ENGINEERING OF AI INTENSIVE SYSTEMS
SUMMER TERM 2024

SUPERMARKET SHELF SCANNER

System Goal:
Assist supermarket employees:
Detect & count number of remaining products on supermarket shelf

Prototyping Scenario:
Detect & count remaining soda can (Coca Cola, Fanta, Sprite) on shelf

TEAM MEMBERS

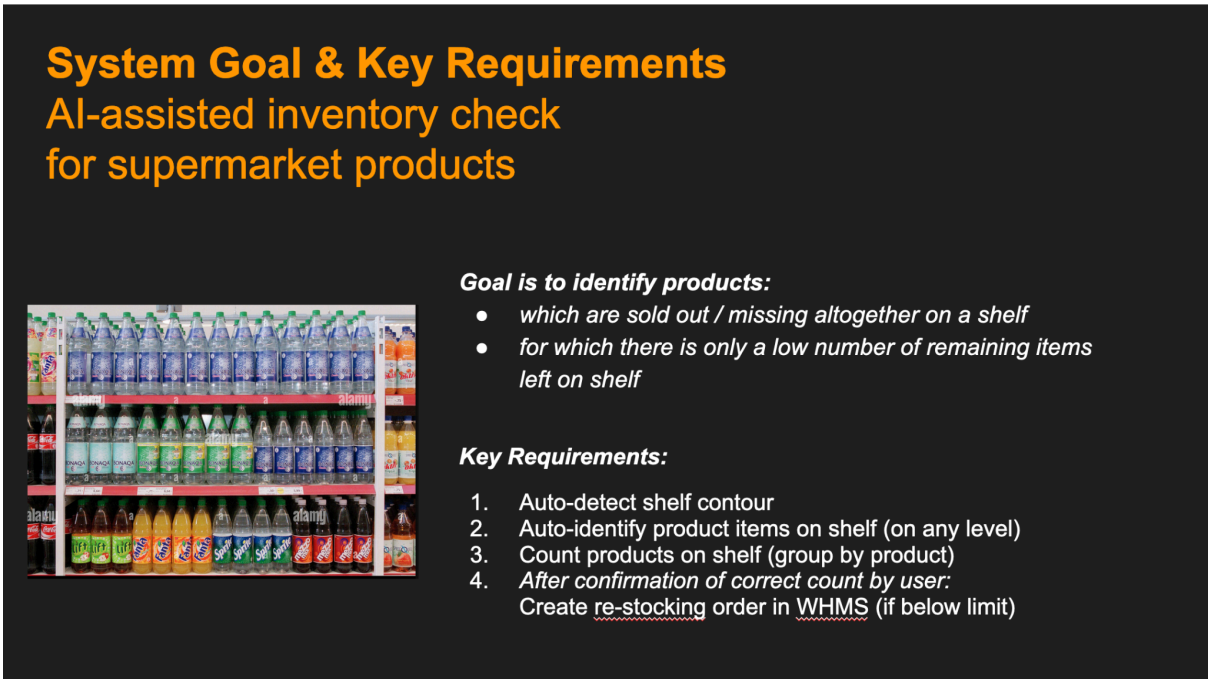
- Amir Suleimenov
  Student ID: k12247291
  eMail: amirdragon0808@gmail.com
  Field of study: Computer Science (Master)

- Michael Schifferdecker
  Student ID: k11937196
  eMail: michael.schifferdecker@posteo.de
  Field of study: Artificial Intelligence (Master)

TABLE OF CONTENTS

High-Level Project Summary

Project Summary: Q&A



Q0. How would you summarise your project in a one-liner?

(Prototype for) AI-assisted inventory check for supermarket shelves

Q1. What is the domain of your system?

Retail / Supermarket

Q2. Who would be the users of your system?

Store Manager and/or Warehouse Manager

Q3. What is the main goal of your system?

Identify shelves which need to be restocked due to customers having all items picked from them already and the respective shelve being almost empty.

After detection of an almost empty shelf this may then trigger an internal notification / alert to the supermarkt store's staff to replenish the stock on the particular shelf. Additionally we may also automatically create an internal order in the distribution centre/central warehouse to ship more of the currently sold-out products to the store with the next re-supply delivery.

Q4. How would your system achieve its goal?

We aim to analyse static images (and alternatively in an optional build-out stage of our product live video stream) of a fictitious supermarket shelf to count the remaining products on the shelf.

Assuming that we are looking at a shelf on which soda cans (Coca Cola, Fanta and Sprite) are stocked we would like to count

1. the total number of remaining soda cans on the shelf as well as
2. the breakdown of this total number of soda cans into
    a. number of remaining Coca Cola cans on the shelf,
    b. number of remaining Fanta cans on the shelf and
    c. number of remaining Sprite cans on the shelf

