Etoeren.com

₹2,332.81 delivery

₹1,47,165.87 + tax

→

SAMSUNG Galaxy S23 Ultra
5G (Green, 256 GB) (12 GB RAM)SAMSUNG Galaxy S23 Ultra ...

★ No Data Available

GCS4U

Free delivery

₹5,500.00

→

Buy Refurbished Samsung Galaxy S23 Ultra 5G - Refurbished - Green - 256 GB/12 GB ...

★ No Data Available

Cashify

Free delivery


₹89,999.00

→

Deal Checker

HomeAbout

Search...



Samsung Galaxy S23 Ultra 5G (12GB RAM, 256GB, Green)

4.5 out of 5 stars26,431 product reviews

SOLD BY	DETAILS	ITEM PRICE	TOTAL PRICE	
Amazon.in	Free delivery	₹1,21,999.00	₹1,21,999.00	Buy now
Samsung.com	Free delivery by Fri, 17 Nov15-day returns	₹1,24,999.00	₹1,24,999.00	Buy now
Flipkart	Free delivery by Wed, 22 Nov	₹1,24,999.00	₹1,24,999.00	Buy now
Croma	Free delivery by Thu, 16 Nov	₹1,24,999.00	₹1,24,999.00	Buy now
Reliance Digital	Free delivery by Fri, 17 NovFree 5-day returns	₹1,24,999.00	₹1,24,999.00	Buy now

HomeAbout

Vivo

amazon.in Deliver to Chandan Burla 768018 Electronics Search Amazon.in

EN Hello, Chandan Account & Lists Returns & Orders Cart

All Today's Deals Buy Again Amazon miniTV Sell Gift Cards Coupons Gift Ideas Amazon Pay Browsing History Chandan's Amazon.in Customer Service

Electronics Mobiles & Accessories Laptops & Accessories TV & Home Entertainment Audio Cameras Computer Peripherals Smart Technology Musical Instruments Office & Stationery

Electronics > Mobiles & Accessories > Smartphones & Basic Mobiles > Smartphones

Samsung Galaxy S23 Ultra 5G
(Green, 12GB, 256GB Storage)

Visit the Samsung Store
4.5 ★★★★★ 697 ratings
467 answered questions

Amazon's Choice

-19% ₹1,21,999
M.R.P.: ₹1,49,999

Fulfilled
Inclusive of all taxes
EMI starts at ₹5,915. No Cost EMI available EMI options
Buy now, pay in EMIs up to 12 months with Amazon Pay Later
Activate now and unlock welcome rewards worth ₹600

Offers
No Cost EMI Bank Offer Cashback

With Exchange
Up to ₹ 51,500.00 off

Without Exchange
₹ 121,999.00 ₹ 149,999.00

Fulfilled
FREE delivery **Wednesday, 15 November**. Order within 11 hrs 14 mins. Details
Deliver to Chandan - Burla 768018

In stock
Sold by Appario Retail Private Ltd and Fulfilled by Amazon.

Add a Protection Plan:
☐ Total Protection Plan by Quesst for ₹7,999.00

Add to Cart
Buy Now
Secure transaction

Project Links:

GitHub Repository:

