Batch Menu Reading with OCR and OMR

MAJOR INDIVIDUAL PROJECT

Jayash Raj Mudbhari (802105673)

November 24, 2019

Contents

1 Acknowledgment

I would like to thank Mel Razmjoo and others who have helped and guided me to help complete the project. I would also like to acknowledge that this project, reading paper based menus is based on the menus and procedures of my workplace.

2 Abstract

According to (Forbes, 2016), doctors spent two hours doing paperwork for every hour they took care of patents. Similarly, (The Independent, 2018) shows that nurses have had to sacrifice patient care due to paperworks that they were required to fill. While paperwork are important part of keeping comliance with regulations and are required for audits, finding ways of automating these processes can help both the patients as well create time for staff for other important tasks. This project in a similar light, aims to automate reading and obtaining meanigful data from paper-based menus found in some hospitals.

3 Introduction

I work in a health-care organization in the catering department. One of many ongoing issues we encounter is not having accurate counts of orders in-order-to prepare the right amount of food for the patients while minimizing waste. The problem in the food ordering system is that it is based entirely on paper which requires manual work to count and sort, and is tedious and prone to error.

Batch Menu Processing with OMR utilizes non-traditional[1] OMR[2] technology in order to read and analyze paper based menus, the marks are then stored into a database and used to generate a meaningful report.

3.1 Aim of the project

The objective of this project is to automate the work of a diet aide[3], or to assist them by making the process faster and reducing manual work. The OMR methodology used in this project can be adapted to read other documents with similar characteristics.

The python program uses open-source libraries like `Open-CV` and `Numpy` for image processing, `Redis` No-SQL database to store data and `python-docx` to generate the report.

3.2 Key Performance Indicators (KPI)

The indicators for the performance of the the project can be measured by:

- Project Schedule
- Estimation of completion
- Milestones completed

3.3 Project Deliverables

The project deliverable for this undertaking is an application that is capable of reading menus and identifying speical diet needs and generate a useful report based on those information.

4 Background

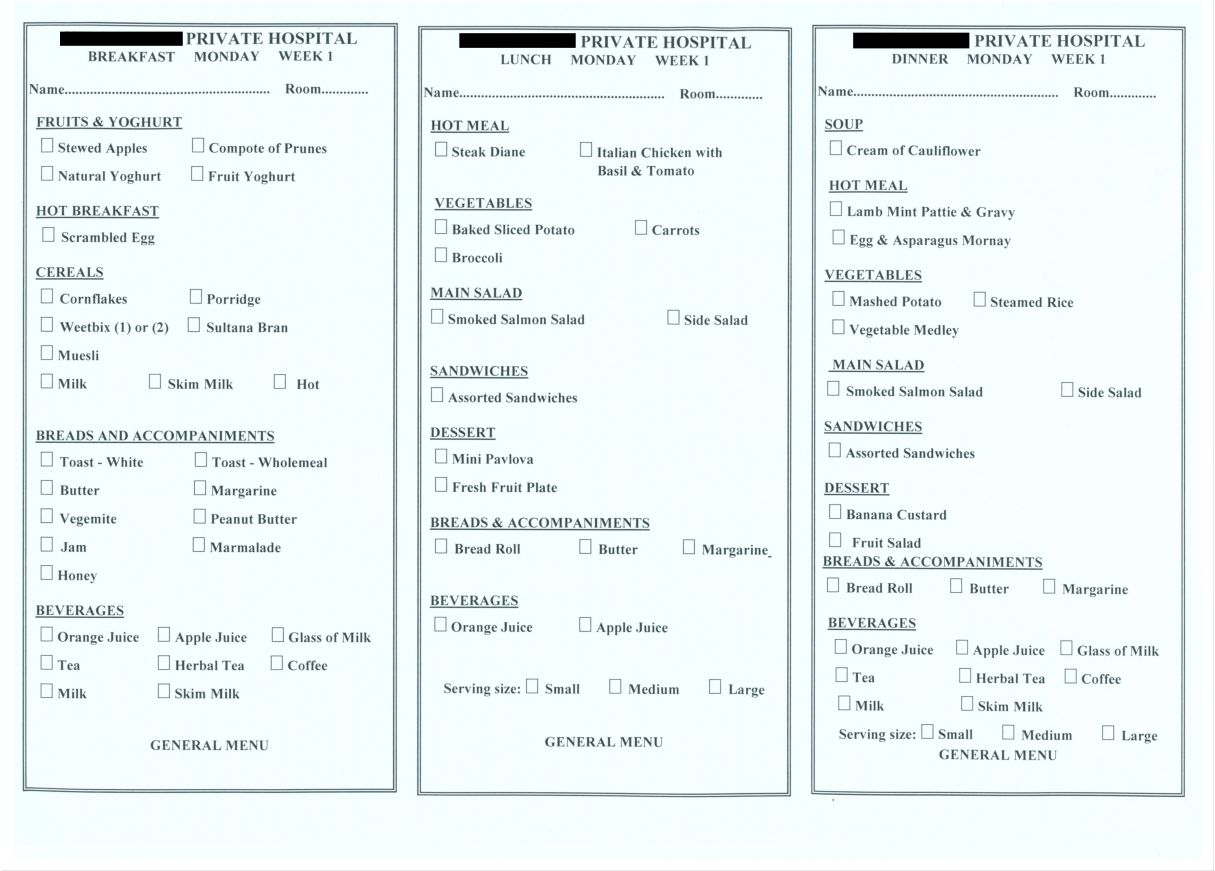
In order to appreciate the program and what it does, it is essential to understand the role of diet aide who currently performs the tasks as well as how the current food ordering system works. If you do not wish to learn too much about currently existing procedures, you can skip to the next section.

The health-care organization provides meals to about 50 patients daily three times a day (breakfast, lunch and dinner). The orders of the patients are filled by patients themselves (shown below):

[1] Non-traditional in a sense that it requires no specialized OMR sheet

[2] Optical Mark Recognition, a process of obtaining marks and marked data from printed documents.

[3] A person responsible for preparation and/or handling meals prepared by cooks. (Indeed.com)



[redacted] PRIVATE HOSPITAL
BREAKFAST MONDAY WEEK 1

Name..... Room.....

FRUITS & YOGHURT
☐ Stewed Apples ☐ Compote of Prunes
☐ Natural Yoghurt ☐ Fruit Yoghurt

HOT BREAKFAST
☐ Scrambled Egg

CEREALS
☐ Cornflakes ☐ Porridge
☐ Weetbix (1) or (2) ☐ Sultana Bran
☐ Muesli
☐ Milk ☐ Skim Milk ☐ Hot

BREADS AND ACCOMPANIMENTS
☐ Toast - White ☐ Toast - Wholemeal
☐ Butter ☐ Margarine
☐ Vegemite ☐ Peanut Butter
☐ Jam ☐ Marmalade
☐ Honey

BEVERAGES
☐ Orange Juice ☐ Apple Juice ☐ Glass of Milk
☐ Tea ☐ Herbal Tea ☐ Coffee
☐ Milk ☐ Skim Milk

GENERAL MENU

[redacted] PRIVATE HOSPITAL
LUNCH MONDAY WEEK 1

Name..... Room.....

HOT MEAL
☐ Steak Diane ☐ Italian Chicken with Basil & Tomato

VEGETABLES
☐ Baked Sliced Potato ☐ Carrots
☐ Broccoli

MAIN SALAD
☐ Smoked Salmon Salad ☐ Side Salad

SANDWICHES
☐ Assorted Sandwiches

DESSERT
☐ Mini Pavlova
☐ Fresh Fruit Plate

BREADS & ACCOMPANIMENTS
☐ Bread Roll ☐ Butter ☐ Margarine

BEVERAGES
☐ Orange Juice ☐ Apple Juice

Serving size: ☐ Small ☐ Medium ☐ Large

GENERAL MENU

[redacted] PRIVATE HOSPITAL
DINNER MONDAY WEEK 1

Name..... Room.....

SOUP
☐ Cream of Cauliflower

HOT MEAL
☐ Lamb Mint Pattie & Gravy
☐ Egg & Asparagus Mornay

VEGETABLES
☐ Mashed Potato ☐ Steamed Rice
☐ Vegetable Medley

MAIN SALAD
☐ Smoked Salmon Salad ☐ Side Salad

SANDWICHES
☐ Assorted Sandwiches

DESSERT
☐ Banana Custard
☐ Fruit Salad

BREADS & ACCOMPANIMENTS
☐ Bread Roll ☐ Butter ☐ Margarine

BEVERAGES
☐ Orange Juice ☐ Apple Juice ☐ Glass of Milk
☐ Tea ☐ Herbal Tea ☐ Coffee
☐ Milk ☐ Skim Milk

Serving size: ☐ Small ☐ Medium ☐ Large

GENERAL MENU

Figure 1: A sample blank menu given to the patents for filling

4.1 Diet Aide

The diet aide performs several important tasks that are required to do prior to providing the menus and a report to the kitchen, who then cook and prepare the meals accordingly. These tasks are:

- handing out menus to every patient (done a day prior, i.e. menus for Monday are given on Sunday),
- collecting them after they are filled by the patients[4] (same day),
- arranging these menus,
- identifying any special diet needs,
- counting the number of each item on the menu
- generating a report for the next day based on the count
- enagaging with patients their, family as well as nurses to ensure diet needs are met

The majority of the time spent by the diet aide is counting and generating the report based on the counts of the menus. While not all of the tasks can be automated, by eliminating this manual task of counting and as well as by adding some intelligence to the program that can recognize some diet needs, the work load of the diet aide can be reduced so that he/she may engage more with the patients and may help imporve the work experience.

