Food Ingredients Check

(A Streamlit app for analyzing food labels, highlighting risks, and augmenting findings with EU E-code references.)

Svetlana Oshcepkova

EC Utbildning

Python project

1 Sammanfattning

Projektet levererar ett transparent, snabbt och praktiskt stöd för att förstå livsmedelsingredienser och kunna fata beslut om köpet. Riskklassningen är sammanfattad till en egen PDF-policy och använder RAG för strikt, spårbar extraktion. När något saknas i PDF:en fylls rapporten ändå med EU-fakta (E-kod, namn, funktion, policy-ID/URL), med svensk översättning och SQLite-cache med TTL (Time-To-Live) för prestanda. Resultatet är en rapport som är både handlingsbar i butik och lätt att dela.

Sammantaget ger Streamlit appen ett praktiskt och transparent verktyg för att tolka etiketter – med tydlig källprioritet (PDF först, EU-berikning därefter) och låg friktion för användaren.

2 Bakgrund

Detta projekt är ett praktiskt beslutsstöd för vardagen: poängen är att snabbt kunna förstå en livsmedelsprodukts ingredienslista och få en tydlig, motiverad bedömning av vad innehållet kan innebära för hälsan. Idén växte fram under mina kurser där vi gick igenom vanliga ingredienser och tillsatser i mat, skönhets- och städprodukter. Jag planerade först att bygga verktyget för skönhetsprodukter, men bytte fokus till livsmedel av följande skäl: För det första etiketter på mat är oftare konsekventa och lättare att läsa, För det andra man kan hämta E-nummer (E-koder) för livsmedelstillsatser från ett officiellt EU-API (https://developer.datalake.sante.service.ec.europa.eu/api-details#api=228d6fda-9092-4c25-af9a-d537666ed0e5&operation=962248de-9273-4578-9385-e86c2388ea3a).

Som primär kunskapskälla har jag skapat ett PDF-dokument – *food_ingredients_list.pdf* – där kontroversiella ingredienser listas med engelska och svenska namn, E-nummer, vilken kategori de tillhör (t.ex. konserveringsmedel, färgämnen, sötningsmedel) och vilken riskgrad de fått (t.ex. “Undvik”). Detta PDF fungerar som “policy” i systemet: alla riskbedömningar och kategoriseringar i den slutliga rapporten måste komma därifrån. Om en ingrediens inte finns i PDF:en, får den **inte** riskklassas av modellen – i stället berikas den med officiell fakta från EU:s API (namn, funktion och policy-ID) och, när möjligt, en svensk översättning.

3. Metod

3.1 Verktyg och programvara

3.1.1.Gränssnitt

Jag använder **Streamlit** för snabb, interaktiv prototyp med kamera-/filuppladdning, redigerbar text och expanderbara resultatsektioner.

3.1.2 Textigenkänning

OCR (Optical Character Recognition) med Tesseract lokalt, i stället för moln-baserad datorseende. Det undviker kvotbegränsningar och internetberoende, men kan ge sämre råprecision på svåra etiketter; därför finns en redigeringsruta i UI.

3.1.3 Sökindex

FAISS (Facebook AI Similarity Search) med Hugging Face-embeddings sentence-transformers/all-MiniLM-L6-v2. Jag övervägde först embeddings från Gemini, men modellen har gått i deprecation (utfasning) och kvarvarande alternativ hade kvoter som slog i taket även för små datamängder. Hugging Face -embeddings är lokala, snabba och stabila.

3.1.4 RAG-klassificering

RAG (Retrieval-Augmented Generation) med Gemini (2.0/2.5 Flash). En strikt prompt tvingar modellen att svara enbart med JSON, och enbart med kategorier/risktermer som finns i PDF:en. Finns ingen träff i PDF:en: source: "NotInPDF", risk: "Unknown".

3.1.5 E-kod berikning

Officiellt EU-endpoint (API) food_additives_details används för E-nummer. Vi testar flera skrivsätt (”E 414”, ”E414”, ”E-414”) och föredrar poster av typen substanceFAD (enskilda ämnen, inte generiska grupper). Fälten (officiellt namn/funktion + policy-ID/URL) cachas i **SQLite** och den engelska benämningen översätts sedan till svenska med **Gemini AI**.
Cache & hållbarhet: EU-svar lagras med TTL (Time-To-Live) så att äldre poster automatiskt uppdateras efter en viss tid (t.ex. 180 dagar). FAISS-index sparas på disk och återanvänds mellan körningar; en “Rebuild FAISS index”-knapp i sidopanelen forcerar ominindexering när PDF ändras.