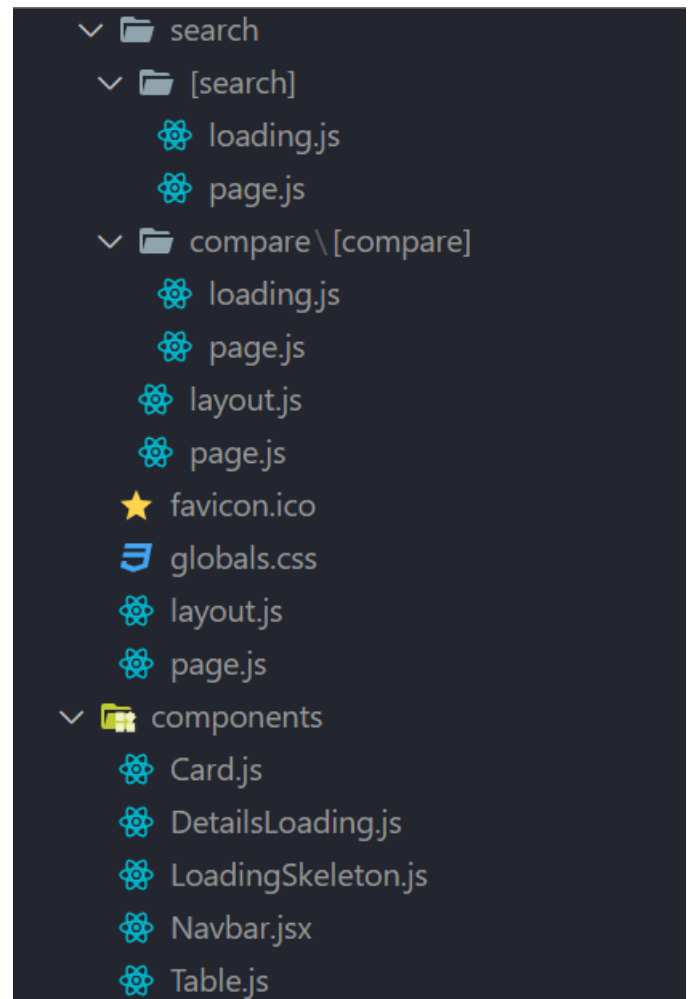
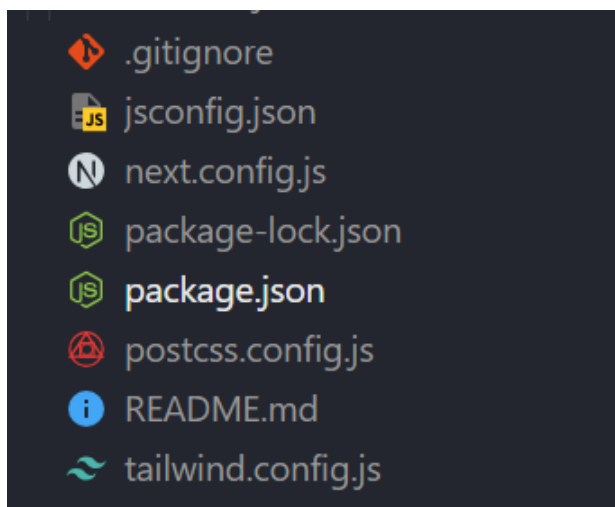
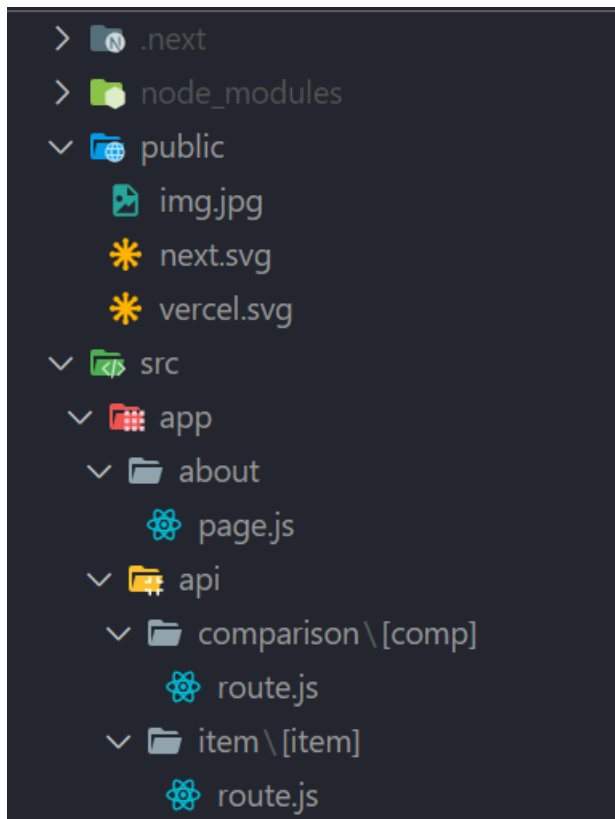
The source code and project files for "Deal Checker" can be found on the GitHub repository: [Deal Checker GitHub Repository](https://github.com/chandanPradhan09/College-Project-Minor) [https://github.com/chandanPradhan09/College-Project-Minor]

Live Deployment:

Explore the live deployment of "Deal Checker" and experience the platform in action: [Deal Checker Live Deployment](https://deal-checker.vercel.app/) [https://deal-checker.vercel.app/]

