# CS651 | Project 2 – Smart Recipe Tagger

**Group:**
Sai Vijay
Sahith Reddy
Moksha Shah
Aryan B.

---

**Application Name:** Smart Recipe Tagger

**Github:**[https://github.com/VIJAYRUR/cs651-project2-smart-recipe-tagger](https://github.com/VIJAYRUR/cs651-project2-smart-recipe-tagger)

## Purpose:

Smart Recipe Tagger is a Single Page Application (SPA) that connects to a user's **Pinterest** account to analyze their **food pins** using **Google Vision** and **Google Gemini APIs**.

The app automatically detects food items and ingredients in each pin using Google Vision, then uses Google Gemini (vision + language model) to create short, human-readable **recipe summaries** and **category tags** (such as "Dessert," "Italian," or "Vegetarian").

All processed data is stored in **Google Firestore**, and users can explore their tagged recipes in an interactive **React dashboard** with filtering, search, and analytics capabilities.

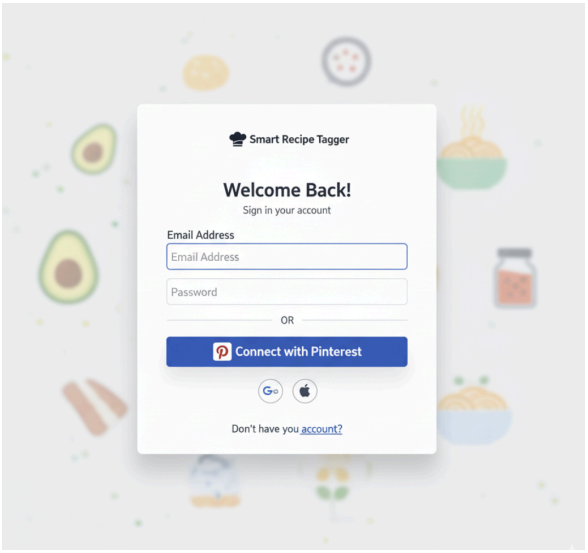
The backend is built with **Node.js and Express**, and the complete SPA is deployed on **Google Cloud Run**. The application will be linked from the group's **Project 1 static site**, as required.

## Key Features:

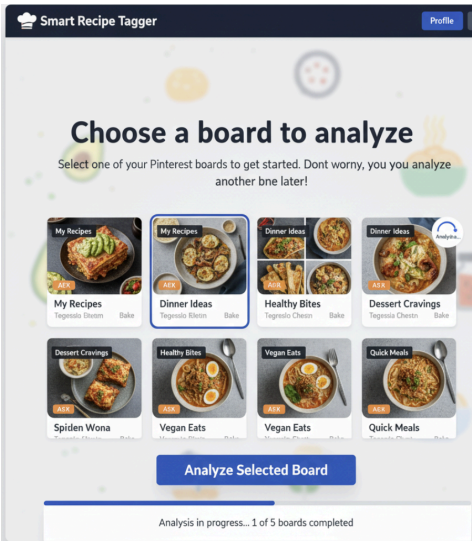
- Pinterest login via OAuth 2.0
- Google Vision API → Detect food items, ingredients, and text
- Google Gemini API → Generate recipe names, short summaries, and category tags
- Google Firestore → Store processing results per pin
- React dashboard → View pins with AI-generated summaries and filters
- Comprehensive API call logging and analytics for performance tracking

