

# Product Analyser – A Cloud-Based MERN Stack Application

Srijan Kumar Dubey

E22CSEU1021

Lakshya Deewan

E22CSEU1012

---

## 1. Introduction

In an age where consumers are becoming increasingly health-conscious and ingredient-aware, understanding product content—especially in cosmetics, food, and personal care—has never been more important. The **Product Analyser** addresses this need by providing an intuitive platform that scans product labels and evaluates ingredient safety using cloud services and modern web technologies.

This document outlines the architecture, functionality, and future roadmap for the Product Analyser application, built using the **MERN stack** and powered by **AWS services** for OCR and AI-based interaction.

---

## 2. Objective

The goal of the **Product Analyser** project is to develop a comprehensive and user-friendly web application designed to simplify the process of analyzing and understanding product ingredients. The application leverages cutting-edge technology to serve several key functions that benefit the end-user:

1. **Extracts Ingredients from Product Label Images Using OCR (Optical Character Recognition):** The web application uses advanced OCR technology to scan and extract ingredient lists from product label images. This allows users to quickly and efficiently access ingredient information from any product they encounter, without needing to manually read the fine print on packaging.

2. **Grades Ingredients Based on a Predefined Safety Database:** Once the ingredients are extracted, the application cross-references them with a predefined safety database. This database includes information on the safety, toxicity, and potential allergens associated with various ingredients, allowing the application to grade each ingredient based on its safety profile. This helps users make informed decisions about the products they are considering.
  3. **Visualizes Safety Information Through Interactive Graphs:** The application provides an intuitive way for users to view the safety information of ingredients through interactive graphs and charts. These visualizations make it easy to understand the relative safety of various ingredients, as well as any potential risks associated with them. Users can interact with these graphs to dive deeper into specific data points or explore related information.
  4. **Allows Real-Time Interaction with an AI Chatbot for Product-Related Queries:** To enhance user experience, the application includes an AI-powered chatbot. This chatbot is capable of answering product-related queries in real time, allowing users to ask questions about specific ingredients, potential health impacts, or safety concerns. The AI chatbot is designed to be highly responsive and can provide personalized advice based on the product's ingredients and the user's preferences or concerns.
- 

### 3. Features

The application includes the following major features:

#### **Image Upload**

Users can upload images of product labels (cosmetics, food, personal care). These images are processed and analyzed for ingredient content.

#### **OCR Extraction**

The uploaded image is sent to **AWS Rekognition**, which uses **Optical Character Recognition (OCR)** to extract textual information from the image.

### **Ingredient Safety Grading**

The extracted ingredients are matched with a safety database stored in **MongoDB**. A 3-tier grading system categorizes ingredients into:

- Safe
- Moderate Risk
- High Risk

### **Graphical Visualization**

Data is presented in a visually intuitive format using **Chart.js** with:

- Circular (donut) graphs for safety distribution
- Bar charts showing the number of ingredients per safety grade

### **Real-Time Chatbot**

A **chatbot** powered by **AWS Lambda** and **AWS Bedrock** allows users to ask product-specific questions. The chatbot responds with real-time AI-generated answers based on extracted ingredients.

---

## **4. Technologies Used**

### **Frontend:**

- React.js – For building user interfaces
- Axios – For making HTTP requests to the backend
- Chart.js – For rendering graphs

### **Backend:**

- Node.js & Express.js – RESTful API server
- MongoDB – Database for ingredient safety grades

### **Cloud Services (AWS):**

- Amazon S3 – Stores uploaded product label images
- Amazon Rekognition – Extracts text using OCR

