

Tech Basics II - Stream A
Course Code: 42028000
Instructor: Sarah Haq
Leuphana Universität Lüneburg
Winter Semester 2024/2025

Student ID: 4000960
Name: Denise Burmester

Link: <https://veganornot.streamlit.app/>

Report: VEGANSCAN

Table of Contents

1. Abstract	2
2. Introduction	2
3. Development Process	2-3
4. Challenges and Limitations	3-4
5. Conclusion	4
6. References	4-5

Abstract

This report describes the development of the *VEGANSCAN* application, a Python-based web app created with Streamlit. The app allows users to scan barcodes of food products to determine whether they are vegan or vegetarian, utilizing the OpenFoodFacts API. The report provides an overview of the development process, including the technical setup, challenges encountered, and solutions implemented. The report also addresses the use of AI tools and concludes with insights on the learning process and the app's potential for further development.

Introduction

In the *Tech Basics I* course, I originally planned to develop a game with gamification elements, which was intended to be more of a shopping app called *Better Choices*. However, I was dissatisfied with my project and felt that I could achieve more and produce better results if I took the time to work independently, practice, and research solutions online. As a result, I decided to abandon my initial idea from *Tech Basics I* and focus on something entirely new for *Tech Basics II*.

I ultimately chose to develop *VEGANSCAN* because I knew that I would soon be moving to Hong Kong for my studies. Being able to quickly and accurately determine whether a product is vegan could be useful when living in a different country. Since I have been vegan for many years, this topic is personally very important to me. My goal was to create a meaningful and practical application that could genuinely help people make informed, ethical food choices.

Beyond personal motivation, this project also aligns with broader themes of sustainability and environmental consciousness. By making it easier for consumers to identify vegan products, the app promotes more sustainable consumption habits and contributes to a more eco-friendly world. Developing *VEGANSCAN* has given me the opportunity to combine my personal values with my improving and growing coding skills while working toward a project with real-world impact.

Development Process

The development started with setting up the basic Streamlit interface and integrating the OpenFoodFacts API. Instead of manually handling HTTP requests, I decided to use a dedicated Python package to manage API queries, as this approach was simpler and more efficient for my

use case. The focus was on creating a minimum viable product (MVP) that delivers the core functionality effectively.




A crucial technical aspect was barcode detection. OpenCV was used to capture images from the camera and extract barcodes using *BarcodeDetector*. It was essential to ensure that the captured images were converted into a compatible format for accurate processing. To make the app more visually appealing and user-friendly, I also used AI tools to generate a banner displaying the question "**Is it vegan or not?**" alongside images of vegetables. Additionally, I customized the background and buttons to be green, aligning with the theme of veganism and sustainable food choices.

To store scanned products, a session state variable was implemented, allowing users to maintain a list of scanned items and review them later. This was particularly important because interacting with the camera resets certain variables, which would otherwise lead to the loss of scanned product data. By using session state, I ensured that information remained persistent even when switching between interactions.

Challenges and Limitations

Several challenges arose during development. One of the main issues was barcode recognition, as OpenCV did not always reliably detect all barcodes. The quality of the camera capture significantly impacted the recognition rate. To improve usability, I provided clear user instructions on how to position the barcode for optimal results.

Additionally, the OpenFoodFacts database did not always contain complete product information. In such cases, I implemented an error message to notify users when data was missing or unclear. To enhance the user experience, I introduced color-coded labels:

-  **Green** for "vegan"
-  **Orange** for "uncertain"
-  **Red** for "not vegan" or "missing data"

Another difficulty was managing the list of stored products. Initially, users could not remove items from the list. After researching different approaches, I decided to implement a delete button using callbacks, as this was the simplest and most effective solution. This allows users

to remove individual products or clear the entire list. Understanding how buttons and callbacks work in Streamlit required additional research in the framework's documentation.

As I am still developing my programming skills, another challenge I faced was understanding certain coding concepts, especially how Streamlit handles state management and API requests. To overcome these difficulties, I conducted additional research online, using resources such as the Streamlit documentation, OpenCV tutorials, and Python API guides. I also used AI assistance, including ChatGPT, to clarify complex concepts and troubleshoot issues. ChatGPT provided general guidance and explanations on debugging errors, but the idea and all implementation decisions and modifications were made independently.

Conclusion

The *VEGANSCAN* application combines modern image processing with API-based data retrieval, offering a convenient tool for verifying vegan and vegetarian food options. The development process involved overcoming technical challenges, particularly in barcode detection and handling incomplete data. However, the result was a functional and user-friendly application that can serve as a foundation for further enhancements.

Through this project, I gained valuable experience working with Streamlit, OpenCV, and APIs. Additionally, researching solutions online and using AI tools for explanations played a role in my learning process, helping me to understand key programming concepts more effectively.

References

ChatGPT. (2025). Personal insights on troubleshooting and learning programming concepts in Python and Streamlit. Unpublished report.

CodeWithHarry. (n.d.). *How to use buttons in Streamlit*. YouTube. Retrieved from <https://www.youtube.com/c/CodeWithHarry>

Data Professor. (n.d.). *Streamlit session state tutorial*. YouTube. Retrieved from <https://www.youtube.com/c/DataProfessor>

OpenCV. (n.d.). *Barcode scanner*. Stack Overflow. Retrieved from <https://stackoverflow.com/questions/17170752/python-opencv-load-image-from-byte-string>

OpenFoodFacts. (n.d.). *OpenFoodFacts database*. Retrieved from <https://world.openfoodfacts.org/data>

ProgrammingKnowledge. (n.d.). *How to use APIs in Python*. YouTube. Retrieved from <https://www.youtube.com/c/ProgrammingKnowledge>

PyImageSearch. (n.d.). *Barcode detection with OpenCV*. YouTube. Retrieved from <https://www.youtube.com/c/PyImageSearch>

Streamlit Docs. (n.d.). *Theming*. Retrieved from <https://docs.streamlit.io/develop/concepts/configuration/theming>

Streamlit Docs. (n.d.). *Session State*. Retrieved from https://docs.streamlit.io/develop/api-reference/caching-and-state/st.session_state

Streamlit Docs. (n.d.). *st.columns*. Retrieved from <https://docs.streamlit.io/develop/api-reference/layout/st.columns>

Streamlit Docs. (n.d.). *Button behavior and examples*. Retrieved from <https://docs.streamlit.io/develop/concepts/design/buttons>

Tech with Tim. (n.d.). *Streamlit tutorial for beginners - Python web apps*. YouTube. Retrieved from <https://www.youtube.com/c/TechWithTim>