Food Review Predictor – Frontend

This is a React + TypeScript frontend for a Food Review Predictor system.
Users can submit reviews, get an AI-predicted star rating, and see all past reviews with pagination.

Features

- Write a review using a form.
- Predict rating with AI (stars animation included).
- List reviews (paginated, “View All” & “Load More” supported).
- Reviews saved locally in localStorage.
- Clean UI with TailwindCSS styling.

Project Structure

```
src/
|
 ├── features/reviewPredictor/
|   ├── components/
|   |   ├── ReviewForm.tsx
|   |   └── ReviewList.tsx
|   |
|   ├── hooks/
|   |   └── useReviewPredictor.ts
|
 ├── lib/
|   └── api.ts
 ├── pages/
|   └──ReviewPredictor.ts
```

How It Works

- User writes a review → clicks Predict Rating.

- Frontend calls API: POST /reviews/predict/ → gets predicted rating.
- Review is added to the reviews list (with stars).
- Reviews are fetched from API: GET /reviews/?page=1.
    - Supports multiple API response formats (paginated).
- User can View All or Load More reviews.

Components & Hooks

- **ReviewForm.tsx**
    - Input box for writing reviews.
    - Button triggers `onPredict()`.

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

interface ReviewFormProps {

  review: string;

  setReview: (val: string) => void;

  onPredict: () => void;

}



const ReviewForm: React.FC<ReviewFormProps> = ({ review, setReview, onPredict }) => {

  const [isOpen, setIsOpen] = useState(false);



  return (

    <div className="w-full max-w-lg">

      {!isOpen ? (

        // ✅ Amazon-style button (collapsed)

        <button

          onClick={() => setIsOpen(true)}

          className="bg-white border border-gray-400 hover:bg-gray-100 text-gray-800 px-6 py-2 rounded-full transition w-full font-semibold"

        >
```

```
          Write a product review
        </button>
      ) : (
        // ✅ Expanded form (when clicked)
        <div className="mt-4 border border-gray-300 rounded-lg bg-white p-4 shadow-sm">
          <textarea
            value={review}
            onChange={(e) => setReview(e.target.value)}
            placeholder="Share your thoughts about this product..."
            className="w-full border border-gray-300 rounded-lg p-3 mb-4 focus:outline-none
focus:ring-2 focus:ring-orange-400 resize-none"
            rows={4}
          />
          <div className="flex justify-end gap-3">
            <button
              onClick={() => setIsOpen(false)}
              className="px-4 py-2 border border-gray-400 rounded-full text-gray-700 hover:bg-gray-
100"
            >
              Cancel
            </button>
            <button
              onClick={onPredict}
              className="px-6 py-2 bg-blue-600 hover:bg-orange-500 text-white rounded-full font-
semibold"
            >
              Submit
            </button>
          </div>
        </div>
      )}
    </div>
  );
};
export default ReviewForm;
```

- **ReviewList.tsx**
  - ➢ Displays list of reviews with stars ☆..
  - ➢ Shows either:
    - ○ 5 reviews only (default), or
    - ○ All reviews (showAll = true).
  - ➢ Buttons:
    - ○ View All Reviews (if >5 exist).
    - ○ Load More (pagination).

```
import React from "react";
import type { ReviewItem } from "../hooks/useReviewPredictor";

interface ReviewListProps {
  reviews: ReviewItem[];
  showAll: boolean;
  loadMore: () => void;
  viewAllReviews: () => void;
  hasMore: boolean;
  insights?: {
    summary: string;
  } | null;
```

```
}

const ReviewList: React.FC<ReviewListProps> = ({

  reviews,

  showAll,

  loadMore,

  viewAllReviews,

  hasMore,

  insights,

}) => {

  const displayedReviews = showAll ? reviews : reviews.slice(0, 5);