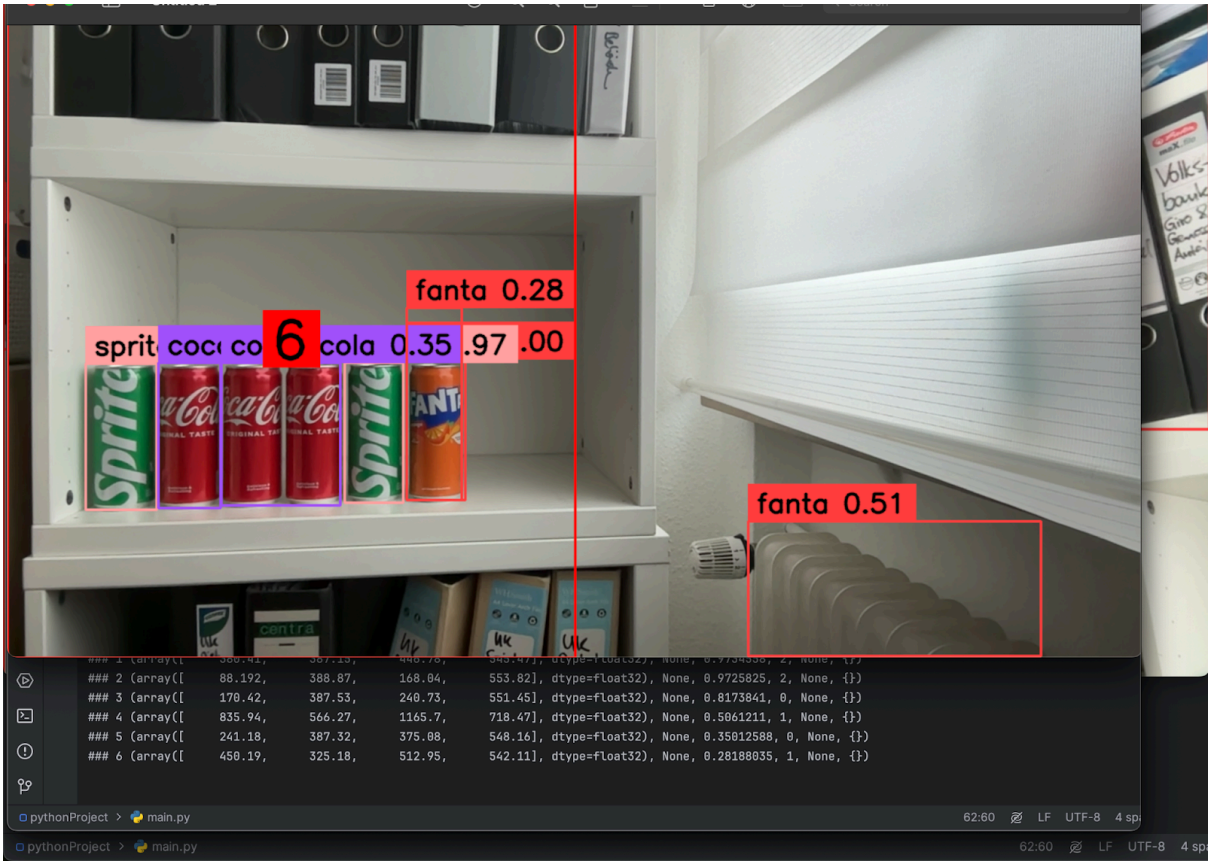
Together with a threshold defined as minimum number of items a particular product on that particular shelf, the system may then

1. trigger notifications to the store team (e.g. via email / push notification onto their mobile phone or similar) and/or
2. automatic re-ordering actions (against a fictitious API of the central distribution warehouse system or similar).

Q5. Which type of AI/ML strategy would be required/useful and why?

1. use pre-trained object classification libraries such as Ultralytics Yolo to identify objects (e.g. soda cans) in an image + to mark them with bounding boxes
2. export of the found objects / cropped images of their bounding boxes
3. feed the cropped images of objects into a product classification module to classify a (generic) soda can into a (specific) Coca Cola, Fanta or Sprite soda can. This module may be based on a simple kNN classification which uses a set of reference images of a Coca Cola, Fanta or Sprita soda can to determine the type of soda can (Coca Cola, Fanta or Sprite). This means that we will probably use a rather simple similarity measure to perform the concrete product classification (generic soda can => Coca Cola, Fanta or Sprite soda can).
4. count the Identified objects (number of soda cans in total and number of soda cans differentiated by product type)
5. *(feed the result of the object count step to the alert/notification module which then may trigger emails / orders in the central distribution warehouse etc.)*

Intended Prototype: Example of Soda Can Detection & Count



Example of live soda can detection using a custom-trained Yolo v8 model

There are a total of 6 soda cans inside the left half of the picture which is the (counting) area of interest and which is marked by a big red rectangular.

Note that the wrongly identified fanta soda can (which is in fact the radiator) is not contributing to the count of a total of 6 soda cans since it is placed outside of the counting area.

Note:

- We are suggesting a simplified product classification logic in step #3 since we currently doubt that it will be easy to adapt the training of the Ultralytics Yolo (or similar libraries) easily and within a reasonable time frame. We may look into this in case the implementation of the MVP progresses faster than anticipated.
- Additionally we would like to limit us to analysis of static images for the initial MVP version of our product. If the implementation of this initial MVP goes smoothly then we may very well attempt to switch to live video stream analysis and instant display of product count figures in/alongside the live video stream that we capture.

Requirements

Summary System Description

(Prototype for) AI-assisted inventory check for supermarket shelves.

The system shall help to identify products which are missing on a shelf or for which there is only a low number of remaining items left and which therefore need be restocked - ideally before customers have also picked the last (few) remaining items picked from them.

Note: in our prototype this will most likely LIMIT ourselves to counting soda cans on shelf, i.e. counting the number of Coca Cola vs. Fanta vs. Sprite cans => if we talk about "objects" or "products" below then we are implicitly talking about these kind of soda cans…

System Goal Description

Non-AI goals

- Spare the employees in a store the manual categorisation and counting of remaining products on store shelves
- Speed up the detection of missing and/or low number of remaining products significantly without unreasonable loss of accuracy (type and number of remaining products on shelf)

AI related goals

- Automatic categorisation of remaining products on shelf
  => AI-based image processing / object classification algorithm
- Automatic counting of remaining products on shelf
  => AI-based image processing / object counting algorithm

Stakeholder Identification

Users

a. Store managers
b. Salesman

People affected by the system

c. Customers and visitors of stores. As they will have broader choice with more quantities
d. Store managers and salesmans. As they will more easier manage product fulfilment
e. Stock and logistic managers. As they will more better understand cycles of product replenishment
f. Store owners, companies. As it could cause more selled products and then more revenue.