4.2 The Menu Life Cycle

During my presentation it seemed that I was not able to convey how menus are used at each step, which is why I have tried to visually represent the steps involved throughout the lifetime of the menus, which may help the reader better understand the scope of the project. While the scope of the project is only to automate the task of one individual (diet aide) and not the whole team of people. The reader must not forget that a software is unable to physically deliver the food to the patients so that must be considered also. In order to narrow it down, the program has only the following functions:

- Read scanned images on menus
- Identify the ticked items and generate a report
- Indentify the special diet needs

All of these tasks are done by the diet aide as described in the section earlier.

4.3 Menu Structure

The menu is divided into three sections, breakfast, lunch and dinner. The menu also consists of other structures such as the day of the week and week. There are two weeks of menus, that is the food items in week one's monday is different from Monday in week two. The diet of the patient is listed at the bottom of the menu, "GENERAL MENU" shown in 6. Other diets can be diabetic, gluten free and minced diets.

5 Literature Review

Optical Mark Recognition (OMR) is a process that allows capturing checkboxes and text written into forms and papers. This process differs from optical character recognition (OCR) which is used to read printed and writted text from paper. (*OMR vs. OCR – Difference and Comparison – Diffzi*, n.d.) The difference in the application of these two technologies also lies in the application of these technologies. OMR is used for analyzing answers and results in a survery or the answers filled out in a form. Whereas, OCR is generally used to digitize printed doducments and is not capable of reading forms and marked surveys. While the technological capabilitites may not always be accurate in both technologies, the major drawback of OMR is that it is generally used with a specialized paper answer sheet as shown below:

The problem with implementing OMR is the use of these specialized sheets which may not be available and are less flexible compared to the use of traditional printed papers. Some OMR papers use a specialized ink that look less pronounced when scanned at specific wavelengths to reduce the margin of error. They may also not be as easy to use compared to traditional paper form for elderly and people with disabilities in the hospitals.

[4]Sometimes diet aide or a nurse has to assist a patient who is unable to fill the menu themselves

Week 2: Saturday Dinner

Steak & mushroom sauce (Large: Rm15a, 30) 13
Honey soy chicken (Large: Rm4a, 13, 20b, 23, 26, 38, 40) 36
Mashed potato 21
Mixed vegetables 24
Rice 13

Turkey salad 3
Vegetarian: Cheese salad (if poss: Smoked salmon, olives, Spanish onion & semi-dried tomato rm18.
Side salad 18 Rm42: No garlic, onion & capsicum.

Assorted sandwiches 5
Supper: Rm8, 20b

Apricot sponge & custard 19
Diabetic apricots & custard Rm8, 17, 44, 14
Two fruits rm22
Fruit salad 21
Ice cream rm 16b, 22, 25, 37, 41, 46.

Banana rm4a & 43, Sliced wholemeal bread rm19 & 28.

SPECIAL:
Half: tomato soup **rm29**, **36**

Minced for soft diet: Honey soy chicken & extra sauce, Diced mix vegetables **rm22**
No garlic, onion & capsicum: Minute steak & lemon & mix vegetables **rm42**
No gravy: Soft Minute steak **rm43**

No corn: mix vegetables **rm34**

Figure 2: An actual report prepared by the Diet Aide.

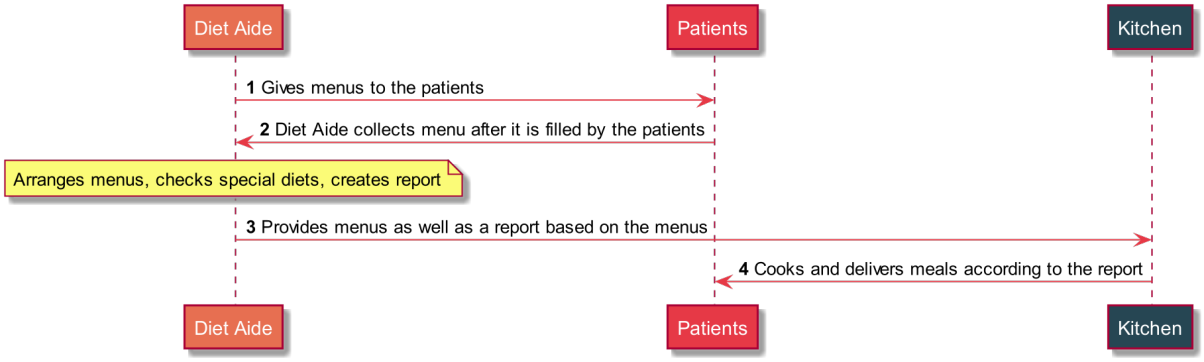


Figure 3: Report generation by the Diet Aide

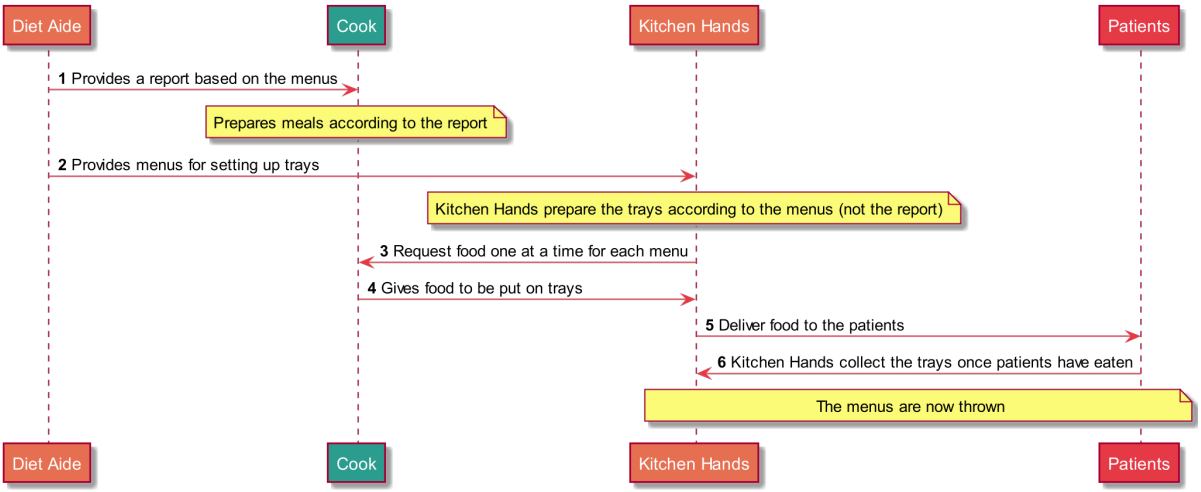


Figure 4: Breakdown of how the report and menus are used

Table 1: Benefits of electronic meal order system (Source: Serial Multivision)

	Traditional	Electronic System
Patient Menu	Genric	Personalized
Safety Checks	Manual	Auto
Order Taker	Nurse	Anyone
Dietitian Instructions	Repeated, Manual	Once, Auto-reccuring
Meal Slip sorting	Manual	Auto
Alerts and Notification	No	Yes
Dish and Menu Management	Manual	Auto
Detailed Reports	Manual Effort	Automatic

