Process document

Calvin Hong - 431518

# Context

This semester was focused on AI for Society. Students were to work on one group project and one individual project. The parents of the student run a snack bar, which was a unique advantage that was used for this project. At the moment, the snack bar has multiple points that could be improved during the process of preparing an order. One of the major bottlenecks that could be improved with artificial intelligence is the gathering of the items for an order.

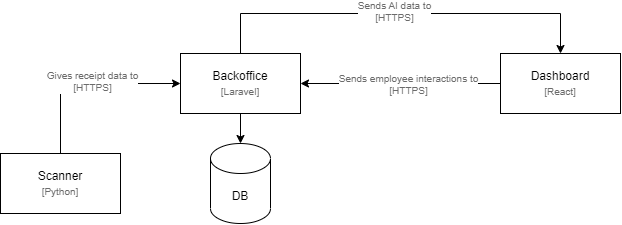
The final idea: **Snack bar preparation assistant**

The assistant AI will scan new receipts as soon as they get ordered and show the employee what items should be prepared first. This should help with the time efficiency, since not all the items cook in the same amount of time. For example, a hamburger takes around 10 minutes, while a frikandel takes around 5. Having a screen that shows you what you should prepare in order of time should make the process more time efficient.

Some items are also not in the front deli case at the work station. To get these items the employee has to go to the back. This sometimes results in having to make multiple trips because it is difficult to remember all the items listed. The assistant can group these items together in an overview so that it is clearer for the employee.

# Project components

To achieve the desired outcome, which could be developed further on in the future, multiple aspects come into play.



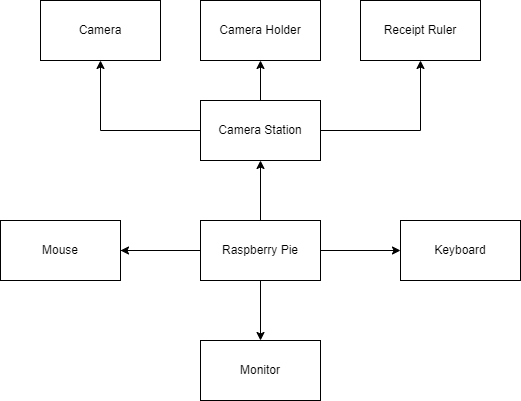
In the diagram above, the parts of the system is displayed.

## Scanner

The scanner is the starting point of the process. When a customer orders a meal, the cashier creates a new order in the cash register. In the ideal situation, the cash register could be connected a system where is could be saved. However, the cash register used is an older model that cannot do this. Because of this, a scanner was created that could scan the receipts.

#### Structure Scanner

The scanner consists out of a Raspberry Pie connected to a camera.

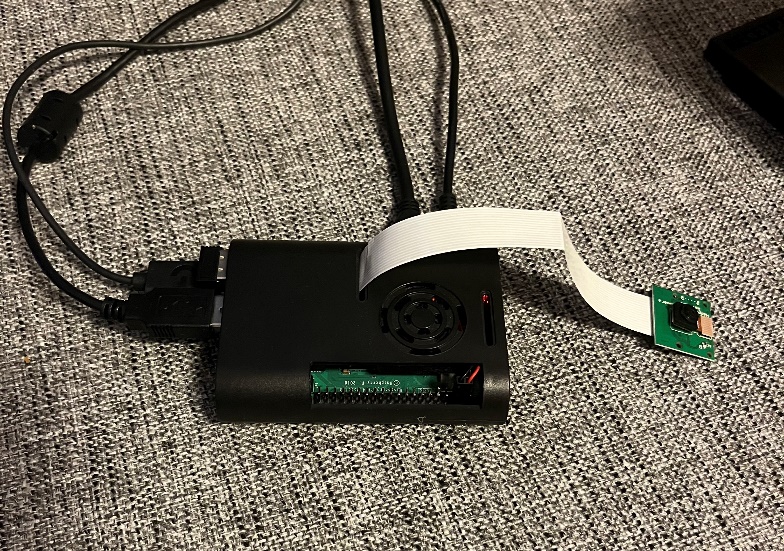


In the diagram and picture above, you can see the final prototype result of the scanner. At the center is the Raspberry Pie, which is connected to the Backoffice. Besides being wired to the necessities (keyboard, mouse and monitor), it is also connected to a camera station where the receipts get scanned. This station consists out of three main parts: a camera, camera holder, and receipt ruler.