Managers (those are probably the instructors)

g. Theoretically store managers or Head of store departments
h. Educationally Professor Alexander Egyed and Doctor Luciano Marchezan de paula

Regulators

i. AI Act of European AI Office
j. Austrian municipal office which regulates on local level supervision of AI Act in Upper Austria

Functional requirements (EARS Template)

List of functional requirements

Image Upload

- **FR100:** The system shall allow for uploading of store shelf images and specification of a shelf identifier for the purpose of classifying and counting the remaining products on a particular shelf.

Object Detection

- **FR150:** The system shall automatically detect the shelf area in the uploaded picture
- **FR151**: The system shall apply image processing techniques to recognize and differentiate between different product given an image of a particular shelf

Counting Algorithm

- **FR201**: The system shall develop a counting algorithm to determine the number of distinct products identified on the shelf which the image has been taken
- **FR202**: If the system wrongly detects an object which is outside the borders of the shelf, then this object shall be excluded from the object count.

Real-Time Processing

- **FR230**: The system shall optimise the program for real-time or near real-time processing of shelf images to facilitate timely inventory management

Reporting

- **FR250**: The system shall generate reports summarising the number and types of products detected on the shelf.
- **FR251 (optional for implementation):** The type and number of distinct products that have been counted on the shelf shall be reported back to the user who has taken & uploaded the image of the shelf

Integration

- **FR270**: The system shall enable integration with existing inventory management systems for seamless data transfer and process of product count information.
- **FR271:** Upon submission of product count information, the inventory management system shall automatically create a re-stocking order if the number of remaining products is below the minimum number of products that are supposed to be available in the store.

Non-Functional Requirements (EARS Template)

List of nonfunctional requirements

External Interfaces (MVP)

- **NFR101:** The system shall be able to take in static images of product shelves into the image processing server via an upload interface that can be used over the web (e.g. by offering a REST- or WebService-based endpoint)
- **NFR102:** The system shall be able to pass on the number of counted products on a particular shelf to an inventory management system via an interface that can be used over the web (e.g. REST or WebServer-based protocol)
- **NFR103:** For the purpose of expedited development and testing of the system the system shall implement a mocked version of the inventory management system.

Performance

- **NFR200:** The classification and counting of remaining products on a particular shelf based on a static image of the shelf shall take no longer than 7 seconds.

Attributes

- **NFR300:** The uploaded images need to be RGB colour images
- **NFR301**: The images should have min. resolution as 96 DPI
- **NFR302**: The image that arrives on the server side for processing may not exceed 5 MB of size (PNG image => uncompressed)
- **NFR305:** The image may be compressed during upload but only to the extent that compression does not introduce artifacts which are detrimental to object detection and counting (we are still lacking good measure for this => needs to be refined later)
- **NFR360:** The system may refuse processing the image if the shelf area covers less of 75% in the image (measured both horizontally and vertically)
- **NFR361**: The system may refuse processing the image if the image is too dark (we lacking good statistical + implementable measure still => needs to be refined later)
- **NFR362:** The system may refuse processing the image if the image is blurred (we currently lacking a good statistical + implementable measure still => needs to be refined later)

Constraints

- **NFR380:** The system shall allow for counting objects on the shelf if the image is taken at an angle of +- 15 degrees deviation from a fully parallel & central position standing before the shelf
- **NFR381:** The system shall allow for counting objects on the shelf if the image is taken with a tilt of +- 15 degrees deviation from a fully parallel & horizontally aligned position standing before the shelf.
- **NFR382:** In order to contribute to the object count the AI object detection / product classification model needs to detect/classify a product on the shelf with more than 50% probability. Otherwise it will be considered a wrong detection & will be excluded from the count.
- **NFR383:** The system shall not have to identify nor does it have to count the product if it overlaps with another product by more than 20% of its size.

Security

- **NFR500:** The system shall only be usable by members of the supermarket trading organisation (authentication)
- **NFR510**: The functionalities of the system shall only be usable by the members of the supermarket trading organisation if they have been permissioned for the respective functionality beforehand (authorisation)
- **NFR520**: The system shall use the existing single-sign-on facility in order to allow for seamless integration with the inventory managements system and usage of the corresponding API (single-sign on)

Important note:

Once the prototyping proceeds more concrete model performance KPIs (precision, recall, F1-score) for the given prototype shelfs / products have to be agreed upon => further refinement of requirements on this end necessary as part of the initial more technical design phase.

AI-related requirements

List of AI-related requirements

FR150 (shelf are), FR151 (object detection), FR201+FR202 (object count)
FR230 (realtime detection & count), FR250+FR251 (report of object count)

NFR200 (max. time for object detect & count),
NFR300+NFR301+NFR302 (image type, resolution, file size)
NFR305 (limit artifacts from image compression),
NFR360+NFR361+NFR362 (image quality),

FR150+FR151, FR201+FR201:
for our prototyp we currently plan to train a Yolo object detection model trained with images taken of a prototype shelf and Coca Cola / Fanta / Sprite cans.

FR230, NFR200: a max. time of 7 secs for classifying and counting of images should be reasonable and achievable given the performance of similar custom-trained Yolo models. Such a still relatively short time span is necessary to also have the option of feeding back the results of the image analysis to the user that has taken the image of the shelf so that he/she is able to check the product classification and count right after the upload of the shelf image and without slowing him or her down from checking the next shelf out.

NFR300-302, NFR360-362, NFR380-383: These NFRs need to be considered when building the training, test and validation set for the model. We currently plan to use standard data augmentation techniques in order to generate images with non-parallel and tilted camera poses so that our prediction results become more robust based on training on a more realistic dataset. See https://blog.roboflow.com/image-augmentation/ for more details.