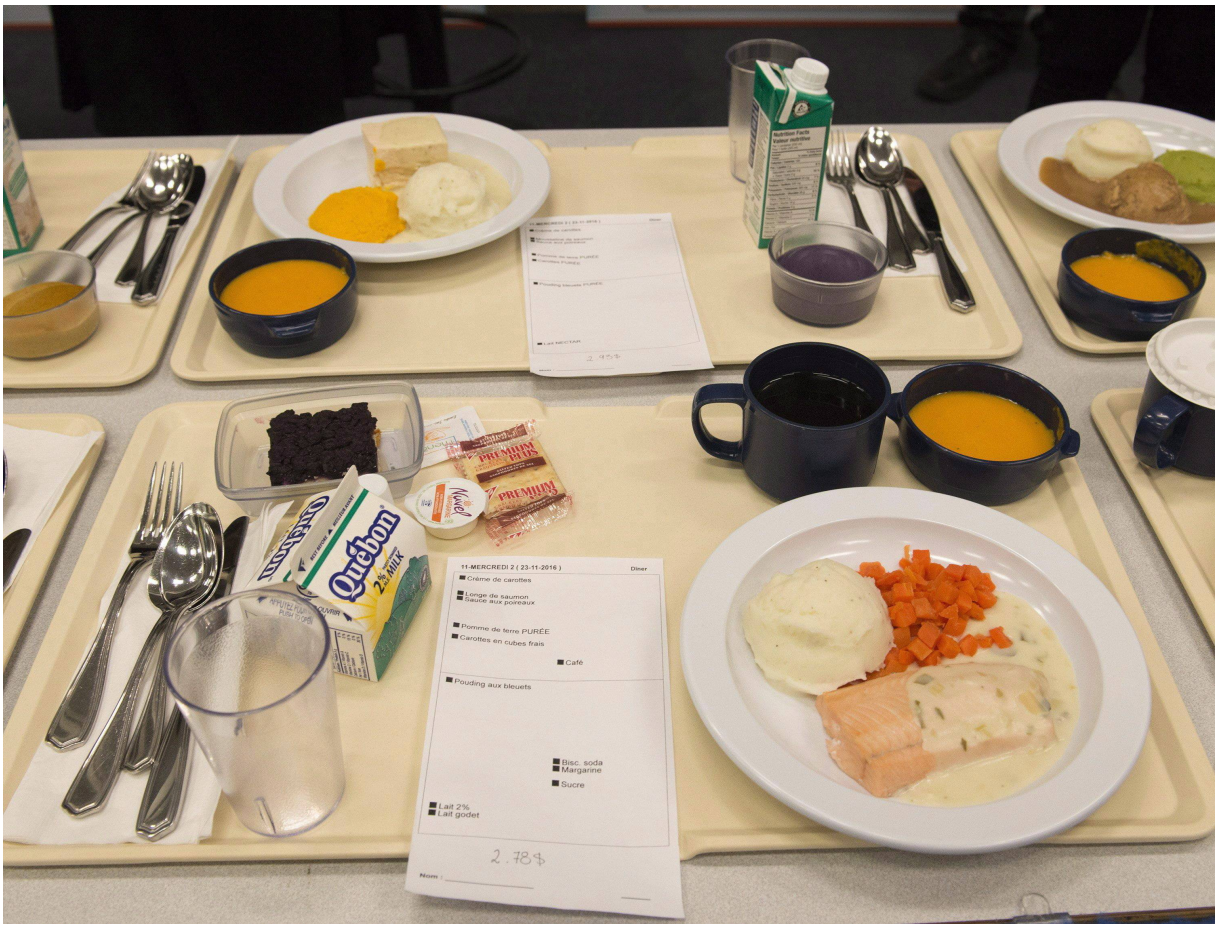


Figure 5: Example of the tray after it is set up by the kitchen (Image Source: The Canadian Press)

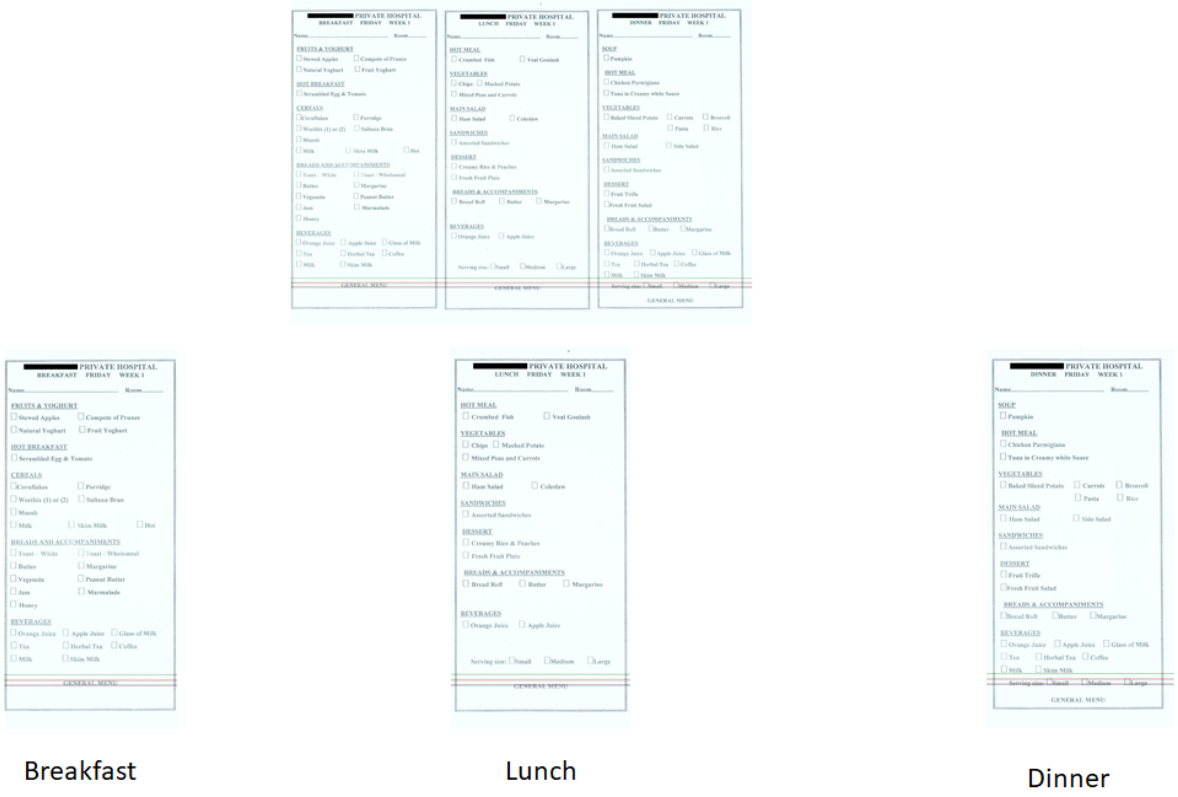


Figure 6: Each paper consists of three section.

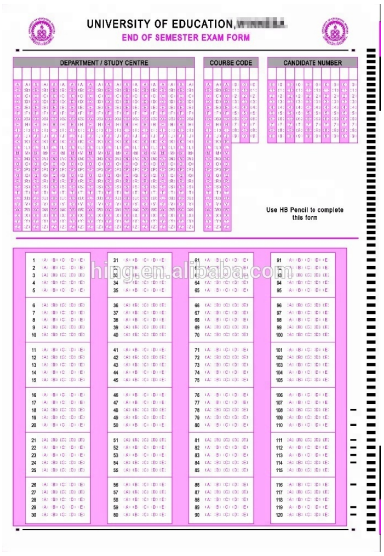


Figure 7: A sample OMR page

5.1 Why not use iPads, Webiste or a digital system?

An alternate method that can be used is to use a tablet device or iPad to allow patients to order their meals. (*Food Ordering System for Patients | Alpha HROS*, n.d.) However implementing this requires a large investment per patient compared to traditional paper based method. If OMR can be used in a non-traditional way (without using the specialized sheet) as demonstrated by programs like RescueOMR (*RescueOMR: batch Optical Mark Recognition without foresight*, n.d.), then this system can be used without distrupting the currently existing system based on paper and can reap the benefits of using a digital system with advanced intelligence.

According to the Patent (Taylor, 2004), firstly, the blank template of form or image is scanned as a guide on which the scanned forms are compared after they are transformed using Affine transformation (lines are not altered and ratio of distances are kept same) to match the base image. After the blank form and new scanned form are aligned as finely as possible, per pixel intensity differences are calculated. The difference in the values can help determine if the region has been marked or not. Same procedure is applied to multiple forms or areas based on the design of the application. Page alignment and resolution changes may also be required in order to achieve accurate results.

5.2 Image Registration

Image Registration is the process of normalizing or transforming an image to match characteristics of another image for the purpose of comparision or other advanced image manipulation techniques. (Zitova, 2019) Image registration as well as other methods such as dynamic resolution scaling can futher improve the capability of the program to function with different image scans.

5.3 Template Matching

Compared to method described earlier in (Taylor, 2004), which uses per pixel difference after a near accurate image registration (transformed to match a previous template) the method that is being implemented in my project uses template matching. This process involves sliding the template over another image to find regions where the template matches most accurately (*Template Matching — OpenCV 2.4.13.7 documentation*, n.d.). The benefit of this process, I belive is that it can allow use of different scanning devices. This may be accomplished by using various ratios to accomodate the template and the menus that are scanned. Template matching may also allow narrowing down of options and futher OMR contour detection can be applied on the space. The logic behind this is that a template can be an item on the menu such as Soup. This template is checked on the filled menu to narrow down the region where soup is present.