Source Code

Folder Structure:



app>api>comparison>route.js

```
import { NextResponse } from "next/server";

const axios = require("axios");
const cheerio = require("cheerio");

const scraper = async (url) => {
  try {
    let url2 = "";
    const baseUrl = "https://www.google.com";
    const axiosResponse1 = await axios.request({
      method: "GET",
      url: url,
      headers: {
        "User-Agent":
          "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36",
      },
    });
    const $$ = cheerio.load(axiosResponse1.data);
    url2 = baseUrl + $$(".EJbZzc").attr("href");
    if (url2 == "https://www.google.comundefined") {
      url2 = url;
    }
    const mainData = [];
    const content = [];
    const axiosResponse = await axios.request({
      method: "GET",
      url: url2,
      headers: {
        "User-Agent":
          "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36",
      },
    });
    const $ = cheerio.load(axiosResponse.data);
    const imgLink =
      $(".lBRvsb").find(".r4m4nf").attr("src") ||
      $(".Xkiaqc").find("img").attr("src");
    const title =
      $(".lBRvsb").find(".BvQan").text() || $(".LDQ1l").find(".BvQan").text();
    const review =
      $(".lBRvsb").find(".UzThIf").attr("aria-label") ||
      $(".aE5Gic").find(".UzThIf").attr("aria-label");
    const reviewOutOf =
      $(".lBRvsb").find(".HiT7Id").find("span").attr("aria-label") ||
      $(".aE5Gic").find(".HiT7Id").find("span").attr("aria-label");
  }
}
```

```

$(".sh-osd__offer-row").each((index, element) => {
  // scraping logic...
  const compBrand =
    $(element)
      .find(".kPMwsc")
      .find("a")
      .text()
      .replace("Opens in a new window", "") ||
    $(element)
      .find("._-ey")
      .find("a")
      .text()
      .replace("Opens in a new window", "");
  const complink =
    baseUrl + $(element).find(".kPMwsc").find("a").attr("href") ||
    baseUrl + $(element).find("._-ey").find("a").attr("href");
  const delivery = $(element).find(".yGibJf").text();
  const itemPrice = $(element).find(".fObmGc").text();
  const totalPrice = $(element).find(".drzWO").text();

  const dataFromResponse = {
    compBrand,
    complink,
    delivery,
    itemPrice,
    totalPrice,
  };
  content.push(dataFromResponse);
});
const mainDataContent = {
  title,
  review,
  reviewOutOf,
  imgLink,
  content,
};
mainData.push(mainDataContent);
return mainData;
} catch (error) {
  console.error(error);
  console.log("Scraper Error");
}
};

export async function GET(request, { params }) {
  const { comp } = params;
  const mainUrl = "https://www.google.com/shopping/product/";
  const url = mainUrl + comp + "?gl=in";

```

```

    const data = await scraper(url);
    return NextResponse.json(data);
}

```

app>api>item>route.js

```

import { NextResponse } from "next/server";

const axios = require("axios");
const cheerio = require("cheerio");

const main_url = "https://www.google.com";
const base_url = "https://www.google.com/search?tbm=shop&gl=in&hl=en&q=";

const scraper = async (url) => {
  const data = [];
  try {
    const axiosResponse = await axios.request({
      method: "GET",
      url: url,
      headers: {
        "User-Agent":
          "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36",
      },
    });
    const $ = cheerio.load(axiosResponse.data);
    $(".sh-dgr__content").each((index, element) => {
      // scraping logic...
      const comp = $(element).find(".iXEZD").attr("href");
      const link = main_url +
$(element).find(".zLPF4b").find("a").attr("href");
      let link_text = $(element).find(".shntl").find(".aULzUe").text();
      let rating = $(element).find(".NzUzee").find(".Rsc7Yb").text();
      let delivery = $(element).find(".vEjMR").text();
      let id = $(element)
        .find(".VOo31e")
        .find("a")
        .attr("href")
        .split("?")[0]
        .split("/")[3];

      let rating_str = $(element)
        .find(".NzUzee")
        .find("div")
        .find("span")
        .text();
    });
  } catch (error) {
    console.log(error);
  }
  return data;
};

