

# Real-Time Automated Review Rating System– Frontend

## Overview

This page allows users to submit a review (text input). When submitted, the app calls an API that predicts the star rating for the review. The predicted rating is displayed with the review. A list of recent reviews with their predicted star ratings is also shown.

## Tech Stack

- **Framework:** React (TypeScript recommended for type safety)
- **Styling:** Tailwind CSS
- **API Calls:** Axios
- **State Management:** React Hooks (`useState`, `useEffect`) with custom hooks for API integration

## Features

- ✓ User can enter a review in a form.
- ✓ On submit, review is sent to the backend.
- ✓ Backend responds with a predicted star rating (1–5).
- ✓ Review and rating are displayed instantly on the UI.
- ✓ Recent Reviews are shown in reverse order (latest first).
- ✓ Review Found with individual rating counts, total review and average rating.

## UI Flow

### 1. Review Input Section

- Textarea for writing a review.
- "Submit" button.

## **2. Review Display Section**

- Show predicted star rating (  ) alongside user review.
- Keep the newest review at the top.

## **3. Recent Reviews Section**

- Display a list of previous reviews + their predicted star ratings.

## **4. Recent Count**

- Review Found with individual rating counts, total review and average rating.

# **Project Structure**

```
src/
  └── pages/
    ├── AddReview.tsx
    ├── Home.tsx
    ├── RecentReview.tsx
    ├── ReviewList.tsx
    └── index.ts
  └── routes/
    └── index.ts
  └── server/
    └── api.ts
  └── features/
    └── addReview/
```

```
| |   |-- components/
| |     └── AddReviewPage.tsx
| |   |-- hooks/
| |     └── useAddReview.tsx
| |   └── index.ts
|
| |
|   |-- home/
|   |   |-- components/
|   |   |   └── HomePage.tsx
|   |   |-- hooks/
|   |   |   └── index.ts
|   |
|   |
|   |-- recentReview/
|   |   |-- components/
|   |   |   └── RecentReviewPage.tsx
|   |   |-- hooks/
|   |   |   └── useRecentReview.tsx
|   |   └── index.ts
|
| |
|   |-- reviewListPage/
|   |   |-- components/
|   |   |   ├── ReviewListPage.tsx
|   |   |   ├── AllReviewList.tsx
|   |   |   ├── ReviewSection.tsx
|   |   |   └── Pagination.tsx
|   |   |-- hooks/
```

```
| |    └── useAllReviews.tsx  
| |    └── useReviewCount.tsx  
| |    └── index.ts  
|  
└── .env
```

## Components & Hooks

### 1. HomePage.tsx

```
const HomePage = () => {  
  return (  
    <div className="max-w-[1600px] mx-auto p-6 text-justify space-y-6">  
      <section>  
        <h2 className="text-2xl font-bold mb-2">  
          Automated Review Rating System for Food Products  
        </h2>  
        <p>  
          An intelligent system that analyzes customer reviews and predicts star ratings (1–5) using  
          a Transformer-based BERT model for food products.  
        </p>  
      </section>  
      <section>  
        <h2 className="text-xl font-semibold mb-2">Dataset Overview</h2>  
        <p>Dataset size: 1.6 lakh rows</p>  
        <ul className="list-disc pl-6">  
          <li>1 <span className="text-green-600">★</span>: 29,893</li>  
          <li>2 <span className="text-green-600">★</span>: 16,234</li>  
          <li>3 <span className="text-green-600">★</span>: 24,260</li>  
          <li>4 <span className="text-green-600">★</span>: 42,517</li>  
          <li>5 <span className="text-green-600">★</span>: 50,000</li>  
        </ul>  
      </section>  
    </div>  
  )  
}
```

```

        </ul>

<p>No. of Products: 40,704</p>

</section>

<section>

<h2 className="text-xl font-semibold mb-2">Sample Reviews</h2>

<div className="space-y-2">

<p><strong>Review:</strong> Worst cookies, absolutely disgusting taste <br />
<strong>Output:</strong> 1</p>

<p><strong>Review:</strong> Overpaid for this cereal box <br /> <strong>Output:</strong>
2</p>

<p><strong>Review:</strong> Energy drink tasted okay <br /> <strong>Output:</strong> 3</p>

<p><strong>Review:</strong> Imaginative sauce, uplifted my meal <br />
<strong>Output:</strong> 4</p>

<p><strong>Review:</strong> Superb chocolate, heavenly and addictive <br />
<strong>Output:</strong> 5</p>

</div>

</section>

</div>

);

};

export default HomePage;