Figure 8: Template matching is used to detect multiple coins

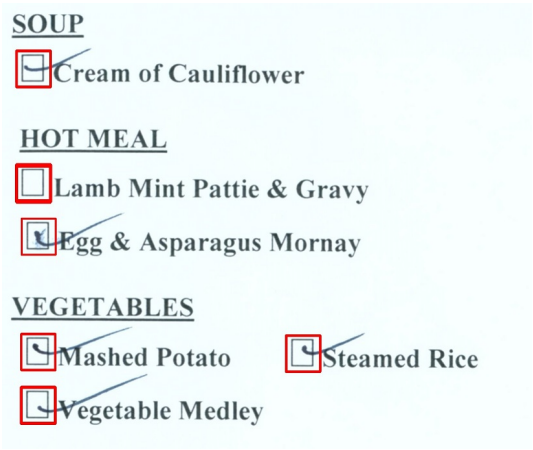
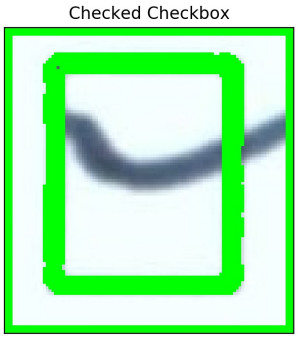


Figure 9: Matches of checkboxes using template matching

5.4 Cotour detection

Contour detection is a process of identifying shapes in images. Shapes such as square and bubbles or circles are used for marking or selecting answers generally with OMR. (*A Box detection algorithm for any image containing boxes.*, n.d.) Contour detection helps the program to select only these marking points and ignore the other fields in the form which are not marked by the user. This can be applied after template matching has been done to actually identify if the item corresponding to the template has been checked or not. The shape that marks the answer should be distinct from other shapes in the form or should be enclosed in a table like structure to futher isolate the points and reduce the posibility of improper readings done by the software. Projects and tutorials such as (PyImageSearch, 2016) have used contour detection and sorting, however for my project, I have relied exclucively on template matching and it has worked effectively. It is also possible to identify the square shaped contours and then loop over the number of squares

Figure 10: How contour detection can be used to detect shapes, using different values (Source: OpenCV)



6 Methodology

The project was undertaken with little understanding and prior knowledge of many components used. Thus, research and learning were major parts of this undertaking. While the general sturucture of the project was based on waterfall methodology of project management, the coding process of the program was done in small increments similar to Agile methodology. It can be said that this was a hybrid approach in undertaking the project and included the best of the both worlds, i.e. the initial phases were conducted sequentially and the implementation of the project was done in small and functional chunks.

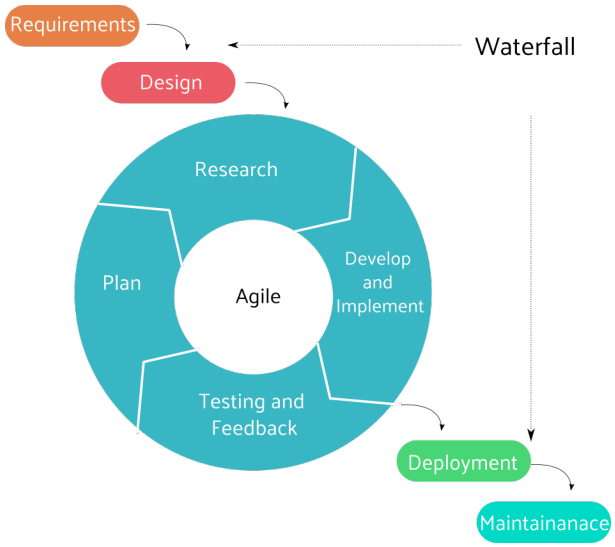


Figure 11: Hybrid project management methodology was used for this project

7 Development

As mentioned prior in the section above, the development of the application was done in small sprints in accordance with the Agile methodology. Small features were introduced one at a time and other features were added on top of existing features to achieve the completion of the project.

7.1 Jupyter Notebook

Jupyter Notebook is an interactive application that allows developers to write live code, that is the file contains the code as well as returns the results which can be stored in the document. Using Jupyter Notebook, small functionalities like reading the images, rotating the images, and so on can be done step by step with progressive increments.

7.2 Timeline

The timeline for the project can be divided in to following stages:

- Collecting Requirements
- Research
- Design
- Development
- Testing
- Feedback
- Release and Maintainance

Rotation and Cropping

[101]:
```
import cv2
import numpy as np
import imutils
from imutils.perspective import four_point_transform
from imutils import contours
%config InlineBackend.figure_format = 'retina'
import matplotlib.pyplot as plt
img_color = cv2.imread('Scans/Full/blank.jpg')
plt.title('How OpenCV images (BGR) display in Matplotlib (RGB)')
plt.xticks([]), plt.yticks([])
plt.imshow(img_color)
plt.show()
```

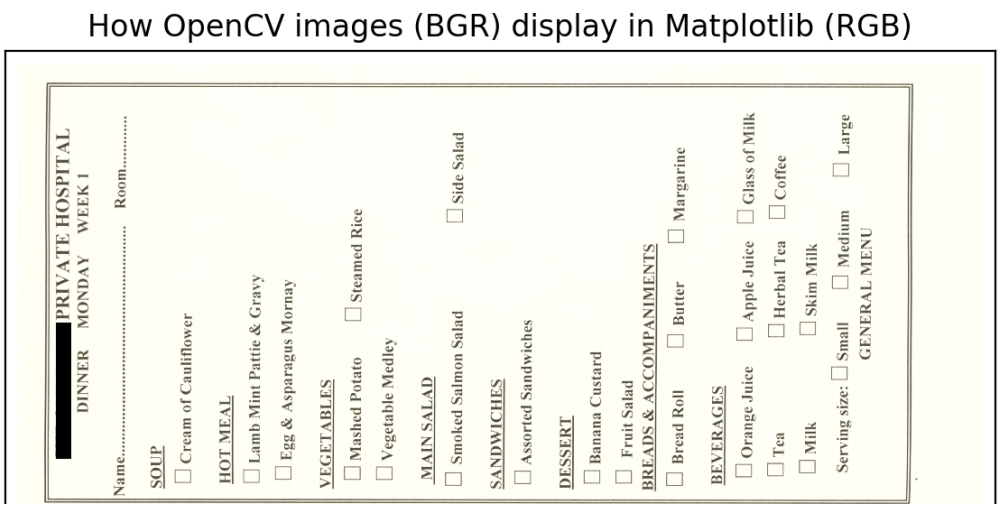
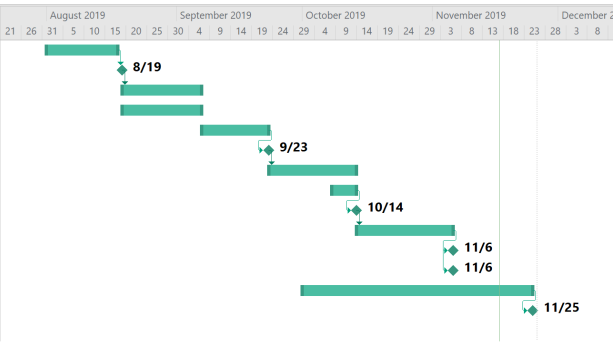


Figure 12: Jupyter Notebook

The timeline for the project was based around the required milestones and deliverables to the subject teacher.

Task Mode	Task Name	Duration	Start	Finish	Predecessors
	Proposal Writing	13 days	August 1	August 17	
	Proposal Submission	0 days	August 19	August 19	1
	Collecting Requirements	15 days	August 19	September 6	2
	Literature Review and Re	15 days	August 19	September 6	
	Analysis and Project Plan	12 days	September 7	September 22	
	Project Plan Submission	0 days	September 23	September 23	5
	Development Phase 1	16 days	September 23	October 13	6
	Testing	5 days	October 8	October 13	
	Assesment	0 days	October 14	October 14	8
	Development Phase 2	17 days	October 14	November 5	9
	Results	0 days	November 6	November 6	10
	Presentation	0 days	November 6	November 6	10
	Report Writing	40 days	October 1	November 24	
	Final Report Submission	0 days	November 25	November 25	13



8 Cost

There are no costs that are required in develeoping or creating this project apart from time spent coding and testing. The equipments used are a scanner and a computer to process the code. Scanners are generrally available in every hospital or office and hence this project should bear no additional cost when implemented. If no such provisions are in place, a cheap computer such as Raspberry Pi and a scanner that has a feeder tray to scan multiple documents at once is all that is needed. The total costs for these seem be be about $250.

9 Analysis

Batch Menu Processing with OMR and OCR uses several components to perform different actions, collectively working to achieve the end goal, which is to generate a word document form scans of multiple menus.