Important Note: it is difficult to state expectations with regards to accuracy / performance KPIs that the object detection + counting algorithm / AI should have (such as precision, recall, F1 score etc.). For this we will have to separately describe the training + prototype application scenario (what shelves + what universe of objects to detect)

Use Cases

Actors

Primary Actors (who use the system in their daily activities)

#	Primary Actor	Type	Description
1	Shop Employee	Human	Responsible for 1. managing stock inventory, 2. detecting empty shelves and/or shelves which are running low w.r.t. number of remaining products on particular shelf 3. re-stocking shelves
2	Store Manager	Human	Interested in getting info / reports about 1. current stock levels on shelves in the store 2. set and number of products which have been re-ordered from central warehouse and which are on the way to the store
4	Image Processing Model	AI	Provides core logic for interpreting images of store shelves incl. 1. Detecting boundaries of invidividual product shelf areas in the image 2. Identifying individual product items on the shelf 3. Counting product items on the shelf => count remaining product items grouped by distinct product

Secondary Actors (enable primary actors to use the system)

Note:

- *we are not required to describe these as part of this assignment*
- *however we still name as few as they spring to mind and in order to practise the differentiation of primary and secondary actors for our system*

#	Secondary Actor	Type	Description
1	IT Manager	Human	Responsible for 1. ensuring that all IT systems which are required by the individual stores and for replenishing the individual stores with goods from the central warehouse are continuously operational and available for daily use

Stakeholders

#	Stakeholder	Description
1	Store Manager	Manages a particular store or a group of stores in a particular region.
2	Store Employee	Works in a particular store. Sits at till in order to perform the checkout / sell products to customers. Responsible to track remaining products on shelves and to issue re-stocking orders. Responsible to transfer palettes from resupply lorry into the stores inventory and then to to re-stock the individual shelves so that customers can again pick them from shelves and eventually buy them at checkout.
3	Customer	Comes to the store, picks product items from the shelves and puts them into his/her shopping cart before proceeding to the tills / checkout in order to purchase the items collected in the shopping cart.
4	Inventory Manager	Manages inventory on the central / regional distribution warehouse level in order to make sure that re-stocking orders from stores can always be fulfilled in a timely manner (within agreed lead time SLAs & ideally with the next daily supply delivery to the store).
5	Logistics Manager	Manages the distribution of products from central / regional distribution warehouse to local stores. Has to ensure that truck capacity is efficiently used and that truck capacity is sufficient for fulfilling the re-stocking orders received from stored in a timely manner (so that products are re-supplied to store within agreed lead time SLA & ideally with the next daily supply delivery to the store)

Descriptions of Individual Use Cases

Reference to USE CASES.XLSX

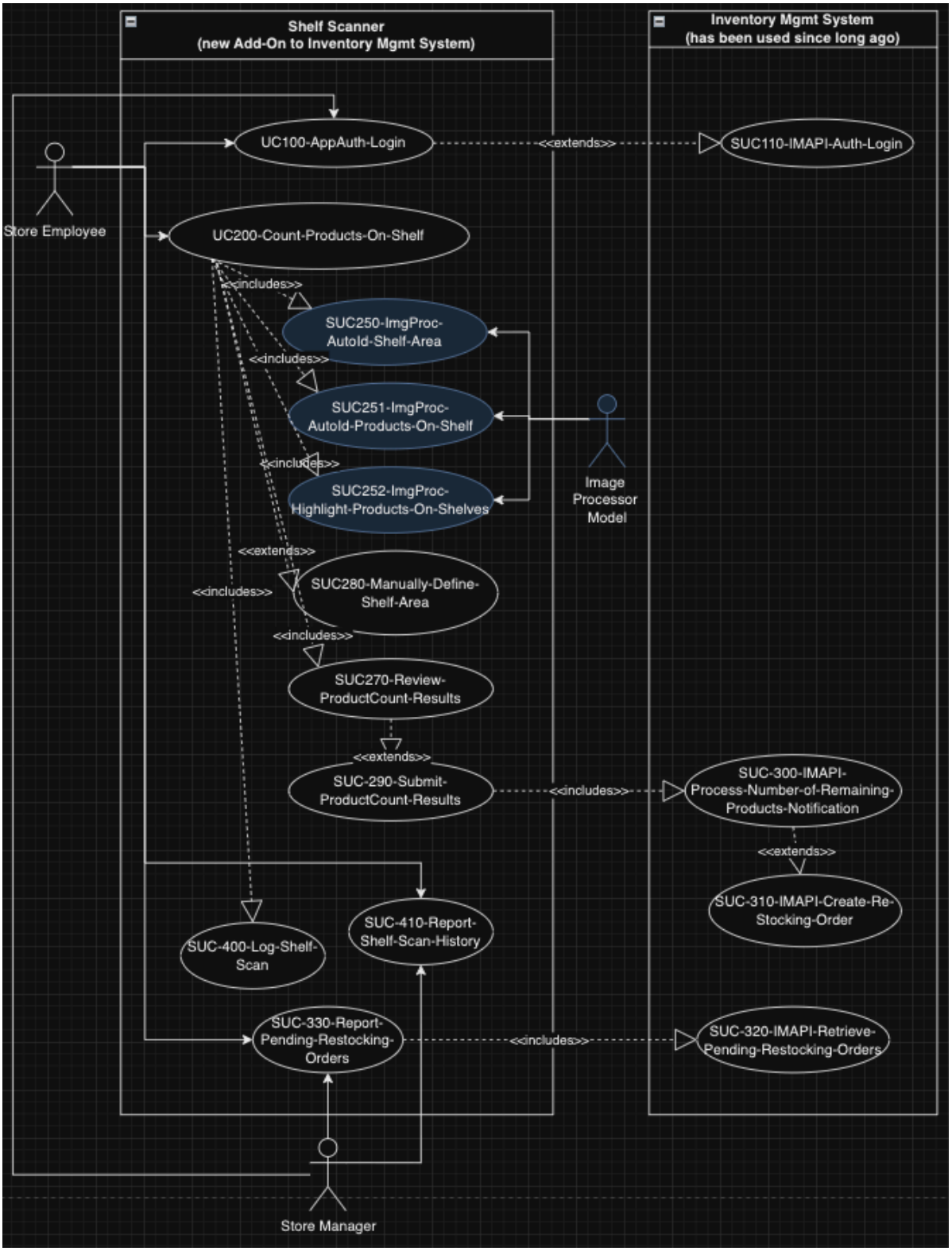
Please see **separate spreadsheet “Use Cases.xlsx”** for details.

