Technical Assessment Project Report

Smart Recipe Generator

Sachin Chandra
22BEY10038

October 16, 2025

Contents

1 Abstract

The Smart Recipe Generator is a modern web application designed to address the common challenge of deciding what to cook based on available ingredients. This project, developed as a technical assessment, provides users with recipe suggestions by analyzing a list of ingredients, which can be entered manually or detected automatically from an uploaded photograph. The application features a clean, mobile-responsive user interface, and is built on a robust, serverless architecture using Next.js, Supabase, and the Hugging Face Inference API. This report details the system's architecture, core functionalities, implementation choices, and the final deployed product.

Keywords: React, Next.js, Serverless, AI, Machine Learning, Image Recognition, Supabase, Vercel.

2 Introduction

2.1 Problem Statement

In many households, a common problem is having a collection of ingredients without a clear idea of what meal to prepare. This often leads to food waste or repetitive meal choices. The Smart Recipe Generator aims to solve this by acting as a digital culinary assistant, intelligently suggesting recipes that can be made with the ingredients on hand.

2.2 Project Objectives

The primary objectives for this project were to develop a functional and user-friendly application with the following key features:

- **Ingredient Input:** Allow users to input ingredients via text.
- **Image Recognition:** Implement a feature to detect ingredients from a user-uploaded photo.
- **Recipe Matching:** Develop an algorithm to suggest relevant recipes from a database based on the provided ingredients.
- **Database:** Maintain a predefined database of at least 20 diverse recipes.
- **User Experience:** Ensure a clean, intuitive, and mobile-responsive design with proper loading and error states.
- **Deployment:** Host the application on a free, publicly accessible service.

3 System Architecture and Technology Stack

A modern, serverless architecture was chosen to ensure scalability, rapid development, and low operational overhead.

3.1 Technology Stack

- **Frontend:** Next.js (React) with Tailwind CSS & shadcn/ui.
- **Backend:** Next.js API Routes (Serverless Functions).
- **Database:** Supabase (PostgreSQL with an integrated API).
- **AI/ML Service:** Hugging Face Inference API.
- **Hosting:** Vercel.

3.2 Architecture Diagram

The application follows a decoupled architecture where the frontend communicates with various services through its own backend API routes. This abstracts away the complexity of third-party services from the client.

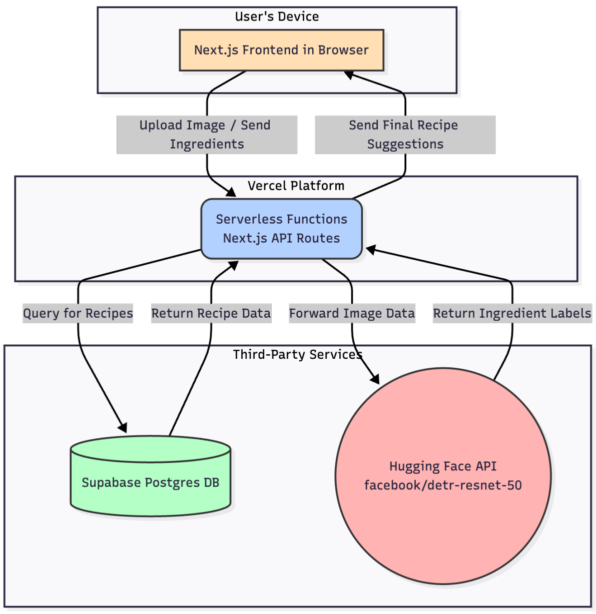
User's Device
Next.js Frontend in Browser
Upload Image / Send Ingredients
Send Final Recipe Suggestions
Vercel Platform
Serverless Functions Next.js API Routes
Query for Recipes
Return Recipe Data
Forward Image Data
Return Ingredient Labels
Third-Party Services
Supabase Postgres DB
Hugging Face API facebook/detr-resnet-50

Figure 1: High-Level System Architecture.

4 Implementation Details

This section covers the logic behind the application's core features.

4.1 Ingredient Recognition from Images

The image recognition feature is a key differentiator of this application.

1. The user uploads an image via the frontend.
2. The frontend sends this image as 'FormData' to a dedicated Next.js API route ('/api/ingredients/detect').

3. This backend route forwards the image buffer to the Hugging Face Inference API, specifically targeting the `facebook/detr-resnet-50` object detection model.

4. The model returns a list of detected objects with confidence scores. The backend filters these results to keep only high-confidence (>90%) detections, extracts the unique labels, and returns them to the frontend as a list of ingredients.

4.2 Recipe Matching Algorithm

The recipe matching logic is designed to prioritize recipes that are most complete given the user's ingredients.

1. The frontend sends the current list of user ingredients to the '/api/recipes/suggest' API route.

2. The backend queries the Supabase database to fetch all recipes.

3. For each recipe in the database, the algorithm calculates a "completeness score":

$$\text{Score} = \left(\frac{\text{Number of Matched Ingredients}}{\text{Total Ingredients in Recipe}}\right) \times 100$$

4. The results are then sorted in descending order based on this score. If two recipes have the same score, the one with fewer missing ingredients is ranked higher.

5. This sorted list of recipes is returned to the frontend for display.

4.3 User Experience Considerations

Significant effort was made to create a smooth user experience:

- **Loading States:** Spinners and loading toasts are used during API calls (both for recipe fetching and image analysis) to provide clear feedback to the user.

- **Error Handling:** User-friendly error messages are displayed using toasts if an API call fails or if no recipes are found.

- **Live Search:** The recipe list updates automatically and instantly as ingredients are added or removed, facilitated by a 'useEffect' hook with debouncing to prevent excessive API calls.

- **Mobile-First Design:** Tailwind CSS was used to ensure the application is fully responsive and usable on all screen sizes.

5 Deployment and Live Application

The application was deployed on Vercel, leveraging its seamless integration with Next.js and GitHub for continuous deployment (CI/CD).

- **GitHub Repository:** The complete source code is available at:
  github.com/SachinChandra2022/smart-recipe-generator

- **Live Application URL:** The deployed application is publicly accessible at: smart-recipe-generator-1brp.vercel.app

The deployment process is automated. Every push to the ‘main‘ branch on GitHub triggers a new build and deployment on Vercel. Environment variables, including API keys for Supabase and Hugging Face, are securely managed within the Vercel project settings.

6 Conclusion

The Smart Recipe Generator project successfully meets all the requirements outlined in the technical assessment. It demonstrates proficiency in full-stack development using a modern, serverless tech stack. The application effectively solves the core problem of recipe discovery through both text and innovative image-based ingredient detection. The final product is a functional, aesthetically pleasing, and well-architected web application deployed for public access.