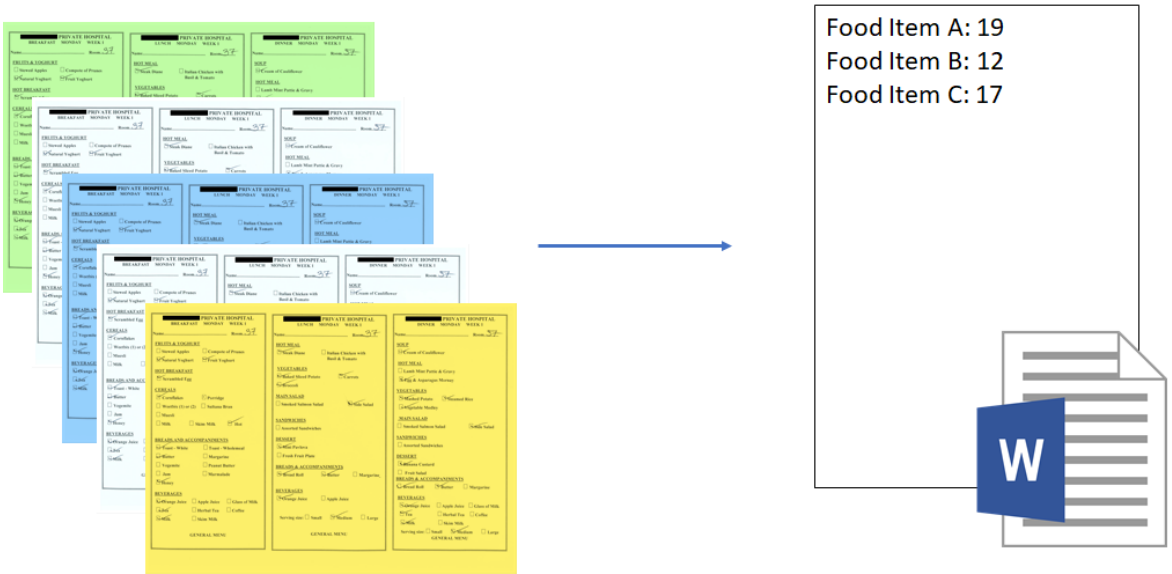


Figure 13: The program, to put it simply generates a report from scanned menus

While it can be seen that the handwriting recognition of `tesseract` is not as accurate, it is accurate in reading the required printed text.

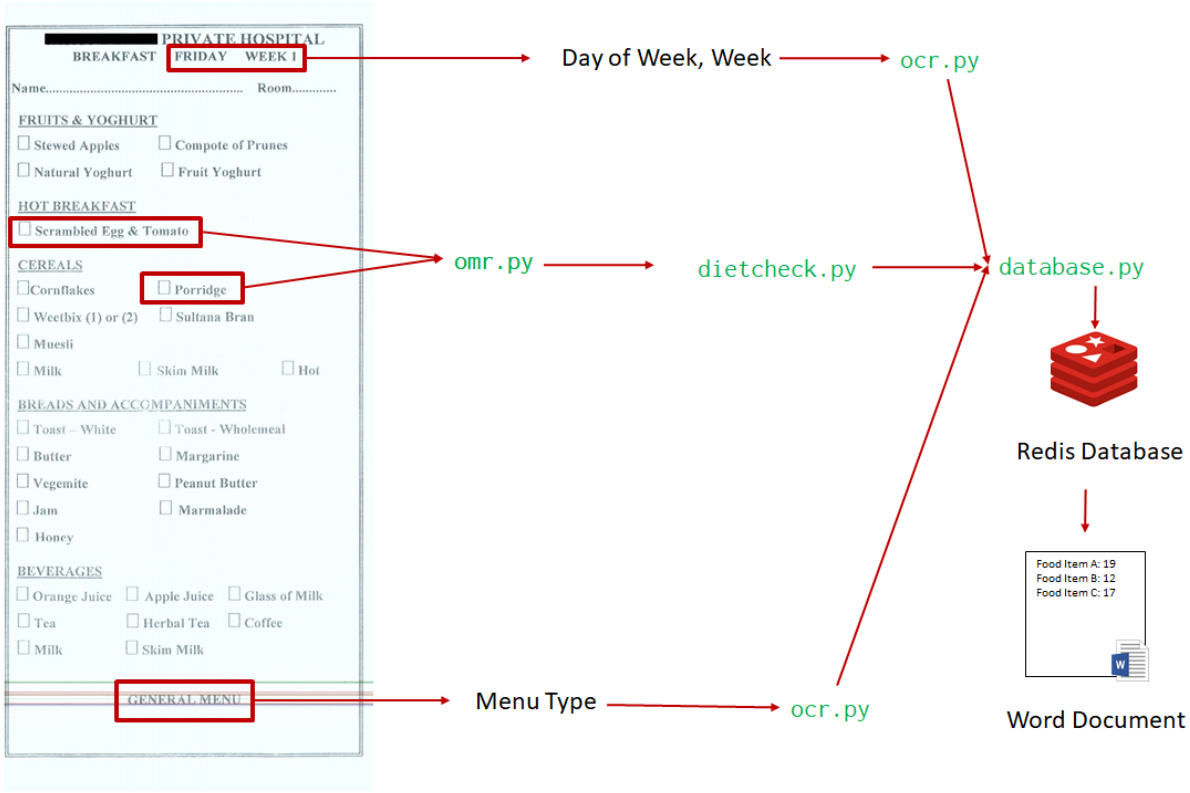
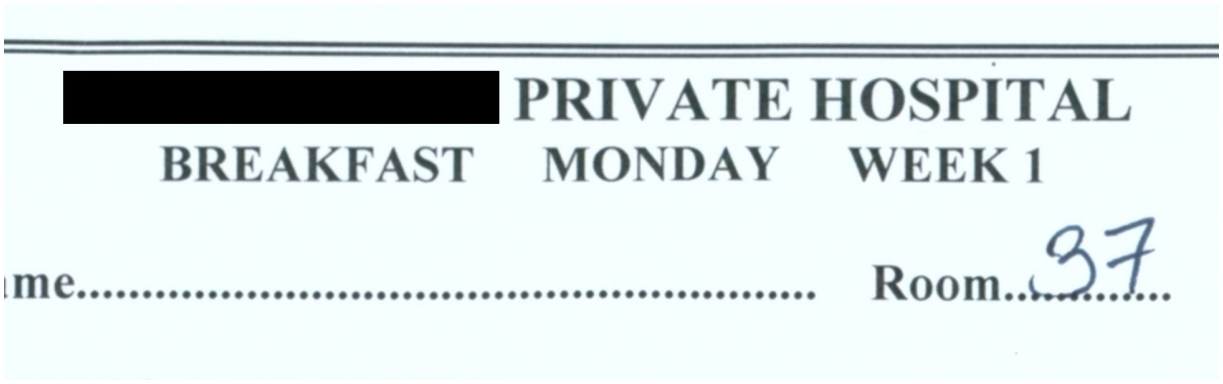


Figure 14: General Breakdown of how the menu is read.

9.1 OCR

Optical Character Recognition is used in identifying the week and the day as shown in 14. A minimal section of the image (top 20 % and bottom 20% as determined by measuring the menu) is is passed to the 14.5 which uses Tesseract and it's python interface pytesseract is used to read the printed characters in the menu. This allows the program to be robust. OCR with printed text has high level of accuracy.

For example, the section of image sent to the OCR library is:



The resulting output is:

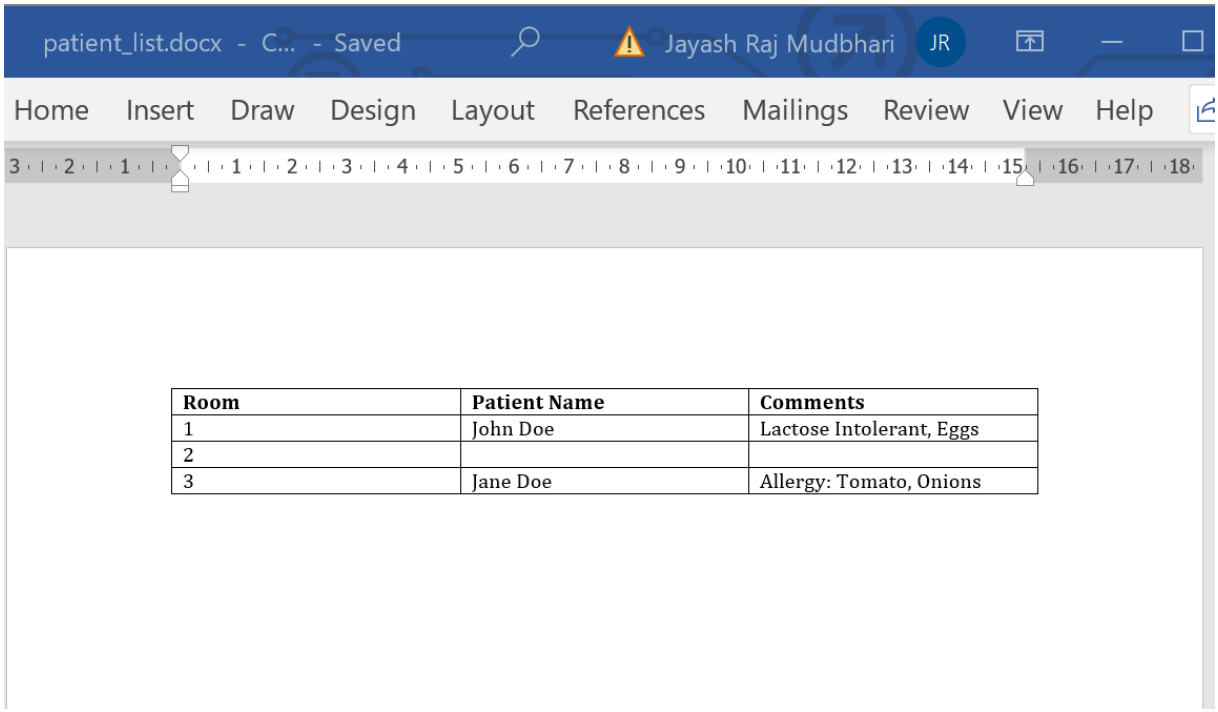
```
‘ERE PRIVATE HOSPITAL

BREAKFAST MONDAY WEEK 1

DG erie ceieerreesrereeerees terns Room. 4.
```