  return (

    <div className="mt-8">

     {/* ✅ Customers say section (plain style, no box) */}

{insights?.summary && (

  <div className="mb-6">

    <h2 className="text-lg font-semibold mb-2">Customers say</h2>

    <p className="text-gray-700">{insights.summary}</p>

    <p className="text-xs text-gray-400 mt-1 flex items-center">

      <span className="mr-1">ℹ</span>

      Generated from customer reviews

    </p>

  </div>

)}

      {/* ✅ Reviews list */}

      <h2 className="text-lg font-semibold mb-4">Top reviews</h2>

      {displayedReviews.length === 0 ? (
  <p className="text-gray-500">No reviews yet.</p>
) : (
  <ul className="space-y-6">
    {displayedReviews.map((item, index) => (
      <li key={index} className="border-b pb-4">
        {/* Date */}
        {item.formatted_date && (
          <p className="text-sm text-gray-500 mb-1">
            {item.formatted_date}
          </p>
        )}
        {/* Stars */}
        <div className="flex items-center text-yellow-400 mb-2">
          {Array.from({ length: 5 }).map((_, i) => (
            <span
              key={i}
              className={
                i < item.rating ? "text-yellow-400" : "text-gray-300"
              }
            >
              ★
            </span>
          ))}
          <span className="ml-2 font-medium text-gray-800">
            {item.rating} out of 5
          </span>
        </div>


        {/* Review text */}
        <p className="text-gray-700">{item.text}</p>
      </li>
    ))}
  </ul>
)}


{/* ✅ Buttons */}
<div className="mt-6 flex flex-col gap-3">
  {reviews.length > 5 && !showAll && (
    <button
      onClick={viewAllReviews}
      className="bg-gray-100 border rounded-md px-4 py-2 text-sm hover:bg-gray-200"
    >
      View all reviews
    </button>
  )}
  {showAll && hasMore && (
    <button
      onClick={loadMore}
      className="bg-gray-100 border rounded-md px-4 py-2 text-sm hover:bg-gray-200"
    >
      Load more reviews
    </button>
```

```
      )}

    </div>

  </div>

 );

};