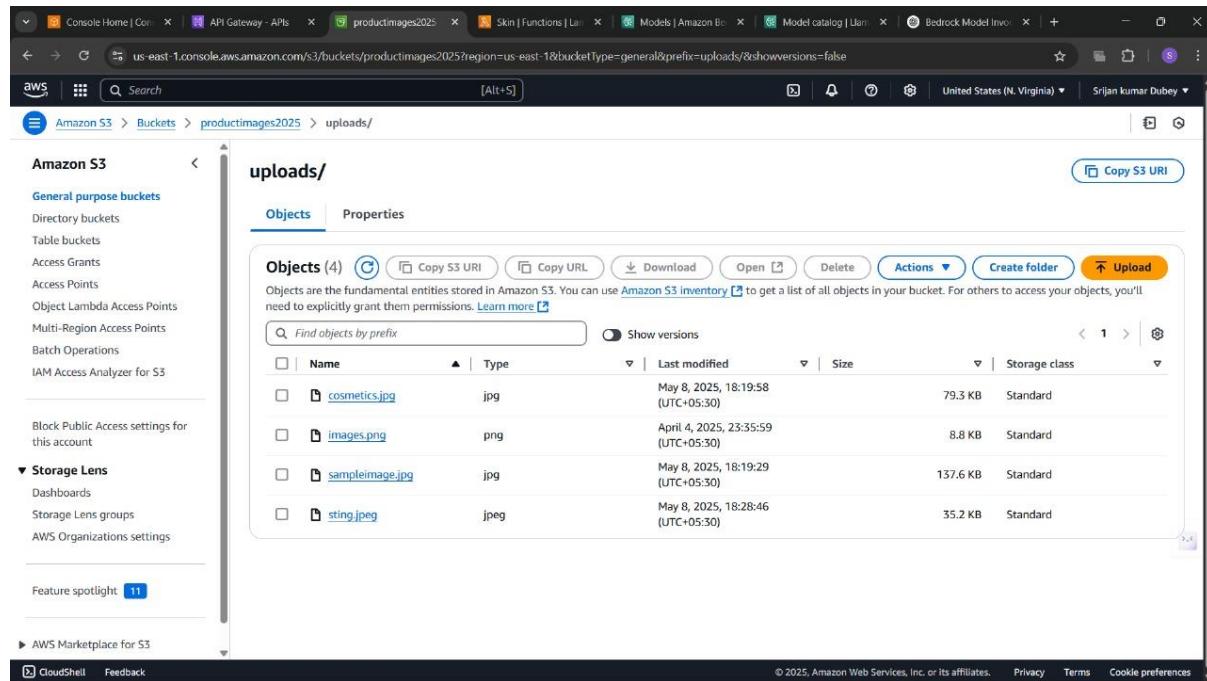
- AWS Lambda – Executes chatbot logic
  - AWS Bedrock – Powers the real-time AI chatbot
  - AWS API Gateway
- 

## 5. Application Workflow

To make everything run smoothly from the moment a user uploads an image to when they get safety results and chatbot responses, the app follows a clear workflow. This involves multiple cloud services and backend processes working together with the frontend to deliver fast and accurate results. Here's a step-by-step breakdown of how the application works behind the scenes:

### 1. Uploading the Image to S3

When the user uploads a product label image from the frontend, it gets sent to the backend (built using Express.js). From there, the image is uploaded to Amazon S3, which is just cloud storage provided by AWS. This is where we store all the images safely before analyzing them.