# Mockup Interfaces

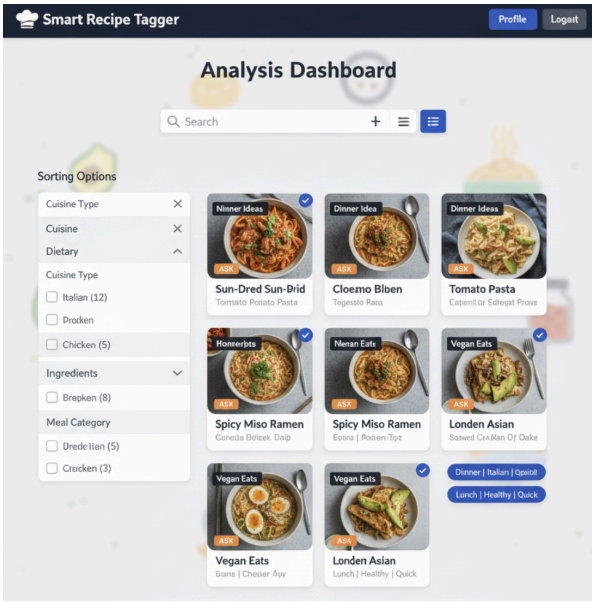
## Login Page



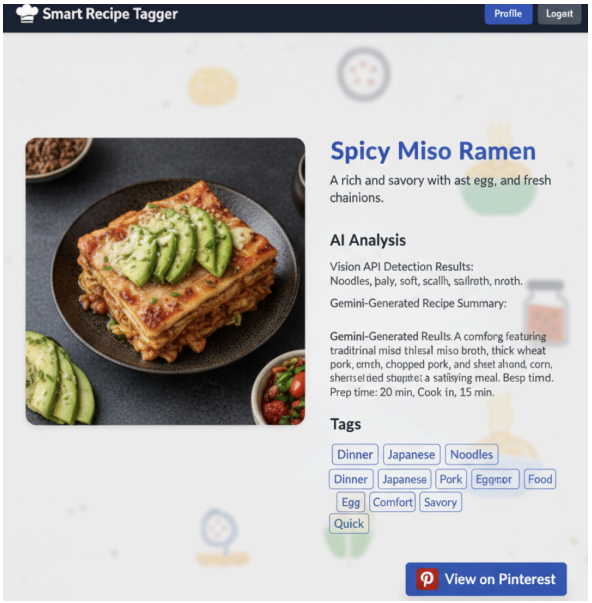

## Board Selection




## Analysis Dashboard



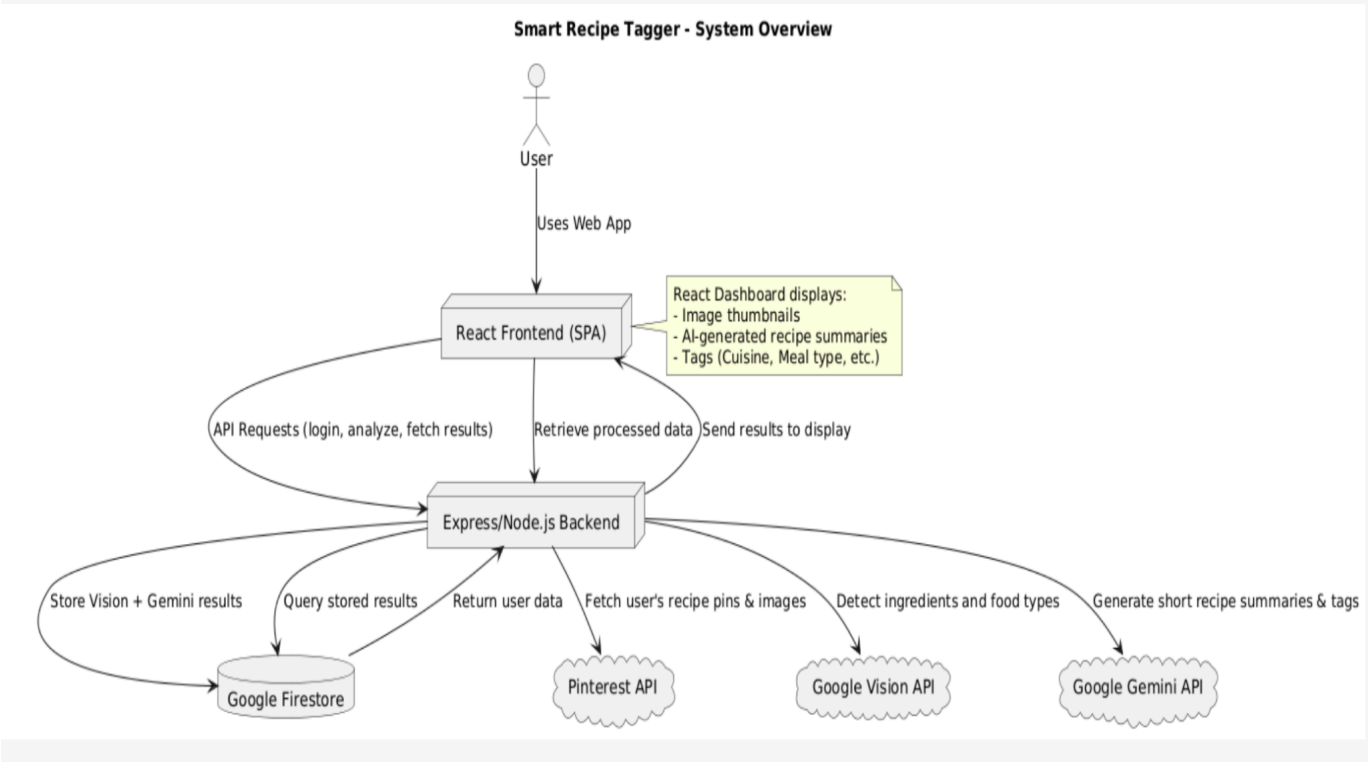

## Recipe Detail View



**Analytics Page**

- Visual charts displaying:
  - Total pins processed
  - Average Vision and Gemini API response times
  - Most frequent cuisine tags
  - Distribution of meal types
  - Processing timeline

## System Diagram




The Smart Recipe Tagger app connects a user's Pinterest food pins to Google's AI services. The React frontend sends requests to a Node/Express backend, which retrieves images from the Pinterest API, analyzes them with Google Vision to detect ingredients, and uses Google Gemini to generate short recipe summaries and tags. All processed data is stored in Google Firestore and displayed back to the user. All API calls are logged in Google Cloud for analytics and monitoring.

### Flow of Control Details

1. User logs in with Pinterest.
2. The app retrieves food-related pins and image URLs for analysis.
3. The backend checks Firestore and skips pins that have already been processed.
   New pins are sent to **Google Vision API** to detect food types, ingredients, and cuisine labels.
4. Vision results are passed to **Google Gemini API**, which generates short recipe summaries and category tags (e.g., Italian, Dessert, Vegetarian).
5. Combined Vision and Gemini results are stored in **Firestore** as `{pinId, imageUrl, visionLabels, geminiSummary, tags}`.
6. The **React Dashboard** retrieves and displays the processed pins with summaries, images, and filters.
7. All API calls and interactions are logged in **Google Cloud Logging** for analytics and performance tracking.

## Cloud Architecture

- **Frontend:** React SPA
- **Backend:** Node.js + Express
- **Database:** Google Firestore
- **APIs:** Pinterest, Google Vision, Google Gemini
- **Deployment:** Google Cloud Run
- **Analytics:** Google Cloud Logging