export default ReviewList;
```

- **useReviewPredictor.ts (Hook)**
  Handles all logic:
  - Manage review input (review, setReview)
  - Fetch reviews from API (fetchReviews)
  - Predict rating (handlePredict)
  - Save reviews
  - Pagination (loadMore)
  - SweetAlert modals for feedback

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

import Swal from "sweetalert2";

import api from "../../../lib/api";



export interface ReviewItem {

  text: string;

  rating: number;

  formatted_date?: string;

}



export interface CustomerInsights {

  summary: string;

  key_points: string[];

  overall_sentiment: string;

  confidence_score: number;
```

```
}


export const useReviewPredictor = () => {
  const [review, setReview] = useState("");
  const [reviews, setReviews] = useState<ReviewItem[]>([]);
  const [page, setPage] = useState(1);
  const [hasMore, setHasMore] = useState(false);
  const [showAll, setShowAll] = useState(false);
  const [fetchingReviews, setFetchingReviews] = useState(false);
  const [predictedRating, setPredictedRating] = useState<number | null>(null);
  const [insights, setInsights] = useState<CustomerInsights | null>(null);


  const fetchReviews = async (pageNum = 1) => {
    try {
      if (pageNum === 1) setFetchingReviews(true);
      const { data } = await api.get(`/reviews/?page=${pageNum}`);


      if (!data) {
        if (pageNum === 1) setReviews([]);
        return;
      }


      let reviewsData = [];
      let nextPage = null;


      if (data.results && Array.isArray(data.results)) {
```

```
      reviewsData = data.results;
      nextPage = data.next;
    } else if (Array.isArray(data)) {
      reviewsData = data;
      nextPage = reviewsData.length >= 10;
    } else if (data.reviews && Array.isArray(data.reviews)) {
      reviewsData = data.reviews;
      nextPage = data.pagination?.has_next || data.has_more || data.next;
    } else {
      if (pageNum === 1) setReviews([]);
      return;
    }

    const mappedReviews = reviewsData.map((item: any) => ({
      text: item.text || item.review_text || "",
      rating: Math.round(item.predicted_rating || item.rating || 0),
      formatted_date: item.formatted_date || "",
    }));

    if (pageNum === 1) {
      setReviews(mappedReviews);
    } else {
      setReviews((prev) => [...prev, ...mappedReviews]);
    }
    setHasMore(Boolean(nextPage && mappedReviews.length > 0));
  } catch (error) {
    console.error("Error fetching reviews:", error);
    if (pageNum === 1) setReviews([]);
    setHasMore(false);
  } finally {
    if (pageNum === 1) setFetchingReviews(false);
  }
};

const fetchInsights = async () => {
  try {
    const { data } = await api.get("/reviews/customer-insights/");
    if (data && data.customer_insights) {
      setInsights(data.customer_insights);
    }
  } catch (err) {
    console.error("Error fetching insights:", err);
  }
};

useEffect(() => {
  fetchReviews(1);
  fetchInsights(); // fetch insights once
}, []);

useEffect(() => {
  if (reviews.length > 0) {
    localStorage.setItem("food-review-predictions", JSON.stringify(reviews));
  }
}, [reviews]);

const showAlert = (
  title: string,
  text: string,
  icon: "warning" | "error" | "success" | "info"
) => {
  Swal.fire({ title, text, icon, confirmButtonColor: "#facc15" });
};

const showAnimatedStars = (rating: number, callback?: () => void) => {
  const starsHTML = Array.from({ length: rating })
    .map(
      (_, i) =>
        `<span class="star" style="animation-delay: ${i * 0.2}s">⭐</span>`
    )
    .join("");

  Swal.fire({
    title: "Prediction Complete 🎉",
    html: `
      <div style="font-size: 2rem; display: flex; justify-content: center; gap: 8px;">
        ${starsHTML}
      </div>
      <p style="margin-top: 10px; font-size: 1.2rem;">Your review rating is ${rating} star${
    rating > 1 ? "s" : ""
  }</p>
      <style>
        .star { display: inline-block; opacity: 0; transform: scale(0.5); animation: popIn 0.5s forwards; }
        @keyframes popIn { to { opacity: 1; transform: scale(1); } }
      </style>
    `,
    confirmButtonColor: "#facc15",
  }).then(() => {
    if (callback) callback();
  });
};

const handlePredict = async () => {
  const trimmedReview = review.trim();

  // 🚫 Empty input
  if (!trimmedReview) {
    return showAlert("Invalid Review", "Please enter a valid review.", "warning");
  }

  // 🚫 Only numbers
  if (/^[0-9\s]+$/.test(trimmedReview)) {
    return showAlert(
      "Invalid Review",
      "Reviews cannot contain only numbers. Please write something meaningful ✍️.",
      "warning"
    );
  }

  // 🚫 Only special characters/emojis
  if (/^[^a-zA-Z0-9]+$/.test(trimmedReview)) {
    return showAlert(
      "Invalid Review",
      "Please enter a valid review with words, not just symbols or emojis.",
      "warning"
    );
  }

  // 🚫 Require at least 2 proper words (≥3 letters each)
  const words = trimmedReview.split(/\s+/).filter((w) => /^[a-zA-Z]{3,}$/.test(w));
  if (words.length < 2) {
    return showAlert(
      "Invalid Review",
      "Please write a meaningful review (at least 2 proper words).",
      "warning"
    );
  }

  try {
    const { data } = await api.post("/reviews/predict/", {
      review_text: trimmedReview,
    });

    const newReview = {
      text: data.text || data.review_text || trimmedReview,
      rating: Math.round(data.predicted_rating || data.rating || 0),
    };

    setReviews((prev) => [newReview, ...prev]);
    setReview("");
    setPredictedRating(newReview.rating);
    showAnimatedStars(newReview.rating);
  } catch (error) {
    console.error("Error predicting review:", error);
    showAlert("Error", "Something went wrong while predicting.", "error");
  }
};

const loadMore = () => {
  if (hasMore && !fetchingReviews) {
    const nextPage = page + 1;
    setPage(nextPage);
    fetchReviews(nextPage);
  }
};
```

```
  const viewAllReviews = () => {
    setShowAll(true);
    if (hasMore && page === 1) {
      loadMore();
    }
  };

  return {
    review,
    setReview,
    reviews,
    predictedRating,
    handlePredict,
    hasMore,
    loadMore,
    viewAllReviews,
    showAll,
    fetchingReviews,
    insights, // ✅ expose insights
  };
};
```