3.2 Databearbetning

3.2.1 pipeline från kamera till rapport

3.2.1.1 *Inmatning & OCR.*

Användaren väljer OCR-språk (eng/swe/båda), fotar etiketten eller laddar upp bild. Tesseract extraherar text. En lätt normalisering görs: gemener, diakritikborttagning, whitespace-komprimering.

3.2.1.2 *Extraktion av ingredienser.*

En regelbaserad parser letar efter rubriker som *ingredients/ingredienser* och delar upp listan på komma, semikolon, punkter, ”and/och”, punkteringssymboler samt parenteser. E-nummer fångas via regex (”E 250”, ”E250”, ”E-250”), normaliseras till en lagringsform (”E250”) och dedupliceras. Den resulterande listan visas för användaren som kan justera texten i en redigeringsruta (kompensationen för OCR-fel).

3.2.1.3 *RAG mot PDF.*

För varje token hämtas relevanta PDF-chunkar via FAISS. Dessa, tillsammans med en strikt RAG-prompt, skickas till Gemini. Modellen måste följa PDF:ens “riskguide”:

- Matchar ingrediensen (namn eller E-kod) i PDF:en → returnera PDF-kategori, PDF-riskterm och citat (“pdf_evidence”).
- Matchas inte → source: "NotInPDF", category: "None", risk: "Unknown", red_flag: false.
  Kort motivering (“reason”) kan inkluderas, men riskklassningen får aldrig komma utanför PDF-termerna.

3.2.1.4 *EU-beräkning för NotInPDF.*

Om en ingrediens saknas i PDF-listan försöker systemet hämta officiell data från EU:

- Vi testar flera query-varianter för E-kod (”E 414”, ”E414”, ”E-414”).
- Vi väljer helst substanceFAD-rader (undviker “Group I/II/…”).
- Vi sparar eu_official_name_en, eu_function_en, eu_policy_item_id och eu_fip_url.
- Engelska fält översätts till svenska med Gemini. För att undvika skräp ersätts platshållare (t.ex. “Namn”, “Funktion”, eller <svenska namnet>) med tomma strängar. Därefter appliceras deterministiska fallbacks för kända E-koder (t.ex. E414 → “Gummi arabicum (akaciagummi)”, E903 → “Karnaubavax”) samt en enkel kartläggning *function_en* → *function_sv* (t.ex. ”Sweetener” → ”Sötningsmedel”).
- Resultatet cachas i SQLite med tidsstämpel; TTL gör att uppgifterna förnyas automatiskt efter ett antal dagar.

3.2.1.5 *Rendering & export.*

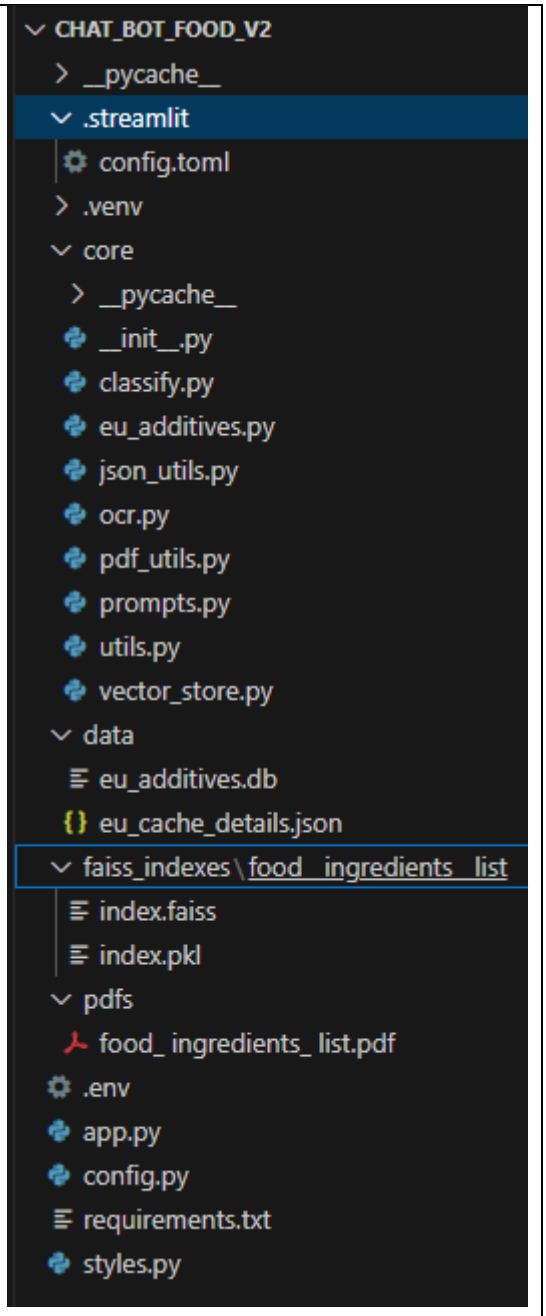
Rapporten visas i två block:

- Kontroversiella ingredienser: allt som är “Undvik” (eller motsvarande högsta risk) lyfts, med kort motivering och PDF-citat.
- Full nedbrytning per kategori: varje ingrediens presenteras med källa (PDF/NotInPDF), risk och EU-rad (E-kod, namn SV/EN, funktion SV/EN, policy-ID). Export finns som JSON och CSV.