```

## 2.AddReviewPage.tsx

```

import useAddReview from "../hooks/useAddReview"

const AddReviewPage = () => {

  const { handleSubmit, inputData, setInputData, loading } = useAddReview()

  return (
    <div className="max-w-[1600px] mx-auto p-4">
      <div className="my-4">
        <h2 className="text-xl font-semibold mb-4">Add Your Review</h2>
        <textarea

```

```

        value={inputData || ""}
        onChange={(e) => setInputData(e.target.value)}
        maxLength={400}
        placeholder="Enter Review..."
        className="border p-4 text-lg w-full h-32 rounded-lg border-gray-300 focus:outline-none
resize-none"
      />
      <button
        disabled={loading || !inputData}
        onClick={handleSubmit}
        className="my-2 bg-blue-600 px-4 py-2 rounded-lg text-white font-semibold">
        {loading ? "Submitting..." : "Submit"}
      </button>
    </div>
  </div>
}

export default AddReviewPage

```

### 3.useAddReview.tsx

```

import { useState } from "react"
import swal from 'sweetalert'
import api from "../../server/api"

const useAddReview = () => {
  const [inputData, setInputData] = useState<string>("")
  const [loading, setLoading] = useState(false)
  const handleSubmit = async () => {
    if (!inputData || inputData.trim() === "") {
      swal("Please Enter a Value")
      return
    }
  }
}

```

```

setloading(true)

try {
    await api.post(`reviews/create/`, { review_text: inputData })
    swal("Review Added Successfully")
    setInputData("")
} catch (error: any) {
    swal(error.response?.data?.message || "Failed Add Review")
} finally {
    setloading(false)
}
}

return { handleSubmit, inputData, setInputData, loading }
}

export default useAddReview

```

#### **4.RecentReviewPage.tsx**

```

import { CgChevronRight } from "react-icons/cg"
import { FaStar } from "react-icons/fa"
import { Link } from "react-router"
import useRecentReview from "../hooks/useRecentReview"
import useReviewCount from "../../reviewListPage/hooks/useReviewCount"
import ReviewSection from "../../reviewListPage/components/ReviewSection"

const RecentReviewPage = () => {
    const { isLoading, fetchData } = useRecentReview()
    const { fetchReviewCount } = useReviewCount()

    return (
        <>
        {
            isLoading ? (

```

```
<div className="flex justify-center items-center my-10">
  <div className="w-8 h-8 border-4 border-gray-300 border-t-blue-500 rounded-full
animate-spin"></div>
</div>
) : (
<div className="max-w-[1600px] mx-auto my-6 p-4">
  <div className="flex gap-4 items-center justify-between">
    <p className="text-3xl font-semibold drop-shadow-md ">Ratings & Reviews</p>
    <div className="bg-green-600 text-white flex items-center gap-1 w-16 rounded-2xl
justify-center">
      <span className="font-bold text-xl">{fetchReviewCount?.average_rating}</span>
      <FaStar />
    </div>
    <p className="text-gray-500 mr-auto font-semibold
textxl">{fetchReviewCount?.total_reviews} ratings</p>
    <Link to={'/add-review'} className="text-white text-lg font-semibold bg-blue-600
rounded-sm px-4 py-2">Rate Product</Link>
  </div>
<ReviewSection/>
<div className="my-4 space-y-4">
  {
    (fetchData?.length == 0 && !isLoading) ? (
      <p className="text-center text-lg my-10 font-semibold">No Recent Reviews</p>
    ) : (
      <>
        {fetchData?.map((item: any) => (
          <div key={item.id} className="border-b-3 border-gray-100 p-4 ">
            <div className="bg-green-600 text-white flex items-center gap-1 w-12
rounded-2xl justify-center">
              <span className="font-bold text-lg">{item?.predicted_rating
|| "0"}</span>
              <FaStar />
            </div>
          </div>
        ))
      </>
    )
  }
</div>
```

```

        <p className="mt-3 text-xl">{item?.review_text}</p>
        <p className="my-2 font-semibold text-gray-400">{item?.created_at}</p>
    </div>
)}
```

<Link to={'/review-list'} className="font-bold text-blue-600 flex items-center">

All {fetchReviewCount?.total\_reviews} reviews

<CgChevronRight className="size-5" />

</Link>

</>

)

</div>

</div>

)

}

</>

)