- **ReviewPredictor.tsx**
  The main page.
    - Combines ReviewForm + ReviewList.
    - Provides UI design

```
import React, { useMemo, useState } from "react";
import { useReviewPredictor } from "../features/reviewPredictor/hooks/useReviewPredictor";
import ReviewForm from "../features/reviewPredictor/components/ReviewForm";
import ReviewList from "../features/reviewPredictor/components/ReviewList";


const ReviewPredictor: React.FC = () => {
  const {
    review,
    setReview,
    reviews,
    handlePredict,
    loadMore,
    viewAllReviews,
    showAll,
    hasMore,
    insights,
    // predictedRating,
  } = useReviewPredictor();


  const [showInfo, setShowInfo] = useState(false);


  // ✅ Calculate average rating + distribution
  const { averageRating, distribution } = useMemo(() => {
    if (reviews.length === 0) {
      return { averageRating: 0, distribution: [0, 0, 0, 0, 0] };
    }
    const dist = [0, 0, 0, 0, 0];
    let total = 0;
    reviews.forEach((r) => {
      if (r.rating >= 1 && r.rating <= 5) {
        dist[r.rating - 1] += 1;
        total += r.rating;
      }
    });
    return {
      averageRating: total / reviews.length,
      distribution: dist,
    };
  }, [reviews]);

  // ✅ Star Renderer with half stars
  const renderStars = (rating: number) => {
    const fullStars = Math.floor(rating);
    const halfStar = rating % 1 >= 0.5;
    const emptyStars = 5 - fullStars - (halfStar ? 1 : 0);

    return (
      <span className="flex items-center text-yellow-400 text-xl">
        {"★".repeat(fullStars)}
        {halfStar && <span className="text-yellow-400">□</span>}
        {"☆".repeat(emptyStars)}
      </span>
    );
  };
```

```
return (
  <div className="min-h-screen bg-white flex flex-col px-6 py-6">
    {/* Header */}
    <h1 className="text-2xl font-semibold mb-4">Customer reviews</h1>

    <div className="flex flex-col md:flex-row gap-10">
      {/* Left - Rating Summary */}
      <div className="w-full md:w-1/3">
        {/* Average Rating */}
        <div className="flex items-center mb-2">
          <div className="text-3xl font-semibold">
            {averageRating.toFixed(1)}
          </div>
          <div className="ml-2">{renderStars(averageRating)}</div>
        </div>
        <p className="text-gray-700 text-sm mb-4">
          {reviews.length} global rating{reviews.length !== 1 ? "s" : ""}
        </p>

        {/* Distribution Bars */}
        <div className="space-y-1 mb-6">
          {distribution
            .map((count, index) => ({ stars: index + 1, count }))
            .reverse()
            .map(({ stars, count }) => {
```

```
      const percentage =
        reviews.length > 0 ? (count / reviews.length) * 100 : 0;
      return (
        <div
          key={stars}
          className="flex items-center text-sm text-gray-700 hover:text-blue-600 cursor-pointer"
        >
          <span className="w-12 hover:underline">{stars} star</span>
          <div className="flex-1 h-3 bg-gray-200 rounded mx-2">
            <div
              className="h-3 bg-yellow-400 rounded"
              style={{ width: `${percentage}%` }}
            ></div>
          </div>
          <span className="w-10 text-right">
            {Math.round(percentage)}%
          </span>
        </div>
      );
    })}
</div>

{/* How are ratings calculated */}
<div className="mb-6">
  <button
    onClick={() => setShowInfo(!showInfo)}
```

```
            className="text-blue-600 hover:underline text-sm"
          >
            How are ratings calculated?
          </button>
          {showInfo && (
            <p className="mt-2 text-sm text-gray-600 leading-relaxed border border-gray-200 rounded
p-3 bg-gray-50">
              To calculate the overall star rating and percentage breakdown by
              star, we don’t use a simple average. Instead, our system
              considers things like how recent a review is and if the reviewer
              bought the item. It also analyses reviews to verify
              trustworthiness.
            </p>
          )}
        </div>

        <hr className="my-4" />

        {/* Review this product */}
        <h2 className="text-lg font-semibold text-gray-800">
          Review this product
        </h2>
        <p className="text-gray-600 text-sm mb-3">
          Share your thoughts with other customers
        </p>
        <ReviewForm
          review={review}
```

```
    setReview={setReview}
    onPredict={handlePredict}
  />

  <hr className="my-4" />
</div>

{/* Right - Reviews */}
<div className="w-full md:w-2/3">
  <ReviewList
    reviews={reviews}
    showAll={showAll}
    loadMore={loadMore}
    viewAllReviews={viewAllReviews}
    hasMore={hasMore}
    insights={insights}
  />

  {/* Predicted Rating */}
  {/* {predictedRating !== null && (
    <div className="mt-6 p-4 border rounded-md bg-gray-50">
      <p className="font-semibold text-gray-800">
        AI Predicted Rating
      </p>
      <div className="flex items-center space-x-1 mt-2">
        {renderStars(predictedRating)}
```

```
            <span className="ml-2 text-gray-600 text-sm">
              {predictedRating} out of 5
            </span>
          </div>
        </div>
      )} */}
    </div>
   </div>
  </div>
 );
};


export default ReviewPredictor;
```