9.1.1 python-docx: Reading Patient List

The `read()` function in 14.6 reads the ”patient_list.docx” file which could be anywehre in the network or in the local drive. The python script reads the table in the document to identify rooms which are occupied, and identify additional information about the patient.



Room	Patient Name	Comments
1	John Doe	Lactose Intolerant, Eggs
2		
3	Jane Doe	Allergy: Tomato, Onions

The output generated by the `read()` function is as follows (note: it skips where patient name field is empty, noting that the room is vacant):

```
[{'Room': '1', 'Patient Name': 'John Doe', 'Comments': 'Lactose Intolerant'}, {'Room': '3', 'Patient Name':
↪  'Jane Doe', 'Comments': 'Allergy: Tomato, Onions'}]
```

9.2 OMR

The optical mark recognition compoment 14.4, reads the templates in the folders organized according to the day, week and the diet type of the menu and identifies if each of the templates are checked or not.

The following is an example such templates:

The templates are sorted in the following way in folders:

```
Week 1
```

```
    ├── Friday
    │   ├── Breakfast
    │   ├── Dinner
    │   └── Lunch
    ├── Monday
    │   ├── Breakfast
    │   │   ├── Porridge.jpg
    │   │   └── Scrambled Egg.jpg
    │   ├── Dinner
    │   │   ├── Assorted Sandwiches.jpg
    │   │   ├── Banana Custard.jpg
    │   │   ├── Cream of Cauliflower.jpg
    │   │   ├── Egg and Asparagus Mornay.jpg
    │   │   ├── Fruit Salad.jpg
    │   │   ├── Lamb Mint Pattie and Gravy.jpg
    │   │   ├── Mashed Potato.jpg
    │   │   ├── Side Salad.jpg
    │   │   ├── Smoked Salmon Salad.jpg
    │   │   ├── Steamed Rice.jpg
    │   │   └── Vegetable Medley.jpg
    │   └── Lunch
    │       ├── Baked Sliced Potato.jpg
    │       ├── Broccoli.jpg
    │       ├── Carrots.jpg
    │       ├── Fruit Plate.jpg
    │       ├── Italian Chicken with Basil and Tomato.jpg
    │       ├── Mini Pavalova.jpg
    │       ├── Sandwiches.jpg
    │       ├── Side Salad.jpg
    │       ├── Smoked Salmon Salad.jpg
    │       └── Steak Diane.jpg
    ├── Saturday
    ├── Sunday
    ├── Thursday
    ├── Tuesday
    └── Wednesday
└── Week 2
    ├── Friday
    ├── Monday
    ├── Saturday
    ├── Sunday
    ├── Thursday
    ├── Tuesday
    └── Wednesday
```

This folder structure allows easliy adding and removing required items according to the diet of the patient.

The templates are created manually and identify the items that require to be read by the program.

The method of recognizing checkboxes is by firstly masking checkboxes in template as well as a template of a blank checkbox. Because it is relavant to this section, and is important part of the program it must be described here.

```
def is_checked(item):
    check = cv2.imread("check.jpg")
    gray_check = cv2.cvtColor(check, cv2.COLOR_BGR2GRAY)
    match_param  = match_template(check, item)
    menu_check = item.copy()

    menu_check = crop_this(menu_check, match_param)

```

```
    kernel = np.ones((5,5),np.uint8)
    modval=100
    (t, maskLayer) = cv2.threshold(src = check,
            thresh = modval,
            maxval = 170,
            type = cv2.THRESH_BINARY)
    erosion = cv2.erode(maskLayer,kernel,iterations = 1)
    (t2, maskLayer2) = cv2.threshold(src = menu_check,
            thresh = modval,
            maxval = 170,
            type = cv2.THRESH_BINARY)
    erosion2 = cv2.erode(maskLayer2,kernel,iterations = 1)
    print()
    print (np.mean(erosion), np.mean(erosion2))
    if np.mean(erosion) > np.mean(erosion2*1.04):
        return True
```

To put it briefly, pixels brighter than 170 are masked out, and only darker pixels are stored in the =maskLayer=s, after which erosion[5] of the layers is done. After that the comparision of the checkbox from the cropped out section of the filled menu is compared with the blank checkbox. If the mean pixels in the filled menu's checkbox is significantly darker it is considered to be ticked.

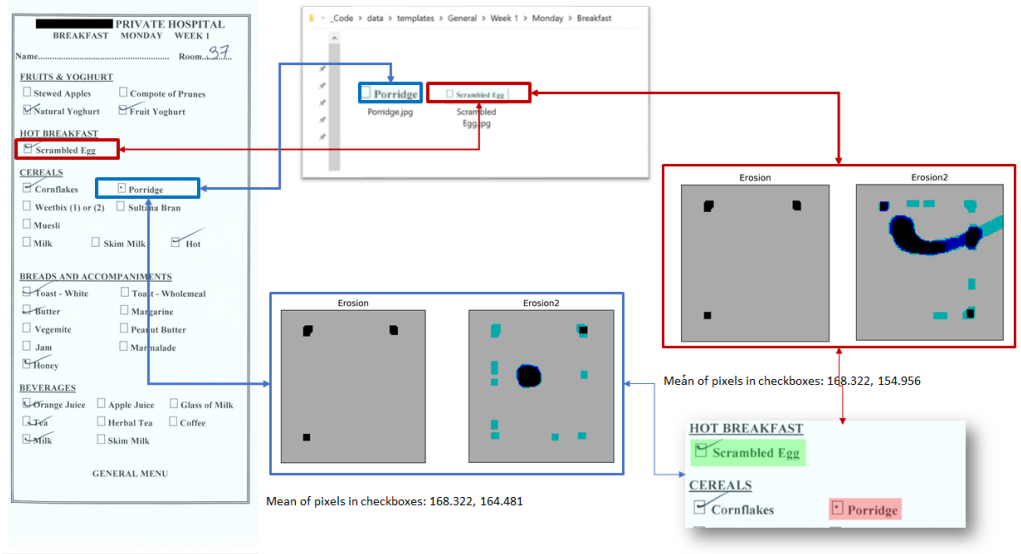


Figure 15: OMR alrogithm works by comparing the mean of pixels to identify if a box is ticked.

9.2.1 python-docx: Finding the best values for accurate recognition

The values `maxval`, `modval`, `iterations` and comparision value of `1.04` used on the above function are sensitive and may require tweaking depending on the conditions to reduce false positives and false negatives. These are significant variables and determine the functioning of the function, and the program itself.