On the receipt itself, there was much data that was unnecessary, so an additional custom receipt “ruler” was also made. This ruler blocks the data that was not needed like the address and phone number of the snack bar. See the picture below.



The initial idea was to use a Raspberry Pie camera, but unfortunately this was not ideal to use. It was focus length not adjustable and the cable was too short (see picture below).



To overcome this issue, an adjustable webcam was used together with a 3D printed holder to keep the distance consistent. The lens could be rotated to adjust the focus/sharpness. See the picture below.



#### Python-Tesseract

To get the data on the receipts, Python-Tesseract, or just Tesseract, was used. Tesseract is an optical character recognition (OCR) tool that can be used with python. It can get/read the text on images. This OCR is based.

Before Tesseract was used to actually get the receipt data, the image first had to be improved. The unedited receipt image looks like this:



OCR systems like to use black and white (0,1) instead of full color(0,255), so the first step was to convert the image to black and white. After this, the gaps were filled in as much as possible, which resulted in the following:



With the improved image, Tesseract now comes into play. When passing along the image, one can also change some configurations. One of the most important setting is PSM, or Page Segmentation Modes. By default, Tesseract expects to receive a block of text, but the image we are providing it is far from a page of text though. By changing the PSM we can change the expectation of Tesseract and in turn change the outcome of the data we receive. A full list of the available modes can be found by looking it up online or by using the command

This will return:

Page segmentation modes:

0 Orientation and script detection (OSD) only.

1 Automatic page segmentation with OSD.

2 Automatic page segmentation, but no OSD, or OCR. (not implemented)

3 Fully automatic page segmentation, but no OSD. (Default)

4 Assume a single column of text of variable sizes.

5 Assume a single uniform block of vertically aligned text.

6 Assume a single uniform block of text.

7 Treat the image as a single text line.

8 Treat the image as a single word.

9 Treat the image as a single word in a circle.

10 Treat the image as a single character.

11 Sparse text. Find as much text as possible in no particular order.

12 Sparse text with OSD.

13 Raw line. Treat the image as a single text line,

bypassing hacks that are Tesseract-specific.

By using PSM 4 (assume a single column of text of variable sizes), we change the expectation and in turn change the data we receive. Using the wrong PSM could result in wrong data or no data at all. Alongside the PSM, we also give the image itself and what kind of output we want. This gives back:



XXX DATA PSM 3 DEFAULT AGAINST PSM 4

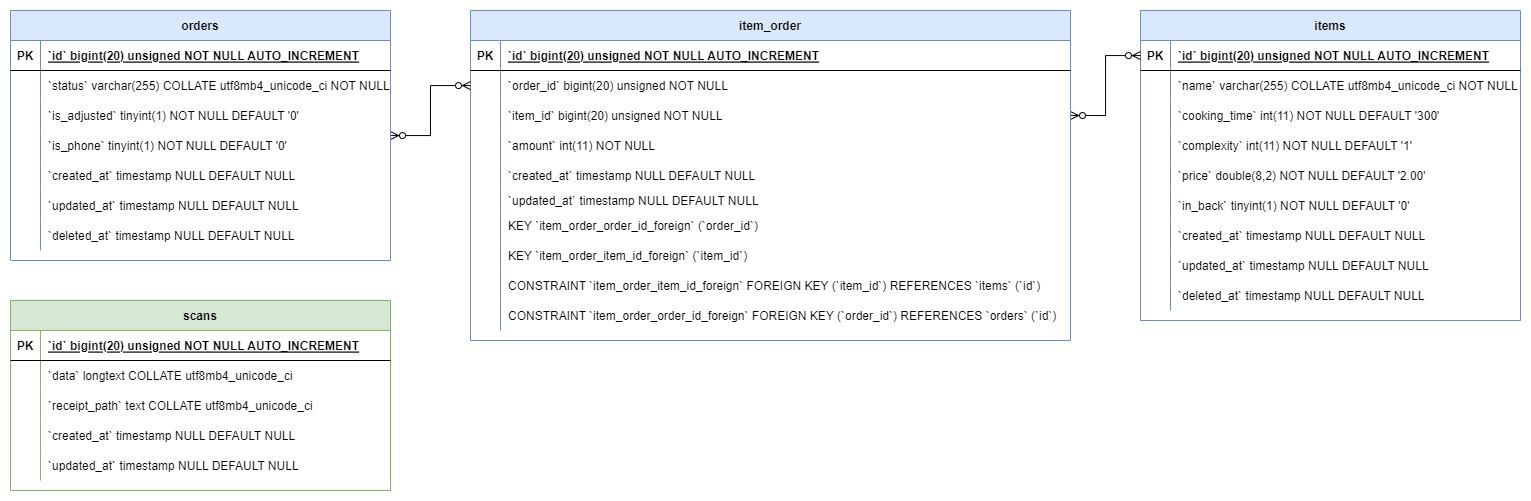
Tesseract is not the only form of AI/Machine Learning in this project. The data could be incorrect if the receipt is adjusted by hand/pen, so a warning should be given. Furthermore, the snack bar has 2 different cash registers with different printers. One for phone orders, and one for one-site orders. These receipts look different, so to distinguish them and see if they are adjusted, Tensorflow is used.

XXX TENSORFLOW EXPLANATION DATA HERE

This data then gets send to the Backoffice.

## Backoffice

The Backoffice stands in the center of the system. It is created with the PHP framework Laravel. In here the scans, orders and items can be seen and adjusted. The database structure looks like the following:



scans: The received text data with the confidence. Also the path where the image is saved.

*orders*: keeps track of orders with its items through the link table *item\_order.* Has status column which can be: WAITING, IN\_PROGRESS or FINISHED. Also has columns for adjusted or phone.

item\_order: pivot table for Order Items

items: product/item (e.g. fries with mayo, burger, drink, etc). Has the default columns like *name, cooking\_time, and price.* The column *in\_back* keeps track of if the item is in the back. For these items, a notification must be given to the employee. The *complexity* is how hard it is to prepare.



For the items, a database seeder is made so that all the items (currently a total of 252 items) do not have to be created one by one, but can be created automatically in one go.

With all this preparation, the scan can be processed.

#### Processing scans

When a scan gets received, the Backoffice first breaks it down into steps:

1. Store the image in Laravel Storage
2. Store scan data in *Scan* model
3. Turn text data into JSON format
4. Create an *Order* model
5. Loop over received text JSON object
6. Check if amount is an integer
   1. If not -> change to 1
   2. If bigger than 99, change to 1
7. Find item by name using levenshtein method
   1. If item found, return item
   2. If no close item found, return “*diverse*”
8. Attach item to order
9. Return response to scanner

XXX EXPLAIN ALL STEPS HERE

XXX ADD VIEWS HERE

This is the completed prototype of the Backoffice. The dashboard can connect to this platform to receive the most recent data.

XXX ADD HOSTING?

## Dashboard

The dashboard will be in the hands of the employee who has to prepare orders. It is created with React.js and will be displayed on an iPad which can be mounted. This results in no hinderance in mobility or working space.

#### Functions

XXX Style first and then add views, For each view, explain what it does

# Live Action

XXX Video demo explanation images etc.

# Follow up suggestions

XXX Look into the document *Follow Up Suggestions*

# Conclusion

XXX Conclusion , emphasize follow up projects and what you learned, look at other docs you made, refer to other docs.