```

```

    const total_people_review = rating_str.slice(
      rating_str.indexOf("stars.") + 7
    );
    if (link_text.startsWith(".")) {
      link_text = link_text.split(".").slice(-1)[0];
    }
    let Compare = null;
    if (comp !== undefined) {
      Compare = main_url + comp;
    }
    const title = $(element).find(".tAxDx").text();
    const price = $(element).find(".a8Pemb").text();
    const dataFromResponse = {
      title,
      price,
      Compare,
      link,
      link_text,
      rating,
      total_people_review,
      delivery,
      id,
    };
    data.push(dataFromResponse);
  });
} catch (error) {
  console.error(error);
}
return data;
};

export async function GET(request, { params }) {
  const { item } = params;
  const url = base_url + item;
  const data = await scraper(url);

  return NextResponse.json(data);
}

```

app>search>page.js

```

"use client";
import React, { useEffect } from "react";

import Card from "@components/Card";
import LoadingSkeleton from "@components/LoadingSkeleton";
import axios from "axios";

```



```

import { useState } from "react";

const page = ({ params }) => {
  const { search } = params;

  const [data, setData] = useState([]);
  const [Loading, setLoading] = useState(false);

  useEffect(() => {
    (async () => {
      setLoading(true);
      const { data } = await axios.get(
        `https://deal-checker.vercel.app/api/item/${search}`
      );
      setData(data);
      setLoading(false);
    })();
  }, [search]);

  return (
    <div>
      {Loading ? (
        <>
          /* <h1 className="mx-5 p-3 text-2xl">Searching.....</h1>
*/}

          <div className="flex-wrap flex justify-evenly w-fit">
            <LoadingSkeleton />
            <LoadingSkeleton />
            <LoadingSkeleton />
            <LoadingSkeleton />
            <LoadingSkeleton />
            <LoadingSkeleton />
            <LoadingSkeleton />
            <LoadingSkeleton />
            <LoadingSkeleton />
            <LoadingSkeleton />
          </div>
        </>
      ) : (
        <>
          <div className="flex-wrap gap-1 flex justify-evenly w-fit">
            {data?.map((item, index) => (
              <Card
                key={index}
                link_text={item.link_text}
                link={item.link}
                compare={item.Compare}
                title={item.title}

```

```

        rating={item.rating}
        id={item.id}
        price={item.price}
        total_people_review={item.total_people_review}
        delivery={item.delivery}
      />
    ))}
  </div>
</>
)}
</div>
);
};

export default page;

```

app>search>compare>page.js

```

"use client";
import Table from "@components/Table";
import DetailsLoading from "@components/DetailsLoading";
import axios from "axios";

import React, { useEffect, useState } from "react";

const page = ({ params }) => {
  const [dataItem, setDataItem] = useState({});
  const [loading, setLoading] = useState(false);

  const getData = async () => {
    setLoading(true);
    const { compare } = params;
    const { data } = await axios.get(
      `https://deal-checker.vercel.app/api/comparison/${compare}`
    );
    setDataItem(data[0]);
    setLoading(false);
  };

  useEffect(() => {
    getData();
  }, []);

  return (
    <>
      {loading ? (
        <DetailsLoading />
      ) : (

```

```

    <div className="relative overflow-x-auto min-h-screen shadow-md
sm:rounded-lg">
      <div className="flex p-2 text-gray-900 bg-white dark:text-white
dark:bg-gray-800">
        <span className="w-44 rounded-md bg-cover overflow-hidden">
          <img src={dataItem.imgLink} />
        </span>

        <div className="p-5 px-2 w-screen text-2xl font-semibold text-left
text-gray-900 bg-white dark:text-white dark:bg-gray-800">
          {dataItem.title}
          <p className="mt-1 text-lg font-normal text-gray-500 dark:text-
gray-400">
            {dataItem.review}
            {dataItem.reviewOutOf}
          </p>
        </div>
      </div>

      <table className="w-full text-sm text-left text-gray-500 dark:text-
gray-400">
        <thead className="text-xs text-gray-700 uppercase bg-gray-50
dark:bg-gray-700 dark:text-gray-400">
          <tr>
            <th scope="col" className="px-6 py-3">
              Sold By
            </th>
            <th scope="col" className="px-6 py-3">
              Details
            </th>
            <th scope="col" className="px-6 py-3">
              Item price
            </th>
            <th scope="col" className="px-6 py-3">
              Total price
            </th>
            <th scope="col" className="px-6 py-3">
              <span className="sr-only">Visit</span>
            </th>
          </tr>
        </thead>
        <tbody>
          {dataItem?.content?.map((item, index) => (
            <Table
              key={index}
              compBrand={item.compBrand}
              compLink={item.compLink}
              delivery={item.delivery}