Traceability Matrix

Reference to REQUIREMENTS-USECASES-TRACEABILITY.XLSX

Please see **separate spreadsheet “Requirements-UseCases-Traceability.xlsx”** for details.

Use Case Diagram



Implemented Use Cases in Our prototype

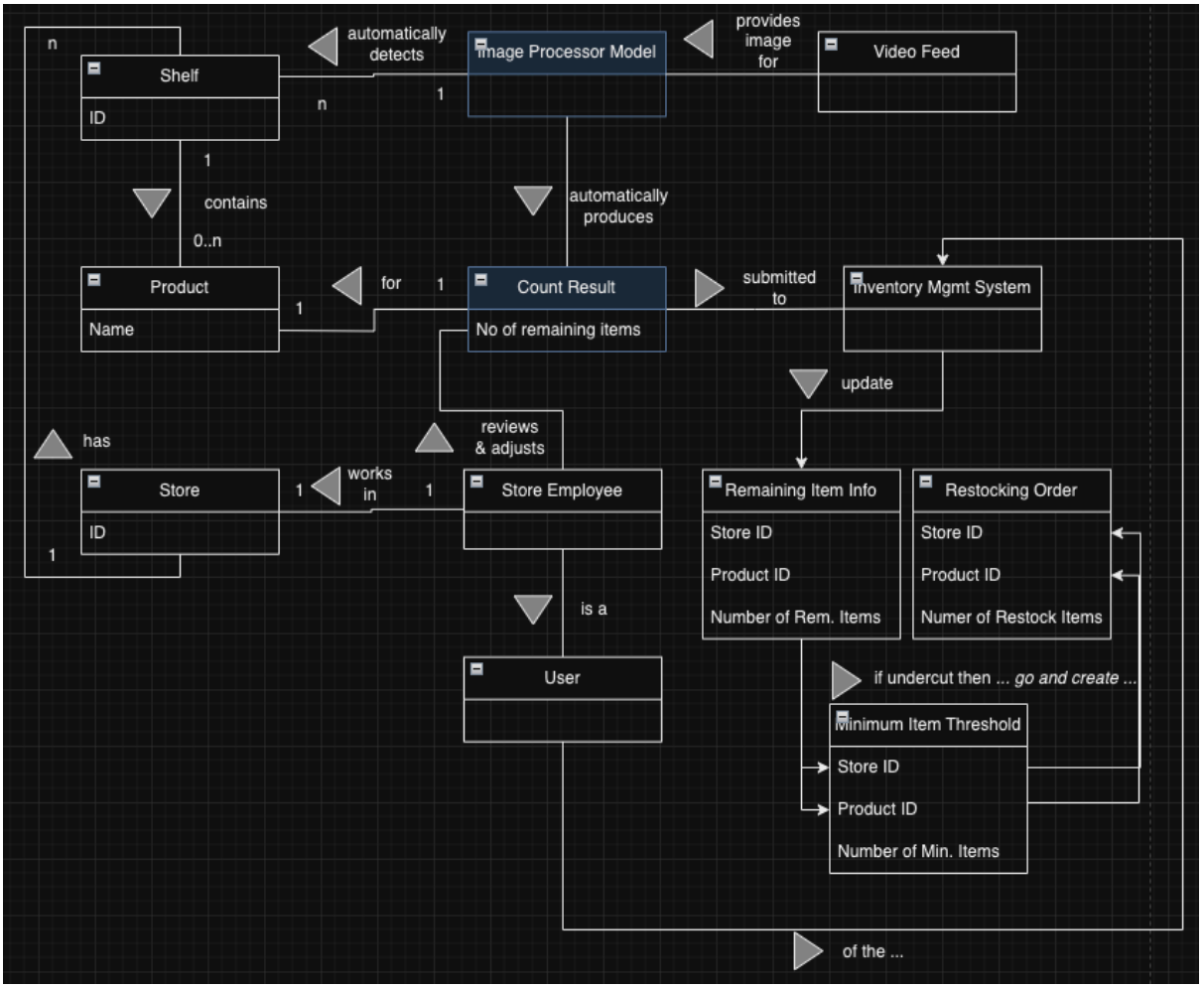
Shelf Scanner System

- UC100
- UC200:
  - SUC 250, SUC 251, SUC 252, SUC 270, SUC 290
  - SUC 400 (rendering messages for each scan, savings in session level)