API Endpoints

- POST /reviews/predict/
  - Input: { review_text: "The food was great!" }
  - Output: { predicted_rating: 5, text: "The food was great!" }
- GET /reviews/page=1
  - Possible responses handled
  - Display reviewed date and place
- GET/reviews/customer-insights
  - It generates average feedback of customers

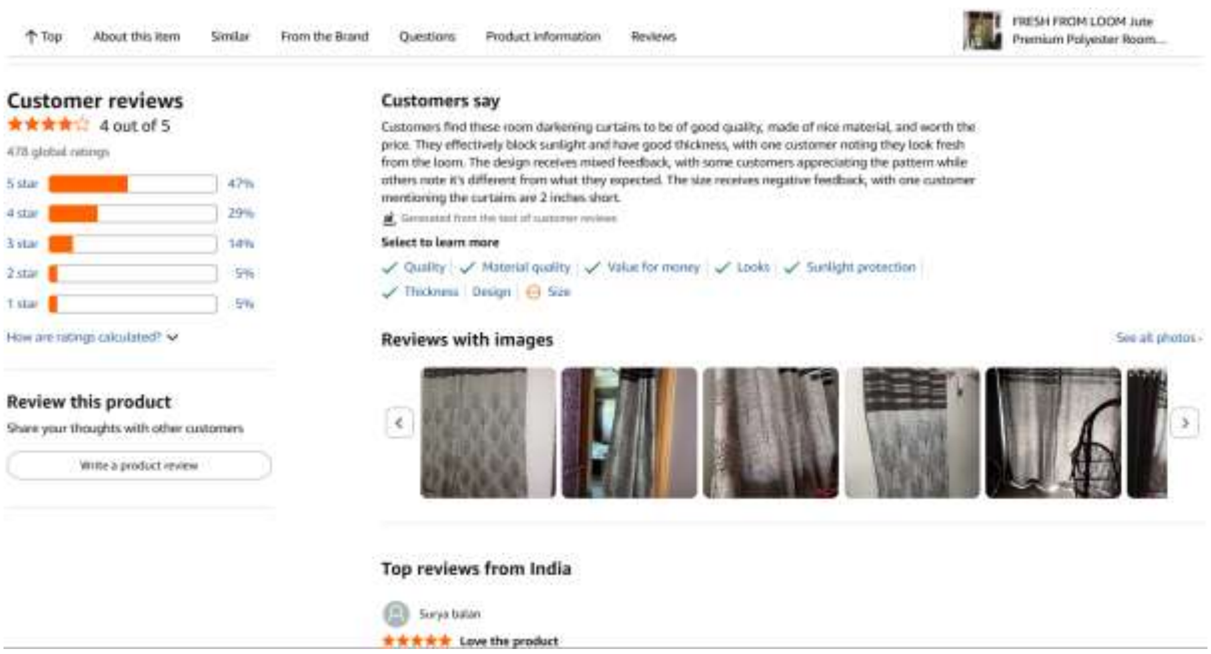
Running Locally

1. Clone repo
2. Install dependencies

   ```
   npm install
   ```
3. Start dev server

npm run dev

Screenshot of Reference Design(Amazone):



Customer reviews

4 out of 5

478 global ratings

5 star	47%
4 star	29%
3 star	14%
2 star	5%
1 star	5%

How are ratings calculated?