9.3 Redis

Redis is a No-SQL database which does not require table stuructures and relationships unlike the traditional SQL programs like MySQL, SQLite. This advantage allowed for quick integration and implementation of counting of the ticked items in the menu. It is used by 14.4 to store the ticked items and by 14.6 to read the database and generate a report, as shown below:

[5]Erosion removes the boundary pixels of foreground by changing the pixels to match the boundary pixels.

```
127.0.0.1:6379> keys *
 1) ”Egg and Asparagus Mornay.jpg:list”
 2) ”Broccoli.jpg:count”
 3) ”Scrambled Egg.jpg:list”
 4) ”Scrambled Egg.jpg:name”
 5) ”Steak Diane.jpg:count”
 6) ”Mashed Potato.jpg:count”

127.0.0.1:6379> get ”Mashed Potato.jpg:count”
”4”
```

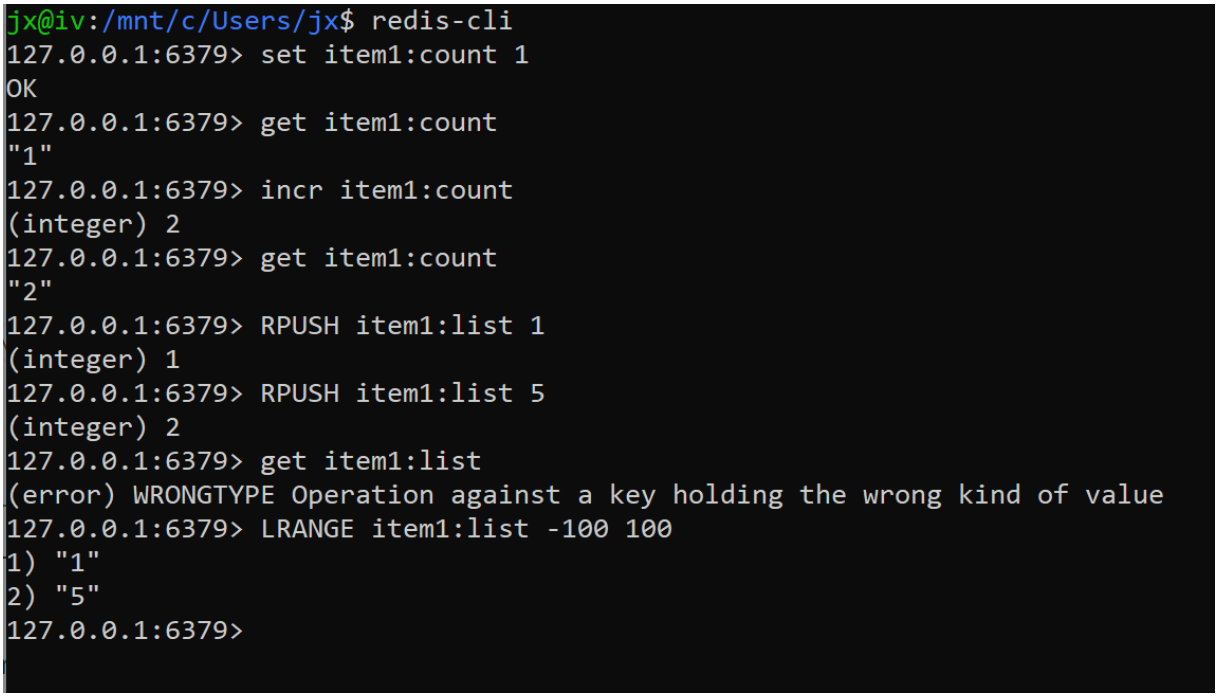


Figure 16: A simple demonstration of how easily one can store and retrieve data from redis.

9.4 python-docx

Python-docx is a python library that allows reading and writing to a word `.docx` filetype. The choice of word document is intentional as it is primarily used by the staff in the organization.

9.4.1 Generating report

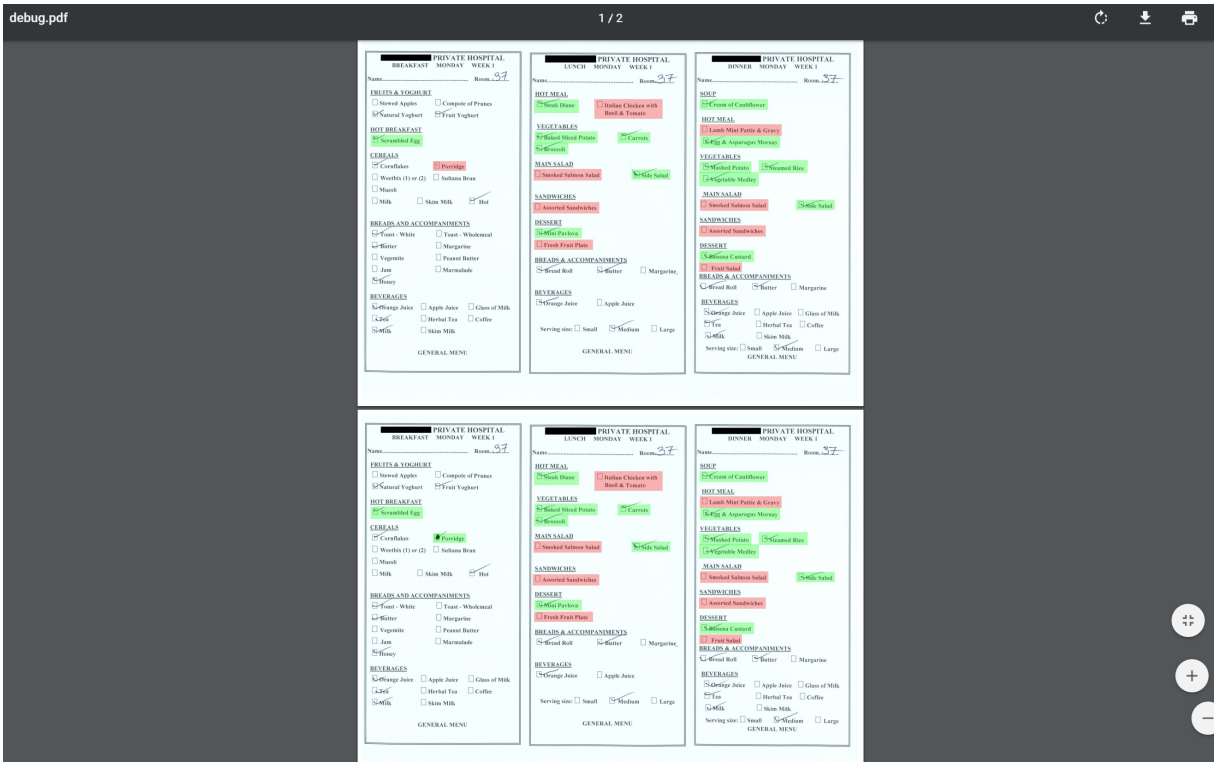
`write()` function in 14.6 is used for generating a report at the end, which facilitates the diet aide. The report consists of three pages, one for each section of menu (breakfast, lunch and dinner), with counts for each item. She can then review the `debug.pdf` file produced by 14.8 to review any false positives or false negatives and make changes to the report produced easily. This is the main desired output from this project.

The output generated by the program looks like:

Notice the ”No Eggs” this can be seen because the `patient_list.docx` for Room 3 contains allergy infromation.

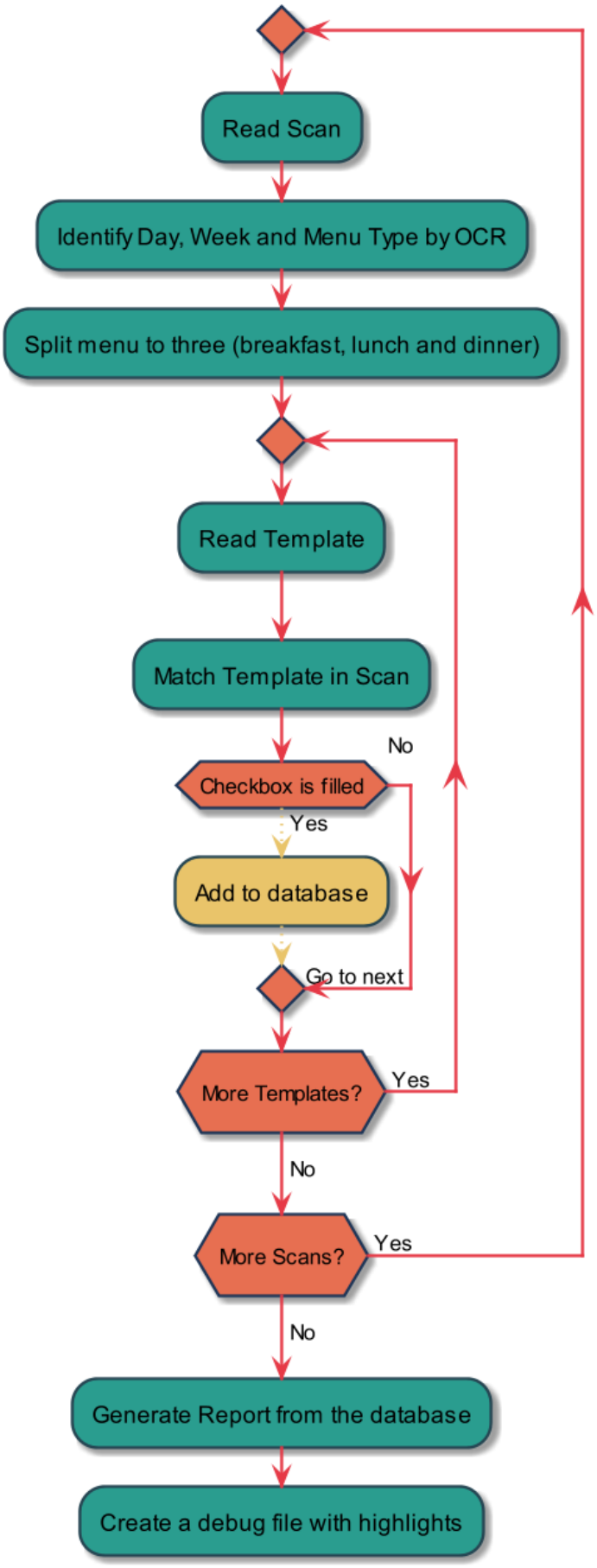
9.5 Informational PDF

A PDF filoel containining highlights is also created for helping review any false positives or false negatives during the process, this is generated by 14.8 andis stored as `debug.pdf` in the project directory.