}

```

export default RecentReviewPage

```

## 5.useRecentReview.tsx

```

import { useEffect, useState } from "react"
import swal from 'sweetalert'
import api from "../../server/api";

const useRecentReview = () => {
    const [fetchData, setFetchData] = useState<any[]>()
    const [isLoading, setIsLoading] = useState(false)
    useEffect(() => {
        const fetchApi = async () => {
            setIsLoading(true)
            try {

```

```

        const response = await api.get(`/reviews/recent/`)
        setFetchData(response.data)
    } catch (error: any)
        swal(error.response?.data?.message || "Failed to Fetch Recent Review")
    } finally {
        setIsLoading(false)
    }
}
fetchApi()
}, [])
return { isLoading, fetchData }
}
export default useRecentReview

```

## 6. ReviewListPage.tsx

```

import ReviewSection from "./ReviewSection"
import useReviewCount from "../hooks/useReviewCount"
import useAllReviews from "../hooks/useAllReviews"
import AllReviewList from "./AllReviewList"

const ReviewListPage = () => {
    const { isLoading: isAllReviewsLoading } = useAllReviews();
    const { isLoading: isReviewCountLoading } = useReviewCount();
    const isLoading = isAllReviewsLoading || isReviewCountLoading;
    return (
        <>
        {
            isLoading ?
                <div className="flex justify-center items-center my-10">

```

```

    <div className="w-8 h-8 border-4 border-gray-300 border-t-blue-500 rounded-full animate-spin"></div>
  </div> :
  (
    <div className="my-6 max-w-[1600px] mx-auto p-4">
      <p className="text-2xl font-bold">All Reviews</p>
      <ReviewSection />
      <AllReviewList />
    </div>
  )
}

</>

)
}

export default ReviewListPage

```

## 7. ReviewSection.tsx

```

import { FaStar } from "react-icons/fa";
import useReviewCount from "../hooks/useReviewCount";

const ReviewSection = () => {
  const { fetchReviewCount } = useReviewCount();
  const total =
    fetchReviewCount?.ratings?.reduce(
      (acc: number, r: { count: number }) => acc + Number(r.count),
      0
    ) || 0;

  return (
    <div className="flex max-w-[1200px] mx-auto items-center gap-8">

```

```

/* Average rating */

<div className="p-4 border-r border-gray-400 w-[200px] flex flex-col items-center">

  <div className="bg-green-600 flex items-center gap-2 w-18 p-1 justify-center rounded-2xl">
    <span className="text-xl text-white font-bold">
      {fetchReviewCount?.average_rating}
    </span>
    <span>
      <FaStar size={20} className="text-white" />
    </span>
  </div>

  <p className="text-center text-gray-500 mt-2 font-semibold">
    {fetchReviewCount?.total_reviews} reviews
  </p>
</div>

/* Rating progress bar */

<div className="flex-1 p-4">
  {fetchReviewCount?.ratings?.map((item: any) => {
    const percentage = total > 0 ? (item.count / total) * 100 : 0;

    // Set color dynamically
    let barColor = "bg-green-500";
    if (item.star === 2) barColor = "bg-orange-500";
    if (item.star === 1) barColor = "bg-red-500";

    return (
      <div key={item?.star} className="flex items-center gap-4 mb-2">
        <span className="text-lg font-semibold">{item?.star}★</span>
        <div className="h-2 w-full bg-gray-100 rounded">
          <div
            className={`h-2 rounded ${barColor}`}
            style={{ width: `${percentage}%` }}
          />
    
```

```

        </div>
        <span className="font-semibold text-gray-500">
          {item.count || "0"}
        </span>
      </div>
    );
  )}

</div>
</div>
);

};

export default ReviewSection;

```

## 8. AllReviewList.tsx

```

import { FaStar } from "react-icons/fa"
import useAllReviews from "../hooks/useAllReviews"
import Pagination from "./Pagination"

const AllReviewList = () => {
  const { isLoading, fetchAllData, next, previous, fetchApi } = useAllReviews()
  return (
    <div className="max-w-[1600px] mx-auto my-6">
      <div>
        <p className="text-2xl font-bold drop-shadow-md">Ratings & Reviews</p>
      </div>
      <div className="my-4 space-y-4">
        {
          isLoading ? (
            <p className="text-center text-lg my-10 animate-pulse text-white/70">Loading...</p>
          ) : (fetchAllData?.length == 0 && !isLoading) ? (