Review this product

Share your thoughts with other customers

Write a product review

Customers say

Customers find these room darkening curtains to be of good quality, made of nice material, and worth the price. They effectively block sunlight and have good thickness, with one customer noting they look fresh from the loom. The design receives mixed feedback, with some customers appreciating the pattern while others note it's different from what they expected. The size receives negative feedback, with one customer mentioning the curtains are 2 inches short.

Generated from the text of customer reviews

Select to learn more

✓ Quality | ✓ Material quality | ✓ Value for money | ✓ Looks | ✓ Sunlight protection | ✓ Thickness | Design | Size

Reviews with images

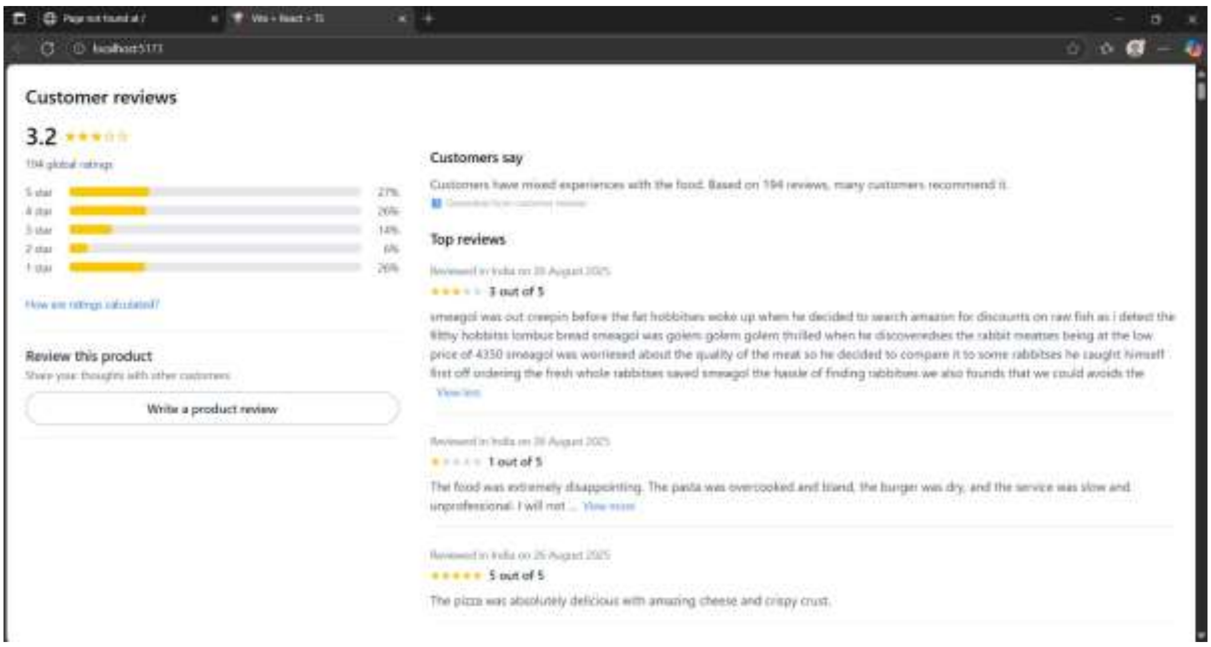
See all photos ›

Top reviews from India

Surya balan

Love the product

Screenshot of My Review Page:



Customer reviews

3.2

194 global ratings

5 star	27%
4 star	26%
3 star	14%
2 star	6%
1 star	26%

How are ratings calculated?

Review this product

Share your thoughts with other customers

Write a product review

Customers say

Customers have mixed experiences with the food. Based on 194 reviews, many customers recommend it.