4 Arkitektur

3.1 Viktiga moduler och ansvar

- app.py – UI och flöde: OCR-val, kamera/uppladdning, redigering, analysknapp, rendering, nedladdningar och sidopanel (Quick E-code lookup, förladda koder till DB, visa senaste DB-rader, "Rebuild FAISS index").
- config.py – Sökvägar (PDF, FAISS-mapp), konstanter (kategorilista), nycklar. Absoluta sökvägar används för robusthet i Windows-miljöer.
- core/pdf_utils.py – PDF-läsning (PyPDF2) och textsplit i överlappande chunkar för RAG.
- core/vector_store.py – Skapar/laddar FAISS med Hugging Face-embeddings, kontrollerar att både index.faiss och metadata finns, samt exponerar similarity_search.
- core/ocr.py – Tesseract-anrop, normalisering och enkel ingrediensparser som fångar E-koder.
- core/prompts.py – Strikt JSON-prompt för RAG-klassificering (PDF-termer, PDF-risk, evidenscitat).
- core/json_utils.py – "Hård" JSON-parser som kan rädda svar som innehåller kodstaket/kommentarer/utf-tecken.
- core/classify.py – Binder ihop retrieval + prompt + LLM-körning + strikt JSON-parsning.
- core/eu_additives.py – EU-API-klient med multi-variant E-kodfrågor, preferens för substanceFAD, svensk översättning (Gemini), placeholder-detektor, deterministiska fallbacks och SQLite-cache med TTL.



4.2 Hur man kör

1. Installera Tesseract (systempaket). På Windows kan sökvägen behöva anges i core/ocr.py.
2. Skapa virtuellt Python-miljö och installera beroenden via requirements.txt.
3. Lägg .env (rekommenderat) med GOOGLE_API_KEY för att aktivera RAG-klassificering och svensk översättning av EU-fält. Utan nyckel fungerar EU-slagningen (engelska namn), men utan översättning.
4. Kör: *streamlit run app.py.* Första körningen bygger FAISS-index, sedan återanvänds det. Vid PDF-byte: använd "Rebuild FAISS index".
5. Flöde i UI:
    - Välj OCR-språk.
    - Fota etiketten eller ladda upp bild.
    - Kontrollera/justera ingredienslistan i redigeringsrutan.
    - Klicka Analyze & Create Report.
    - Använd sidopanelen för EU-slagningar, förladdning till SQLite, och DB-insyn.

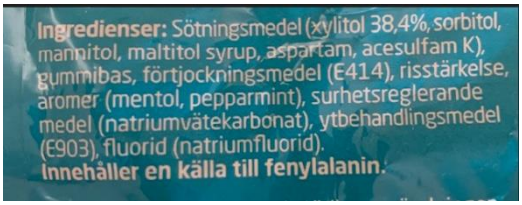
Om filbevakning krånglar (vissa PyTorch-paket): skapas .streamlit/config.toml och fileWatcherType = "poll"