Inventory Mgmt System

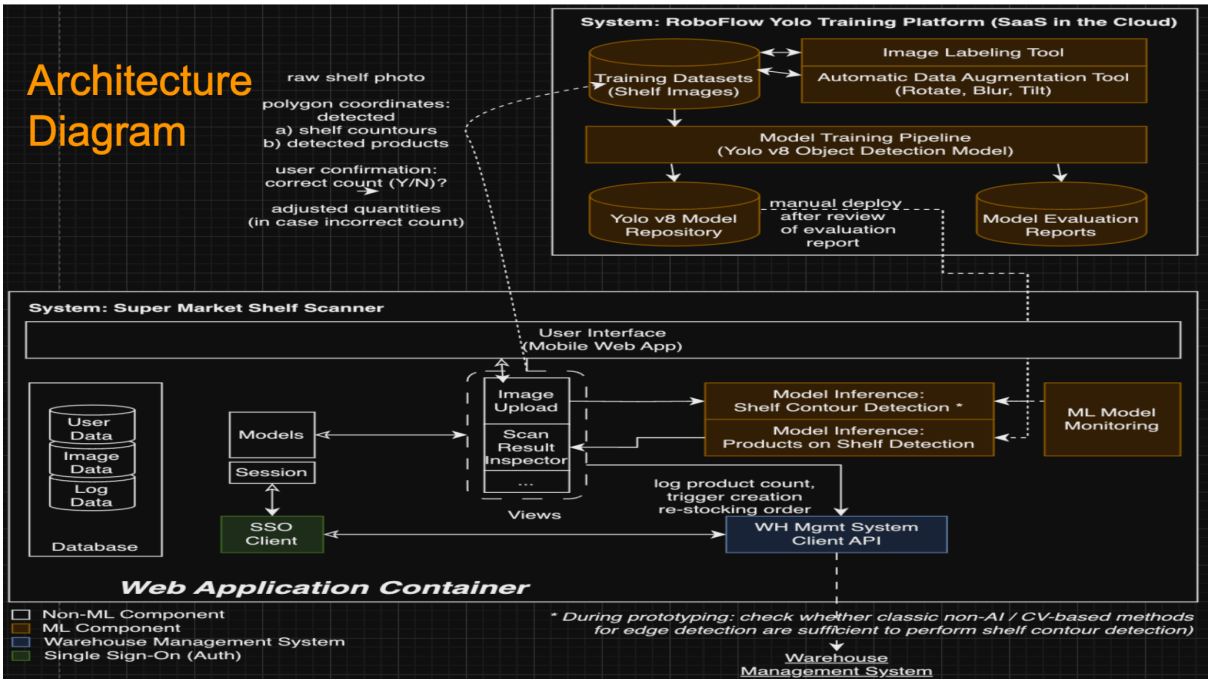
- UC110
- SUC300

Domain Model

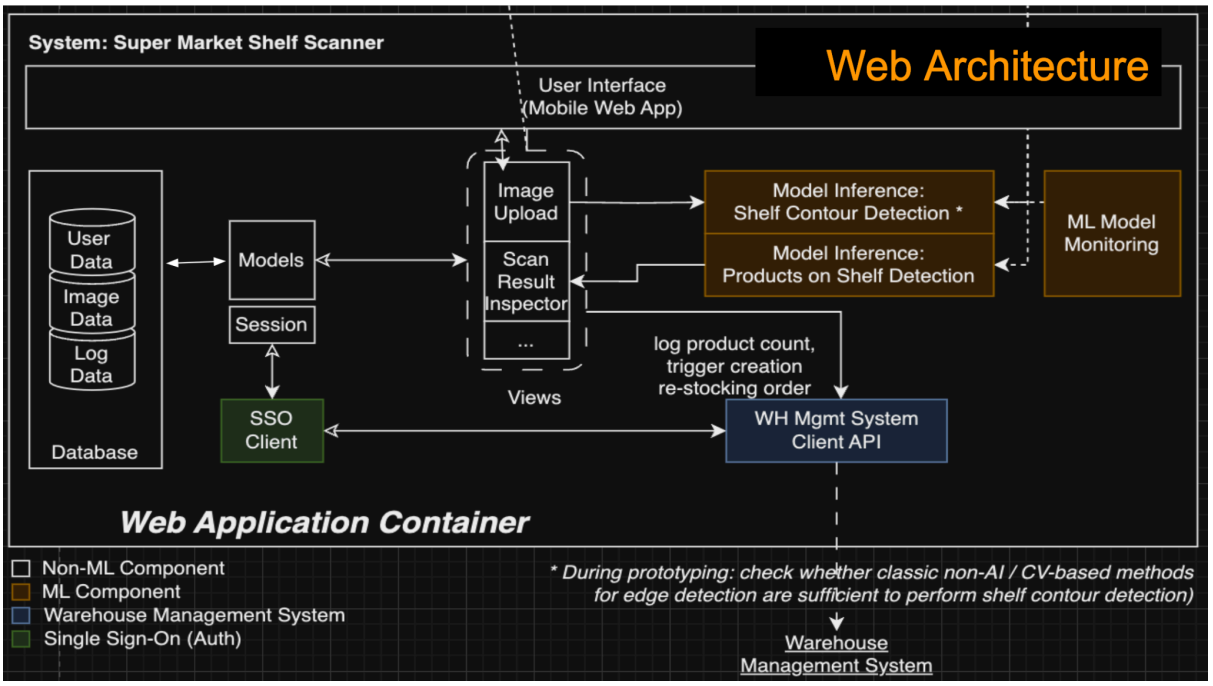


Architecture

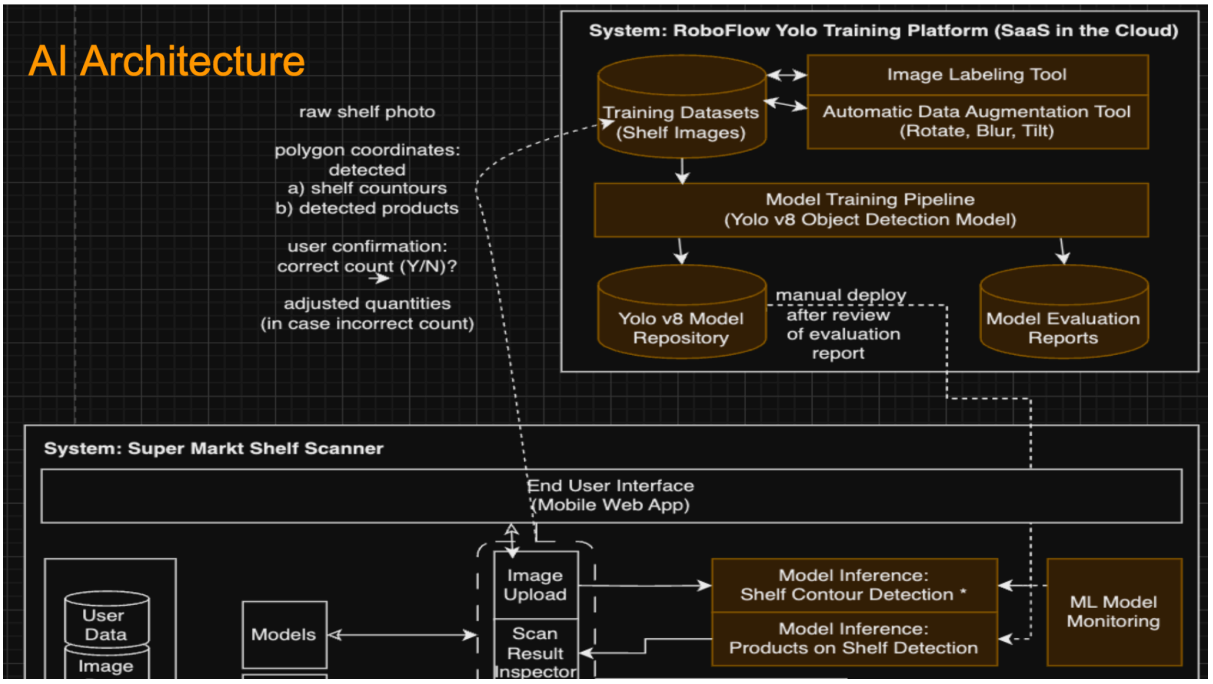
Overview



Web Application



AI


AI Architecture
raw shelf photo
polygon coordinates:
detected
a) shelf countours
b) detected products
user confirmation:
correct count (Y/N)?
adjusted quantities
(in case incorrect count)
System: RoboFlow Yolo Training Platform (SaaS in the Cloud)
Training Datasets
(Shelf Images)
Image Labeling Tool
Automatic Data Augmentation Tool
(Rotate, Blur, Tilt)
Model Training Pipeline
(Yolo v8 Object Detection Model)
Yolo v8 Model
Repository
manual deploy
after review
of evaluation
report
Model Evaluation
Reports
System: Super Markt Shelf Scanner
End User Interface
(Mobile Web App)
User
Data
Image
Models
Image
Upload
Scan
Result
Inspector
Model Inference:
Shelf Contour Detection *
Model Inference:
Products on Shelf Detection
ML Model
Monitoring

Components

COMPONENT	Description	Requirements
Web Application Container	For Django application model it provides a robust runtime environment to host and execute Django web applications. It includes various components and services necessary for deploying, managing, and scaling Django applications.	In order to deliver the functionality in the most flexible way it is a good idea to implement a (mobile-ready cross-platform) web application.
User Interface	It is an interface that provides information transfer between a human user and hardware and software components of app running device	See Web Application Container
Database	It is collection of data relevant to the app.	See Web Application - general app infrastructure. Esp. Requirements for Use Case to log shelf scan (input & output). FR100 FR250
… User Data	Sub database or several tables which store user data as tokens, login, encrypted password etc	FR270 FR271 NFR102 NFR103 NFR520
… Image Data	Sub database which stores raw images as well as the user reviewed and approved images	Esp. Requirements for Use Case to log shelf scan (input & output). FR100 FR250
… Log Data	Log data is the records of all the events occurring in the application	All across the application. Cross cutting infrastructure concern +

		best practive in Software Development. Esp. Requirements for Use Case to log shelf scan (input & output). FR100 FR250
Models	MVC-Pattern => Model	n/a grouping item in architecture
Session	Web Session	Esp. retrieval and session storage of auth token from SSO NFR520
SSO Client	Single-Sign On Client. Both the app and and the WHMS authenticate against the same ID server.	FR270 FR271 NFR102 NFR103 NFR520
Views	MVC-Pattern => View	n/a grouping item in architecture
… Image Upload	View which is used for image upload & to trigger model inference (shelf contour detection and detection of products on shelf (on any layer)	Esp. FR100 and pot. further requirements related to UC200