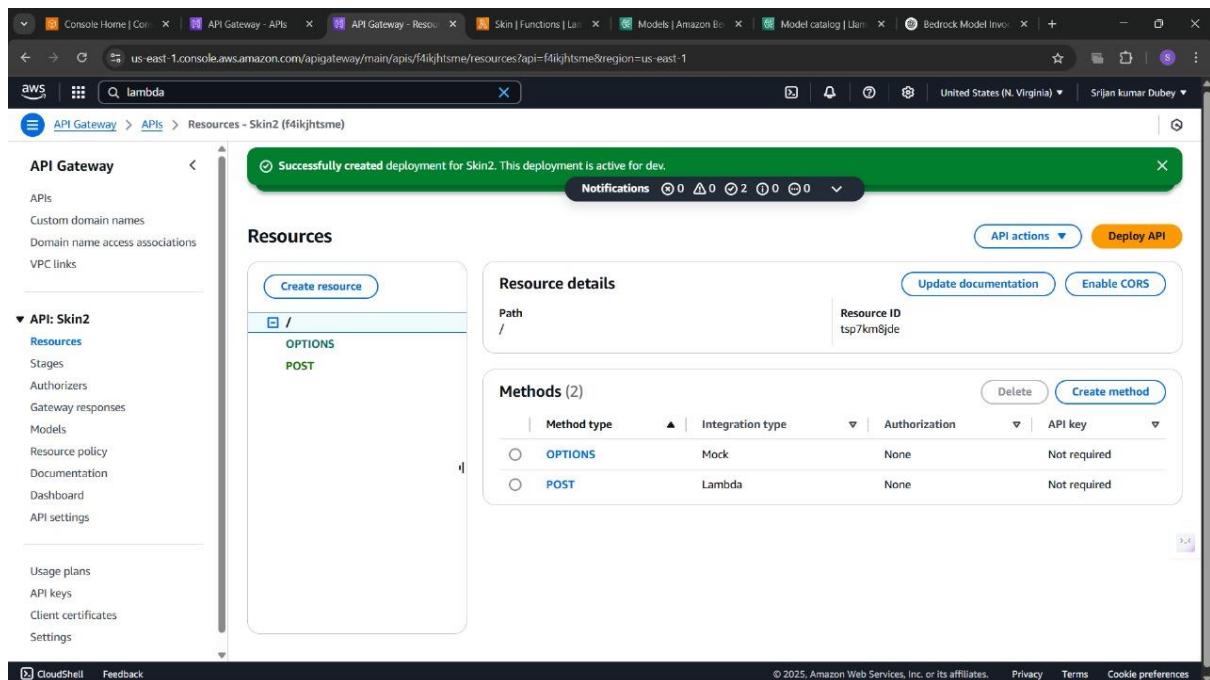


## 2. Extracting Text with Rekognition

Once the image is in S3, the backend sends it to AWS Rekognition to do OCR (optical character recognition). Rekognition goes through the image and pulls out any text it can find — in our case, that means the list of ingredients on the label.

## 3. Matching Ingredients and Giving Safety Grades

The extracted text is split into individual ingredients. Each one is checked against a collection in MongoDB, which contains the ingredient names and their safety grades. We use a 3-level grading system: Safe, Moderate Risk, and High Risk.



## 4. Showing the Results with Graphs

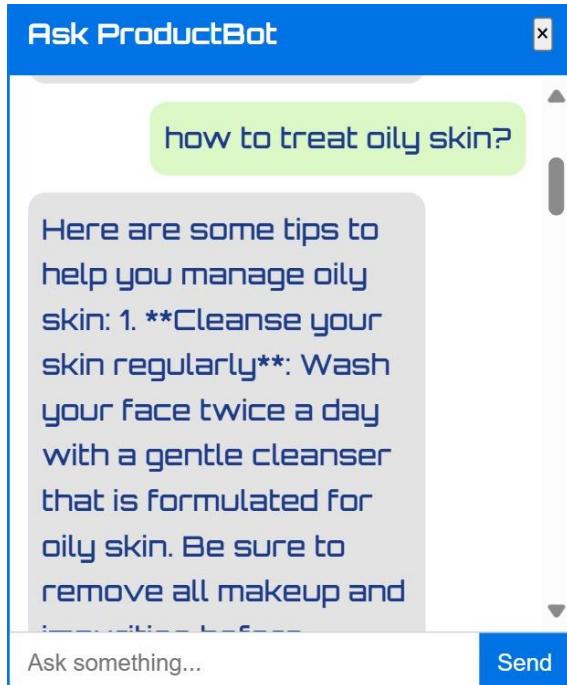
After grading, the backend sends all the results back to the frontend. We use Chart.js to show the ingredient grades using a circular chart and a bar chart. This helps users quickly see if the product is safe or not.

