**⭐ Star Rating ⭐**

**Introduction:**

The Star Rating project is a simple and interactive web component that allows users to rate products by selecting stars. The component dynamically updates the rating based on user interaction, providing a smooth and engaging user experience. The implementation follows best practices in React and CSS, ensuring a visually appealing and responsive design.

**About the Project:**

This project consists of a product card that displays product details such as title, description, price, and an image fetched from an API. Below the product information, a star rating system is implemented, allowing users to provide feedback dynamically. The rating system highlights stars based on hover and click events, updating the selected rating visually.

**Project Summary and Aim:**

**The main goal of this project is to:**

* Provide an interactive and user-friendly rating system.
* Demonstrate dynamic state management in React using useState.
* Enhance user engagement through visual feedback with hover effects.
* Fetch and display real product data from an API.

**Tools and Frameworks Used:**

* **React.js –** Component-based UI development.
* **CSS3 –** Styling, hover effects, and responsiveness.
* **JavaScript (ES6+) –** State management and dynamic interactions.
* **FakeStoreAPI –** Fetching product details dynamically to display the product in the star rating card component.

**Project Structure:**

A screenshot of a computer program

AI-generated content may be incorrect.

**File & Directory Explanations:**

1. **node\_modules/**
   * This folder contains all installed dependencies for the project, managed via npm or yarn.
   * Automatically generated upon running npm install.
2. **public/**
   * Contains static files like index.html, which is the main entry point for the web app.
   * Assets placed here are directly accessible via the browser.
3. **src/**
   * The core source code of the project.
   * **assets/**
     + Stores images, fonts, and other static files used in the project.
   * **css/**
     + App.css: Contains styling specifically for the App.jsx component.
     + index.css: Global styles applied throughout the application.
   * **App.jsx**
     + The root component that wraps other components like ProductCard.jsx.
     + Manages overall layout and state.
   * **main.jsx**
     + Entry point that renders the App.jsx component into the DOM.
   * **ProductCard.jsx**
     + A reusable component that displays product information, including:
       - Product image
       - Title
       - Description
       - Star rating system
4. **.gitignore**
   * Lists files and directories to be ignored by Git (e.g., node\_modules/, build/).
5. **eslint.config.js**
   * Configuration file for ESLint, which enforces coding style and detects potential issues.
6. **package-lock.json & package.json**
   * package.json: Defines the project metadata, dependencies, and scripts.
   * package-lock.json: Ensures consistent dependency versions.
7. **README.md**
   * Project documentation, including installation instructions, features, and usage.
8. **vite.config.js**
   * Configuration for Vite, a fast frontend build tool used in this project.

**Project Implementation:**

A screen shot of a computer

AI-generated content may be incorrect.

**Step 1: Importing Required Dependencies**

At the beginning of the file, we import the necessary React hooks:

A black rectangular object with white text

AI-generated content may be incorrect.

. **useState** → Manages the rating state for the star rating component.

. **useEffect** → Fetches product data from an external API when the **ProductCard** component mounts.

**Step 2: Implementing the Star Rating System (StarRating Component)**

A screen shot of a computer program

AI-generated content may be incorrect.

**. rating: Stores the selected rating (initially set to 0).**

**. hover: Temporarily stores the hovered star rating (for hover effect).**

**Rendering the Stars Dynamically:**

A screen shot of a computer screen

AI-generated content may be incorrect.

. We create an array of 5 elements ([...Array(5)]) to generate 5 stars.

. Each star:

* Has a key (index-based unique identifier).
* Uses CSS classes (filled) to change color based on the rating.
* Triggers:
  + **onClick** → Updates the selected rating.
  + **onMouseEnter** → Highlights stars up to the hovered index.
  + **onMouseLeave** → Resets the highlight.

**Step 3: Fetching Product Data (ProductCard Component)**

A screen shot of a computer program

AI-generated content may be incorrect.

. **product State** → Initially null, stores the fetched product data.

. **useEffect Hook** → Runs **only once** ([] as dependency array), fetching data from **FakeStoreAPI**.

**Step 4: Rendering Product Information**

A screen shot of a computer screen

AI-generated content may be incorrect.

**. Conditional Rendering:**

* If product exists, display product details (title, image, description, price).
* Otherwise, show a loading indicator (<p className="loader"></p>).

**.**  The StarRating component is used inside the card, allowing users to rate the product.

**Final Output Behavior:**

✅ Product card displays real-time data (image, title, description, price).  
✅ Users can rate the product by clicking on stars (rating updates dynamically).  
✅ Hovering over stars provides a preview before selection.  
✅ Interactive, responsive, and user-friendly UI**.**

**Sample-Result:**

**A black backpack with a logo on it

AI-generated content may be incorrect.**