```

```

<p className="text-center text-lg my-10 font-semibold">No Recent Reviews</p>

) : (
  fetchAllData?.map((item: any) => (
    <div key={item.id} className="border-b-3 border-gray-100 p-4 " >
      <div className="bg-green-600 text-white flex items-center gap-1 w-12 rounded-2xl justify-center">
        <span className="font-bold text-lg">{item?.predicted_rating || "0"}</span>
        <FaStar />
      </div>
      <p className="mt-3 text-xl">{item?.review_text}</p>
      <p className="my-2 font-semibold text-gray-400">{item?.created_at}</p>
    </div>
  ))
)
}

</div>

/* pagination */

{(next || previous) && (
  <Pagination
    isLoading={isLoading}
    next={next}
    previous={previous}
    fetchApi={fetchApi}
  />
)
}

</div>
)

}

export default AllReviewList

```

## 9. Pagination.tsx

```
type PaginationProps = {
    isLoading: boolean;
    next: string | null;
    previous: string | null;
    fetchApi: (url: string) => void;
};

const Pagination = ({ isLoading, next, previous, fetchApi }: PaginationProps) => {
    // Function to extract page number from URL
    const getPageNumber = (url: string | null): number | null => {
        if (!url) return null;

        const params = new URLSearchParams(url.split("?")[1]);
        return Number(params.get("page"));
    };

    const nextPage = getPageNumber(next);
    const prevPage = getPageNumber(previous);

    // Calculate current page
    let currentPage = 1;
    if (prevPage) {
        currentPage = prevPage + 1;
    } else if (nextPage) {
        currentPage = nextPage - 1;
    }

    return (
        <div className="flex justify-center items-center gap-4 mt-4">
            <button
                onClick={() => previous && fetchApi(previous)}
                disabled={!previous || isLoading}
                className="px-4 py-2 bg-gray-300 rounded disabled:opacity-50"
            >
                Previous
            </button>
            <span>Page {currentPage}</span>
            <button
                onClick={() => nextPage && fetchApi(nextPage)}
                disabled={!nextPage || isLoading}
                className="px-4 py-2 bg-gray-300 rounded disabled:opacity-50"
            >
                Next
            </button>
        </div>
    );
}
```

```

    >
    Previous
  </button>

  {/* ✅ Page Number Display */}

  <span className="text-lg font-medium">Page {currentPage}</span>

  <button
    onClick={() => next && fetchApi(next)}
    disabled={!next || isLoading}
    className="px-4 py-2 bg-blue-500 text-white rounded disabled:opacity-50"
  >
    Next
  </button>
</div>
);

};

export default Pagination;

```

## 10. useAllReviews.tsx

```

import { useEffect, useState } from "react"
import swal from 'sweetalert'
import api from "../../server/api";

const useAllReviews = () => {

  const [fetchAllData, setFetchAllData] = useState<any[]>()
  const [next, setNext] = useState<string | null>(null);
  const [previous, setPrevious] = useState<string | null>(null);
  const [isLoading, setIsLoading] = useState(false)

  const fetchApi = async (url: string = "/reviews/") => {
    setIsLoading(true)

```

```

try {
    const response = await api.get(url)
    setFetchAllData(response.data.results)
    setNext(response.data.next || null);
    setPrevious(response.data.previous || null);
} catch (error: any) {
    swal(error.response?.data?.message || "Failed to Fetch All Review")
} finally {
    setIsLoading(false)
}
}

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

return { isLoading, fetchAllData, next, previous, fetchApi }
}

export default useAllReviews

```

## 11. useReviewCount.tsx

```

import { useEffect, useState } from "react"
import swal from 'sweetalert'
import api from "../../server/api";

const useReviewCount = () => {
    const [fetchReviewCount, setFetchReviewCount] = useState<any>()
    const [isLoading, setIsLoading] = useState(false)
    useEffect(() => {
        const fetchApi = async () => {
            setIsLoading(true)
            try {
                const response = await api.get('/reviews/detail/')
                setFetchReviewCount(response.data)
            }
        }
    }, []);
}

```

```

    } catch (error: any) {
      swal(error.response?.data?.message || "Failed to Fetch Review count")
    } finally {
      setIsLoading(false)
    }
  }
  fetchApi()
}, []))

return { isLoading, fetchReviewCount }
}

export default useReviewCount

```

## API INSTANCE

```

import axios from "axios";
const api = axios.create({
  baseURL: import.meta.env.VITE_BASE_URL,
  headers: {
    "Content-Type": "application/json"
  }
})
export default api

```

## API Endpoints

- POST /reviews/create/
  - Input: { review\_text: "The food was great!" }
  - Output: { predicted\_rating: 5, text: "The food was great!" }
- GET /reviews/recent/
  - Returns a list of the most recent reviews.
- GET /reviews/
  - Returns all reviews with pagination.
- GET /reviews/detail/
  - Review found with individual rating counts, total review, and average rating.