## 5. Chatbot with API Gateway, Lambda & Bedrock

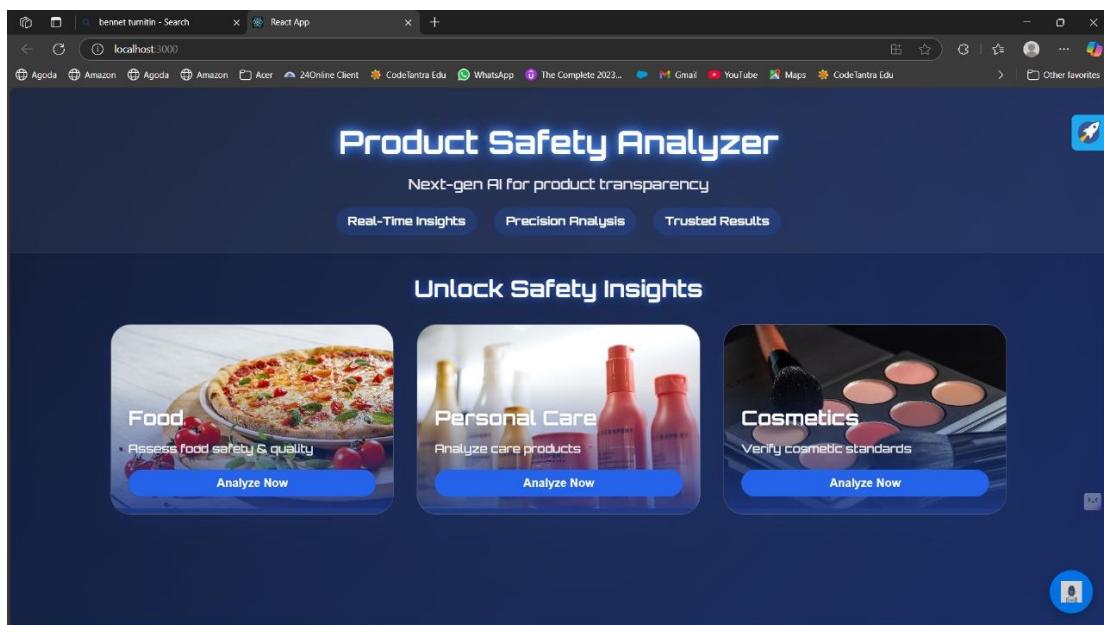
There's also a chatbot in the app. When the user types a question, the frontend sends the message through API Gateway, which is basically a bridge that connects frontend requests to AWS services. It then triggers a Lambda function, and inside that function, we use AWS Bedrock to generate an AI response. The chatbot tries to give helpful answers based on the product and its ingredients.