… Scan Result Inspector	View which is used to displaying the product count results based on the processed shelf image	Esp. FR251 and pot. further requirements related to UC200
WHMS Client API	API to the Warehouse Management System. Required in order to submit product count results.	NFR102 NFR103
Model Inference	AI model inference endpoints	grouping item / details see below
… Shelf Contour	It is for detection of shelf	FR100

Detection	contour in the submitted image	FR150 FR230 NFR101 NFR200 NFR300 NFR301 NFR302 NFR305 NFR360 NFR361 NFR362 NFR380 NFR381 NFR500 NFR510
… Products on Shelf Detection	It is(given the previously detected shelf contour) detection of product items within the shelf area / inside the shelf contour boundaries	FR100 FR150 FR151 FR201 FR202 FR230 FR251 NFR101 NFR200 NFR500 NFR510
ML Model Monitoring	Closely tracks call to the model, the number and duration of individual model calls, returned parameters in the happy case. Logs & archives all info to reproduce the error cases as easily as possible later in a test environment	FR100 FR250
RoboFlow Yolo Training Platform for Super Market Shelf Scanner	roboflow.com => Cloud-based best practice and industry standard ML pipeline for training custom object identification models (image & video feed). Includes easy to use & powerful features for	No explicit requirement. Implicit in our requirements that we need to have training pipeline. These requirements might have to be refined again.

	image labeling, data augmentation, model performance evaluation, model repository and model hosting/deployment	
Training Datasets	Stored in Roboflows cloud-based feature store for raw, labeled and augmented training datasets together with trained model & post training evaluation results	No explicit requirement. Implicit in our requirements that we need to have training pipeline. These requirements might have to be refined again.
Image Labeling Tool	Roboflow's cloud-based industry-class image labeling tool for custom model training with intelligent object marking / AI-assisted bounding box / polygon suggestions.	No explicit requirement. Implicit in our requirements that we need to have training pipeline. These requirements might have to be refined again.
Automatic Data Augmentation Tool	Roboflow's cloud-based industry-class image data augmentation tool (generate tilted, rotated, blurred image to enrich variation in training data set	No explicit requirement. Implicit in our requirements that we need to have training pipeline. These requirements might have to be refined again.
Model Training Pipeline	Roboflow's cloud-based training pipeline for training custom object detection & identification models.	No explicit requirement. Implicit in our requirements that we need to have training pipeline. These requirements might have to be refined again.
Model Repository	Roboflow's cloud-based model repository (part of feature store)	No explicit requirement. Implicit in our requirements that we need to have training pipeline.