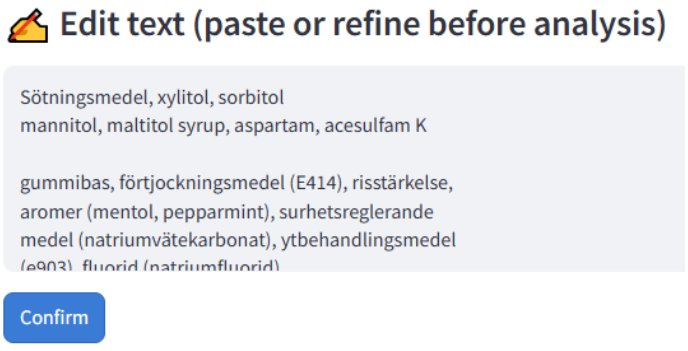
5. Resultat

5.1 Vad levererar appen?

1. Spårbar riskbedömning. All riskklassning kommer från PDF-policyn. Varje "Undvik" kan motiveras med ett kort citat ("pdf_evidence").
2. Meningsfulla svar även vid "okänt". Saknas en ingrediens i PDF:en blir den inte riskklassad; i stället visas EU-fakta: E-kod, officiellt namn, funktionsklass och policy-ID/URL – plus svensk översättning när tillgänglig. Användaren får alltså ändå nyttig information utan att vi gissar risk.
3. Praktisk svenska, inte platshållare. Översättningen rensas från placeholder-fraser (t.ex. "Namn", "Funktion"). För frekventa E-koder finns fasta svenska namn (E903/E414/E300/E967) och en kartläggning av vanliga funktioner (t.ex. "Sweetener" → "Sötningsmedel").
4. Prestanda & robusthet. FAISS byggs en gång och återanvänds. EU-uppslag cachas i SQLite och uppdateras efter TTL (Time-To-Live). Hugging Face-embeddings lokalt ger snabbhet och inga deprecation-/kvotproblem.
5. vExport & delbarhet. Resultat kan laddas ned i JSON och CSV för vidare analys, dokumentation eller delning.
6.

5.2 App input example





5.3 App output example

aspartam (E951) — 🛑 Avoid

- Why it's flagged: Aspartame is an artificial sweetener listed under 'Avoid'.
- Backed by PDF: "Aspartame (E951) / Aspartam (E951)"

Classification uses ONLY the PDF. Items marked NotInPDF had no supporting match in the document (EN/SV name or

Analyzed items: **21**

Sweeteners (4)

sorbitol *(E-code: E420)* — ⚠ Lower risk — 🔷 *PDF*

- Reason: Sorbitol is listed as a lower risk sweetener.
- PDF evidence: "Sorbitol (E420) / Sorbitol (E420)"

mannitol *(E-code: E421)* — ⚠ Lower risk — 🔷 *PDF*

- Reason: Mannitol is listed as a lower risk sweetener.
- PDF evidence: "Mannitol (E421) / Mannitol (E421)"

aspartam *(E-code: E951)* — 🛑 Avoid — 🔷 *PDF*

- Reason: Aspartame is an artificial sweetener listed under 'Avoid'.
- PDF evidence: "Aspartame (E951) / Aspartam (E951)"

acesulfam *(E-code: E950)* — 🛑 Avoid — 🔷 *PDF*

- Reason: Acesulfame K is an artificial sweetener listed under 'Avoid'.

None (17)

sotningsmedel — ❓ Unknown — ◯ *NotInPDF*

- Reason: The term 'sötningsmedel' (sweeteners) is a category heading in the PDF,
- EU Additives (official): *no match*

xylitol — ❓ Unknown — ◯ *NotInPDF*

- Reason: Ingredient not found in the provided PDF guide.
- EU Additives (official): **E967** — Xylitol — function: Sötningsmedel — id: 359641

maltitol syrup — ❓ Unknown — ◯ *NotInPDF*

- Reason: Ingredient not found in the provided PDF guide.
- EU Additives (official): *no match*

gummibas — ❓ Unknown — ◯ *NotInPDF*

- Reason: Ingredient not found in the provided PDF guide.
- EU Additives (official): *no match*

6 Begränsningar & lärdomar

6.1 OCR-kvalitet:

Lokal Tesseract är resurssnål och privat, men kan missa text på blanka/böjda ytor eller vid låg kontrast. Redigeringsrutan är en medveten “safety-net”. En framtida bildförbehandling (deskew, binarisering, kontrast) skulle hjälpa.

6.2 RAG-disciplin vs täckning:

Att strikt låsa klassificeringen till PDF:en ger hög tillförlitlighet, men betyder också att helt nya ingredienser inte får någon risk. Det är ett medvetet val för att undvika felaktiga antaganden.

6.3 EU-fältens variation:

Funktionsfält saknas ibland i EU-svaret. Därför krävs översättning + heuristik + fallbacks.

6.4 Kvoter och modellval

Deprecation av tidigare embeddings tvingade fram Hugging Face-lösningen. Att kombinera Gemini 2.0/2.5 Flash för olika uppgifter minskar risken att slå i samma kvot under testning.

7. Möjliga förbättringar

7.1 Smartare OCR-pipeline

Förbehandling (skärpa/kontrast/avsmetning), enkel layoutdetektion (skilja ingredienslista från näringsdeklaration), och bättre hantering av parenteser/procent/allergener.

7.2 Fler källor

Utöver min PDF och EU-API:t kan nationella rekommendationer och vetenskapliga översikter integreras. UI kan visa källor per rad, så att varje uppgift är klickbar/granskbar.

7.3 Forcerad E-cod-refresh

Knapp för att radera specifika E-koder ur SQLite och hämta om; batch-förvärmning av vanliga E-nummer vid start; schemalagd uppfräschning (t.ex. veckovis).

7.4 Offline-läge

Lättviktig LLM (Large Language Model) lokalt för förklaringar/översättning när nät saknas.