## 6. Screenshots and UI

### Chatbot



### Home Page



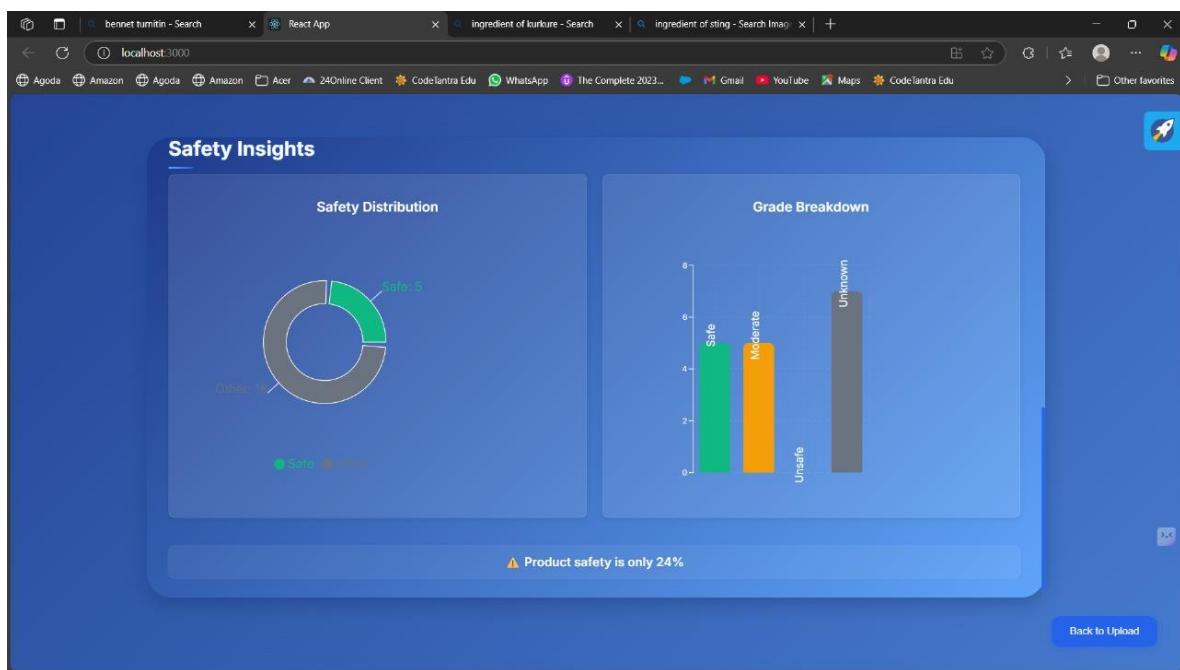
## Ingredients Breakdown through Image

The screenshot shows a 'Product Snapshot' interface. At the top, there is a red box containing a list of ingredients: Carbonated Water, Sugar, Citric Acid, Artificial Flavour, Maltdextrin, Sodium Citrate, Sodium Hexametaphosphate, Taurine, Potassium Sorbate, Caffeine (200 mg/l), Sodium Benzoate, Allura Red, Inositol, Calcium Disodium EDTA, Vitamin B3, Ginseng Powder (40 mg/l), Vitamin B6, and Vitamin B12. Below this, the 'Ingredient Breakdown' section displays the same ingredients in a grid format, each with its name and a 'Grade' value:

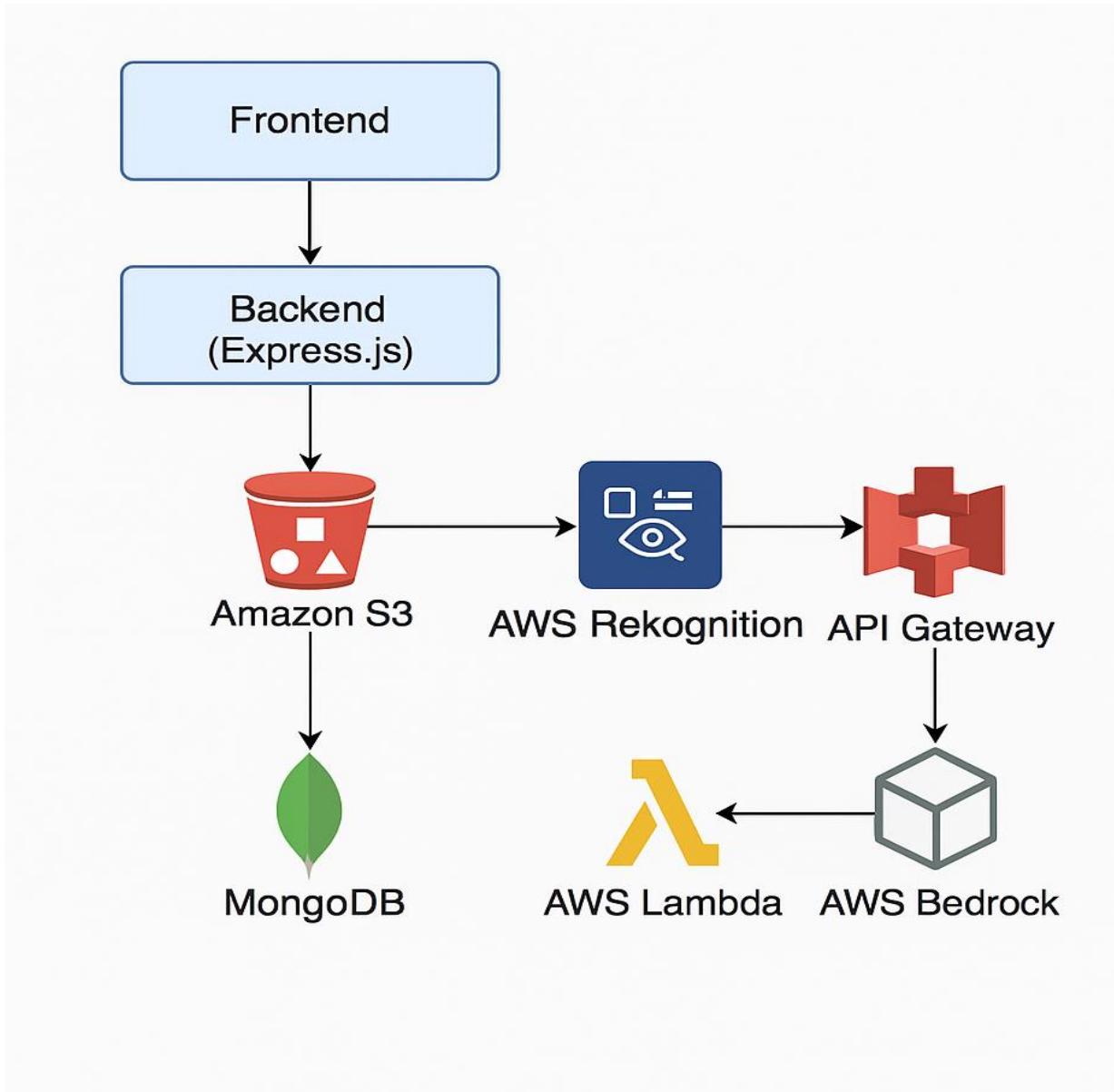
Ingredient	Grade
Carbonated Water	Grade: 1
Citric Acid	Grade: 2
Sugar	Grade: 3
Artificial Flavor	Grade: 3
Maltdextrin	Grade: 2
Sodium Citrate	Grade: 1
Sodium	Grade: Unknown
Taurine	Grade: 1
Hexametaphosphate	Grade: Unknown
Potassium Sorbate	Grade: 2
Caffeine (200 mg/l)	Grade: 2
Sodium Benzoate	Grade: 3
Allura Red	Grade: 3
Inositol	Grade: 1
Calcium	Grade: Unknown
Disodium EDTA	Grade: 3
Vitamin B3	Grade: Unknown
Ginseng Powder	Grade: Unknown
(40 mg/l)	Grade: Unknown
Vitamin B6	Grade: 1
Vitamin B12	Grade: Unknown