[illegible]

Top reviews

Reviewed in India on 28 August 2025

3 out of 5

smeagol was out creepin before the fat hobbitses woke up when he decided to search amazon for discounts on raw fish as i detect the filthy hobbitss lombus bread smeagol was golem golem golem thrilled when he discoveredses the rabbit meatses being at the low price of 4350 smeagol was worriesed about the quality of the meat so he decided to compare it to some rabbitses he caught himself first off ordering the fresh whole rabbitses saved smeagol the hassle of finding rabbitses we also founds that we could avoids the

View less

Reviewed in India on 28 August 2025

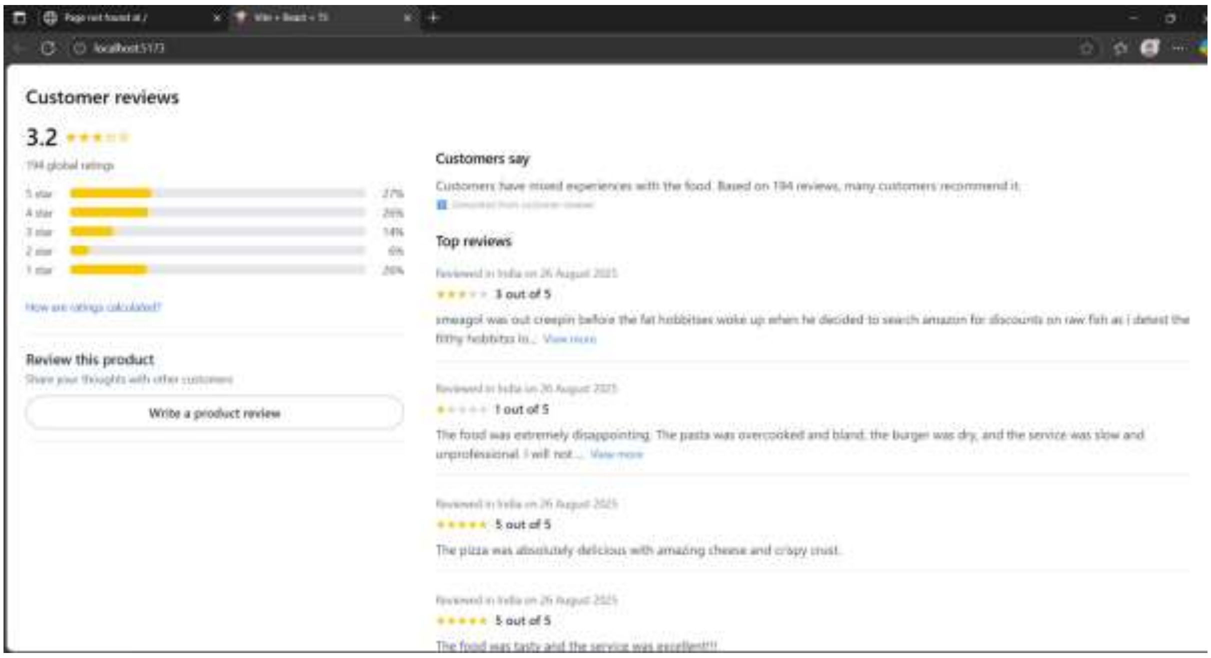
1 out of 5

The food was extremely disappointing. The pasta was overcooked and bland, the burger was dry, and the service was slow and unprofessional. I will not ... View more

Reviewed in India on 26 August 2025

5 out of 5

The pizza was absolutely delicious with amazing cheese and crispy crust.



Page not found at /

Vite + React + TS

localhost:5173

Customer reviews

3.2 ★★★☆☆

194 global ratings

5 star	▬	27%
4 star	▬	26%
3 star	▬	14%
2 star	▬	6%
1 star	▬	26%

How are ratings calculated?

Review this product

Share your thoughts with other customers

Write a product review

Customers say

Customers have mixed experiences with the food. Based on 194 reviews, many customers recommend it.

Generated from customer reviews

Top reviews

Reviewed in India on 26 August 2025

★★★☆☆ 3 out of 5

smeagol was out creepin before the fat hobbitses woke up when he decided to search amazon for discounts on raw fish as i detest the filthy hobbitss lo... View more

Reviewed in India on 26 August 2025

★☆☆☆☆ 1 out of 5

The food was extremely disappointing. The pasta was overcooked and bland, the burger was dry, and the service was slow and unprofessional. I will not... View more

Reviewed in India on 26 August 2025

★★★★★ 5 out of 5

The pizza was absolutely delicious with amazing cheese and crispy crust.

Reviewed in India on 26 August 2025

★★★★★ 5 out of 5

The food was tasty and the service was excellent!!!