		These requirements might have to be refined again.
Model Evaluation Reports	Roboflow's evaluation reports are part of the the standard training pipeline features provided.	No explicit requirement. Implicit in our requirements that we need to have training pipeline. These requirements might have to be refined again.

Design Questions

Q1-M: How can we train a image processor to detect our specific products without loosing too much time but not giving up on quality / industry best practices?

Use SaaS / Cloud-based Yolo training platform from RoboFlow (roboflow.com). Avoids the - by experience - big learning curve for label, augmenting training and evaluating the performance of custom object identification models (based on image / video feed).

Roboflow has become an industry-wide standard for training / adopting pre-trained object detection models. Most attractive is the strong support for labeling, image augmentation, performance evaluation and tracking over training datasets which can also very efficiently be organised in re-usable and extendable training sets/versions.

https://blog.roboflow.com/how-to-train-yolov8-on-a-custom-dataset/

Q2-M: How can we easily compare the models of our candidate models?

Roboflow provides extensive metrics from evaluation after training and good organisation of training data into datasets. Using these tools it is easy to review & compare the evaluation results after training and make an informed decision on whether you want to use the new candidate model for inference into staging or even production environment.

https://blog.roboflow.com/evaluate-roboflow-models/

Q3-A: How independent / flexible is the system to be delivered to different users / on different devices?

Our app can be achieved from any device with the possibility of internet connection and storing Images. So users can open web app despite different device types,browsers and OS.

Python/Django does this very well….

Q4-A: Can we adjust model performance for supermarket in special cases like products of “Eigenmarken” (internal/own brands Like Lidl)?

Note that we are saving the user-reviewed and approved results for image recognition together with the raw image and (potentially wrong) model inference results.

This allows us to detect cases where the model made an inaccurate prediction. Since the user also reviews and save correct product and count per product at the end we can use this information for focused model training - e.g. focusing on wrong count results of “Eigenmarke” products which might be introduced occasionally and are not as voluminous / prominent in the stores.

As we find a new candidate model in training with better performance on the “Eigenmarken” and thereby positively adjust the performance of these cases by deploying it to staging and eventually production.

Q5-A: Why do we choose Django Framework as web application framework for our web app?

For the starting phase there are several advantages of using Django as it provides better integration with AI module: Having AI models and the web container running in one common python run time environment, increases transparency, reliability of whole system and shortens system development and time to market time. This is a set of frameworks libraries which is well known to Python developers and therefore does not require expensive specialist upfront.

As the application grows one can easily use components like the Roboflow model server (in the cloud or as framework/wrapper for an independent microservice) without entirely changing the application architecture / overall design. The web application container might easily be scaled by running it together with a load balancer in the cloud to parallelise multiple web containers similar to AWS Beanstalk.

I.e. for the foreseeable future it looks like it will be easier to maintain with smaller budgets, team and qualification.

Q6-M: Will the app be usable for supermarkets like LIDL or HOFER from data privacy / DSGVO perspective?

Not clear. We need to check whether all legal prerequisites are fulfilled esp. from data hosting from an EU-DSGVO perspective. This might be tricky since we are using Roboflow for training which looks like a US-based cloud service & our data may leave the EU. Here license and hosting models need to reviewed closely and asap before we burn much development resources on an option which is not legally feasible.

If Roboflow out of the box solutions cannot be used then we can still build this ourselves since esp. Yolo is open source but then have develop part of the very

elobrated Roboflow platform ourselves (pot. high development effort and cost associated with this).

Q7-M: How can we debug and fix problems in principal and as quickly as possible (focus: ML models)? Do we have all data to reproduce errors in a separate staging / prod-copy environment?

This is addressed by design of our ML monitoring component. It continuously monitors the number of duration of model inference request and logs out input/output to the model. Errors are logged with all input parameters incl. the raw image and (partial) output / stack traces. In the error case the ML monitoring component shall produce a dump file with all info required to reproduce the error in separate environment.

Q8-M: How reliable does the model inference / product count have to be so that the users accept it (maybe) + what do they care when they use the AI product count most?

For users it is eventually of at most importance that results are correct / do not need to be reviewed and that they receive them quickly. Esp. correct identification of product. It is not so important whether the count is correct +-1 but rather that the magnitude is right since we will only re-order if threshold in inventory is undercut.

This needs to be managed from a phased project rollout and overall expectation management perspective. For example start with soda can counting, have users review result => input for focused training / model improvement. Once models stabilise move to next product group. This will be better than tackling all product at once and facing bad average model performance.

Q9-M: How much EUR are going to be burned / have to be paid for obligatory expense like Roboflow training, Hosting, Infrastructure in the Cloud etc.?

Esp. Roboflow could be prohibitive cost given that products change / shelves look slightly different in stores etc. Again: Eigenmarken!

We need to plan scenarios and do some research early to understand this better using a representative product set.

These scenarios need to be reviewed regularly with project management. We recommend a conscious decision on whether to proceed with development / implementation of proposed architecture only after initial review.

Q10-M: What could go wrong in the application flow because of unsafe interface or human error? How can the adverse effect be remediated?

One thinkable case is that the system submits a wrong product count (way too high or way to low) because a user accidentally misadjusted a product count during the results review. This would mean that we might submit a way to high or way to low

inventory which just does not make sense. Repeated submission of low inventory might lead to unnecessary restocking orders / upheaval in store replenishment logistics.

As countermeasure either our app or the WHMS should be extended with plausibilty check that highlight to the user for example that restocking has been triggered in pattern which is not consistent with normal inventory variance for the respective products.

The user should be warned and may only submit the product count finally after explicit confirmation.