At the bottom right of the interface is a 'Back to Upload' button.

## Graph Visualization



## 7. Architecture Diagram



## 8. Usage Summary

Here's a quick overview of how users interact with the app and what happens at each step. From uploading the label to chatting with the AI, each part of the flow is designed to be smooth, informative, and helpful. The table below summarizes the key actions and what they do behind the scenes.

Task	Description
Image Upload	Users start by uploading a clear image of a product label (for cosmetics, food, or personal care). This image is then stored securely in Amazon S3.
OCR Extraction	The stored image is sent to AWS Rekognition, which performs Optical Character Recognition to extract any visible text—mostly ingredient lists.
Safety Analysis	Once ingredients are extracted, they are parsed and checked against a database in MongoDB that holds safety grades. Each ingredient is tagged as Safe, Moderate Risk, or High Risk.
Visual Feedback	The final graded list is sent to the frontend where it's visualized using Chart.js in the form of circular and bar graphs, giving users a clear idea of the product's safety level.
AI Interaction	Users can chat with a built-in assistant powered by AWS Bedrock and Lambda. The chatbot can answer questions about ingredients, their effects, or general product safety.

---

## 9. Future Improvements

To make the platform more robust and scalable, future enhancements could include:

- **Expanding the Ingredient Database**  
Add more ingredient records and better categorization.
- **Improved Chatbot Intelligence**  
Enhance context-awareness and domain understanding in chatbot.
- **Advanced Safety Metrics**  
Include toxicity, allergenicity, and regulatory compliance indicators.

---

## 10. Conclusion

Product Analyser is a smart web application that brings together cloud computing, artificial intelligence, and the MERN stack to help users better understand the ingredients in their everyday products. Whether it's a skincare item, food, or something used for personal care, users can upload a label image and instantly get a breakdown of its ingredients. The app uses AWS Rekognition to extract text from the image and matches the ingredients with a database to assign safety grades, which are then shown through simple, interactive graphs.

What makes this project special is how everything is connected behind the scenes. From image storage in Amazon S3 to real-time chat responses powered by AWS Bedrock and Lambda, the app shows how multiple modern technologies can work together to solve a real-world problem. It's more than just a tool—it's an example of how OCR, AI, and full-stack development can be used to make information more accessible and useful for everyday decisions.