```

```

            itemPrice={item.itemPrice}
            totalPrice={item.totalPrice}
        />
    ))}
</tbody>
</table>
</div>
)}
</>
);
};

```

```
export default page;
```

src>components>card.js

```

import Link from "next/link";
import { GrCompare } from "react-icons/gr";

const Card = (props) => {
  return (
    <div className="w-full max-w-xs mt-4 relative bg-white border border-gray-
200 rounded-lg shadow dark:bg-gray-800 dark:border-gray-700">
      <a href={props.link} target="_blank">
        {/* Image component Insert TODO */}
        <p className="bg-blue-500 h-3 rounded-b-2xl mb-2"></p>
      </a>
      <div className="px-5 pb-5">
        <a href={props.link} target="_blank">
          <h5 className="text-xl font-semibold tracking-tight text-gray-900
dark:text-white">
            {props.title}
          </h5>
        </a>
        <div className="flex items-center mt-2.5 mb-2.5">
          <div className="flex items-center">
            <svg
              className="w-4 h-4 text-yellow-300 mr-1"
              aria-hidden="true"
              xmlns="http://www.w3.org/2000/svg"
              fill="currentColor"
              viewBox="0 0 22 20">
              <path d="M20.924 7.625a1.523 1.523 0 0 0 -1.238-1.044l-5.051-
.734-2.259-4.577a1.534 1.534 0 0 0 -2.752 0L7.365 5.847l-5.051.734A1.535 1.535
0 0 0 1.463 9.213.656 3.563-.863 5.031a1.532 1.532 0 0 0 2.226 1.616L11

```

```

17.03314.518 2.375a1.534 1.534 0 0 0 2.226-1.617l-.863-5.03L20.537 9.2a1.523
1.523 0 0 0 .387-1.575Z" />
    </svg>
    {props.rating == "" ? (
      <code>No Data Available</code>
    ) : (
      <>
        <p className="ml-2 text-sm font-bold text-gray-900 dark:text-
white">
          {props.rating} / 5
        </p>
        <span className="w-1 h-1 mx-1.5 bg-gray-500 rounded-full
dark:bg-gray-400"></span>
        <span className="text-sm font-medium text-gray-900 dark:text-
white">
          {props.total_people_review}
        </span>
      </>
    )}
  </div>
</div>
<code className="text-sm font-medium text-gray-900 dark:text-white mb-
6 ml-0 mt-0">
  {props.link_text}
  <br />
  <span className="text-xs">{props.delivery}</span>
</code>
<div>
  <div className="text-3xl font-bold text-gray-900 dark:text-white">
    {props.price}
  </div>
  {props.compare !== null ? (
    <Link
      href={` /search/compare/${props.id}`}
      className="group text-white bg-blue-700 hover:bg-blue-800 font-
medium rounded-tl-2xl rounded-tr-2xl rounded-bl-2xl text-sm px-5 py-2.5 text-
center dark:bg-blue-600 dark:hover:bg-blue-700 absolute bottom-0 right-0 flex
items-center gap-2"
    >
      <GrCompare />
      <span className="inline-block transition-transform group-
hover:translate-x-1 motion-reduce:transform-none">
        -&gt;
      </span>
    </Link>
  ) : (
    <a
      href={props.link}

```

```

        className="group text-white bg-blue-700 hover:bg-blue-800 font-
medium rounded-tl-2xl rounded-tr-2xl rounded-bl-2xl text-sm px-5 py-2.5 text-
center dark:bg-blue-600 dark:hover:bg-blue-700 absolute bottom-0 right-0 flex
items-center gap-2"
        target="_blank"
      >
        <span className="inline-block transition-transform group-
hover:translate-x-1 motion-reduce:transform-none">
          -&gt;
        </span>
      </a>
    )}
  </div>
</div>
</div>
);
};

```

```
export default Card;
```

package.json

```

{
  "name": "clg-project-final",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "axios": "^1.6.0",
    "cheerio": "^1.0.0-rc.12",
    "next": "14.0.0",
    "react": "^18",
    "react-dom": "^18",
    "react-icons": "^4.11.0"
  },
  "devDependencies": {
    "autoprefixer": "^10",
    "postcss": "^8",
    "tailwindcss": "^3"
  }
}

```