## Running Locally

### Running Locally

1. Clone repository
2. Install dependencies

```
npm install
```

3. Start dev server

```
npm run dev
```

## Preview

### 1. Home

The screenshot shows the homepage of a web application. At the top, there is a blue header bar with the text "E-Cart" and a search input field labeled "Enter...". On the right side of the header, there are three links: "HOME", "ADD-REVIEW", and "REVIEW-LIST". The main content area has a white background. At the top of this area, the title "Automated Review Rating System for Food Products" is displayed in bold. Below the title, a subtitle reads "An intelligent system that analyzes customer reviews and predicts star ratings (1-5) using a Transformer-based BERT model for food products.". Underneath the subtitle, there is a section titled "Dataset Overview" which includes a table of data points:

Rating	Count
1 ⭐	29,893
2 ⭐	16,234
3 ⭐	24,260
4 ⭐	42,517
5 ⭐	50,000

No. of Products: 40,704

Below the dataset overview, there is a section titled "Sample Reviews" which lists five examples with their inputs and outputs:

- Input: Worst cookies, absolutely disgusting taste  
Output: 1
- Input: Overpaid for this cereal box  
Output: 2
- Input: Energy drink tasted okay  
Output: 3
- Input: Imaginative sauce, uplifted my meal  
Output: 4
- Input: Superb chocolate, heavenly  
Output: 5

### 2. Add Review

The screenshot shows the "Add Your Review" page. At the top, there is a blue header bar with the text "E-Cart" and two links: "HOME" and "REVIEW". The main content area has a white background. At the top, the heading "Add Your Review" is displayed. Below the heading, there is a text input field with the placeholder "Enter Review...". At the bottom of the input field, there is a blue "Submit" button.

### **3.Recent Review**

Ratings & Reviews		3.2 ★	71 ratings	<a href="#">Rate Product</a>
	71 reviews	5★	22	
		4★	13	
		3★	11	
		2★	5	
		1★	20	
	Overpaid for this cereal box			
	2025-08-26 20:15			
	I have owned a Keurig brewer for over 2 years and have never had a problem with any K-cups. I had previously purchased these K-cups as part of the Subscribe and Save program and had been happy.@@\$@%%\$%5656			
	2025-08-26 20:10			
	the perfect brew i love this tea and was so grateful when it showed up in my google search my aunt in new york started bringing this tea on her visits to north carolina at first i didnt want to try it but once i did i got hooked now i dont have to wait for her visits anymore i can order them right here on amazon			
	SAYING AS THE NAME			

## Ratings & Reviews

3.8★ 3,247 ratings and 151 reviews

[Rate Product](#)

What our customers felt:

- Look
- Colour
- Comfort
- Material Quality
- Light Weight
- True to Specs

Images uploaded by customers:

+ 47

**4★** Good product according to price

rohit badaraddi 4 months ago

Certified Buyer, Gokak Falls

16 4 ...

**5★** Pretty good in this price range.

Amartya Mondal 5 months ago

Certified Buyer, Suri

51 19 ...

**5★** nobelite Product is perfect good fantastic.. i return only for size problem

SHUBHAM SINGH S 6 months ago

Certified Buyer, Koraon

23 11 ...

[All 151 reviews >](#)

## 4.All Reviews



This screenshot shows the 'All Reviews' section with more detailed reviews listed below the summary.

**3.6 ★**  
11,096 ratings and 408 reviews

**Read reviews that mention:**

Overall   Look   Colour   Comfort   Material Quality  
Light Weight   True to Specs

**5★ Good 🔥**  
Kalpesh Bhavsar   Feb, 2024  
Certified Buyer, Ahmedabad

**3★ Useful 🌟**  
Mk majhar Ansari   Jan, 2024  
Certified Buyer, Dumka District

**4★ Paisa wasool poora ,**  
Rahil Khan   Jun, 2023  
Certified Buyer, Bengaluru

For each review, there are like and dislike counts (e.g., 7 likes, 2 dislikes) and a small edit icon.