10 Program Flow

The following flowchart describes the flow of the program which utilizes the components mentioned in the section 9 and can be reviewed using the code presented in the section 14. Another benefit of this application and the way it is created is that various functions and files have their own functioning and can be modified without affecting the rest of the program.


Read Scan
Identify Day, Week and Menu Type by OCR
Split menu to three (breakfast, lunch and dinner)
Read Template
Match Template in Scan
No
Checkbox is filled
Yes
Add to database
Go to next
More Templates?
Yes
No
More Scans?
Yes
No
Generate Report from the database
Create a debug file with highlights

11 Assumptions and Requirements

The project makes the following assumptions in order to function properly:

- The templates need to be the same DPI[6] as the scans in order to make proper matches.
- A user checks the output report file as well as the debug pdf containing all the scans to identify any false negatives after the program is fininshed counting the menus.
- This project assumes that no currently exisiting procedures are changed, i.e. it aims to perfom the tasks with least distruptions (for example: the layout of the menus do not need changing).

12 Future Work / Recommendations

The program is not complete in a sense that in order to completely replace the knowledge of the diet aide, there must be more work put into the program as the diet aide has some formal knowledge of dietetics.

Some more functionalites that can be added in the future are:

- Handwriting recognition neural network can be trained using the menus, by cropping out the room numbers so identify room numbers written in the menu, currently the menus need to be sorted and are matched with the list of paitents.
- More knowledge of food items by adding recepies can help furthur reduce diet aide involvement.

13 Conclusion

Through this project, I was able to demonstrate how the program can automate the job of a diet aide or assist them in making their jobs easier by use of existing devices such as a scanner and a computer, as well as by using free and open source programs and libraries.

The program is able to perfrom several functions, which are:

- Read a document (List of paitents) to identify number of patients
- Read the scanned menus according to the list of patients
- Identify the day of the week, to look for related menu templates
- ”Count” or store the items into a database
- Identify allergies
- Generate a word document with the diet needs mentioned

I would like to remind the assessor to not overlook the complexity of the program and the various functions. I would like to arguee that my project is much more complex than presented by other students and this can be seen though the amount of work I’ve put into the code.

14 Appendix

14.1 Structure

The program is broken down into following files:

```
/
├── main.py
└── modules
    ├── database.py
    ├── deitcheck.py
    ├── document.py
    ├── ocr.py
    ├── omr.py
    └── pdf.py
```

[6] This project uses 600dpi scans

14.2 main.py

```
import modules.ocr as ocr
import modules.database as db
import modules.document as doc
import modules.pdf as pdf
import modules.omr as omr
import imutils
import cv2
import numpy as np
import os
# how having closely cropped tempaltes affect the matching (crop too closely and its gone)
# how having ddifferent threshold affets stuff

scans = os.listdir('scans/')
patients = doc.read()
prefix = ""
db.init()

def menu_type(text):
    if "GENERAL MENU" in text:
        return "General"
    elif "DIABETIC MENU" in text:
        return "Diabetic"
    elif "GLUTEN FREE MENU" in text:
        return "Gluten Free"

def week_day(text):
    def week():
        if "WEEK 1" in text:
            return "Week 1"
        elif "WEEK 2" in text:
            return "Week 2"
    def day():
        if "MONDAY" in text:
            return "Monday"
        elif "TUSEDAY" in text:
            return "Tuesday"
        elif "WEDNESDAY" in text:
            return "Wednesday"
        elif "THURSDAY" in text:
            return "Thursday"
        elif "FRIDAY" in text:
            return "Friday"
        elif "SATURDAY" in text:
            return "Saturday"
        elif "SUNDAY" in text:
            return "Sunday"
    return week() + "/" + day()


for filename, info in zip(scans, patients):

    scan = cv2.imread("Scans/"+filename)

    gray_menu = cv2.cvtColor(scan, cv2.COLOR_BGR2GRAY)

```

```
    # https://docs.opencv.org/master/d3/df2/tutorial_py_basic_ops.html
    # Shape returns number of rows (height) and columns (Width) and channels (none if gray)
    h,w = gray_menu.shape[:2]
    if h > w:
        # Rotate Image
        #
        ↪  https://stackoverflow.com/questions/43892506/opencv-python-rotate-image-without-cropping-sides/47248339
        gray_menu = imutils.rotate_bound(gray_menu, 90)
        menu_w = int(h/3)
        menu_h = int(w)
    else:
        menu_w = int(w/3)
        menu_h = int(h)


    menu_names = ["breakfast", "lunch", "dinner"]
    menus = {}
    menu_text =""
    week = ""
    for i in (range(len(menu_names))):

        temp = scan.copy() #change to gray_menu to use only one channel
        if i == 2:
            #
            ↪  https://stackoverflow.com/questions/6181935/how-do-you-create-different-variable-names-while-in-a-loop
            # crop_img = img[y:y+h, x:x+w]
            menus["{}".format(menu_names[i])] = temp [0:menu_h, menu_w*i-50:menu_w*(i+1)]
        else:
            menus["{}".format(menu_names[i])] = temp [0:menu_h, menu_w*i:menu_w*(i+1)] # (x1:x2, y1:y2) which
            ↪  form a rectangle
        print(menus["{}".format(menu_names[i])].shape)
        snip_top = menus["{}".format(menu_names[i])][int(0.8*menu_h):menu_h, 0:menu_w]
        menu_text = menu_text.join(ocr.read(snip_top))
        snip_bottom = menus["{}".format(menu_names[i])][0:int(0.2*menu_h), 0:menu_w]
        week = week.join(ocr.read(snip_bottom))


    menu_text = menu_type(menu_text)
    week = week_day(week)
    prefix = menu_text + "/" + week + "/"

    print(prefix, info)
    for i in menu_names:
        # Snips bottom half of menu for menu type (general, diabetic, gluten free)


        print(menu_text)
        omr.read_templates(prefix+i,menus[i],menu_text, info)

    combined = np.hstack((menus["lunch"], menus["dinner"]))
    # adjust the correction of 50 in dinner menu, due to geomentry, more borders towards side
    dinner_adjustments = menus["dinner"].shape[:2]
    menus["dinner"] = menus["dinner"][0:dinner_adjustments[0], 50:dinner_adjustments[1]]
    combined = np.hstack((menus["breakfast"],menus["lunch"], menus["dinner"]))
    combined= cv2.resize(combined,None,fx=0.25,fy=0.25)
    cv2.imwrite('debug/{}'.format(filename), combined)

```

```
doc.write(prefix)
pdf.create()
```

14.3 database.py

```
import redis
import modules.deitcheck as diet

# FOR REDIS
redis_host = "localhost"
redis_port = 6379
redis_password = ""
prev_item=""

r = redis.StrictRedis(host=redis_host, port=redis_port, password=redis_password, decode_responses=True)

def update(item, menu_type, pt_info):
    """
        pt_info: {'Room': '1', 'Patient Name': 'John Doe', 'Comments': 'Lactose Intolerant'}
    """
    try:
        # The decode_repsonses flag here directs the client to convert the responses from Redis into Python
        ↪  strings
        # using the default encoding utf-8.  This is client specific.

        print ("UPDATE INITIATION")
        print (item, menu_type, pt_info)
        note = diet.check(item, menu_type, pt_info)
        print(r.keys("*:name"))
        print("msg")
        msg = r.get("{}:name".format(item))
        if not msg:
            r.set("{}:name".format(item), item)
        r.incr("{}:count".format(item))
        if note:
            r.rpush("{}:list".format(item), pt_info["Room"]+note)
        else:
            r.rpush("{}:list".format(item), pt_info["Room"])
        print("count")
        count = r.get("{}:count".format(item))
        print("{} exists and the count is {}".format(item, count))
        ptlist = r.lrange("{}:list".format(item), -100, 100)
        print("{} exists and the list is {}".format(item, ptlist))
        print("{} is created and the count is {}".format(item, count))

    except Exception as e:
        print(e)

def init():
    r.delete("Contains:*")
    r.rpush("Contains:Tomato", "Italian Chicken with Basil and Tomato", "Side Salad", "Smoked Salmon Salad")
    r.rpush ("Contains:Eggs", "Scrambled Egg", "Egg and Asparagus Mornay")
    r.rpush ("Contains:Gluten", "Chicken Parmigiana", "Crumbled Fish")
    print("Database initialized")
```

14.4 omr.py

```
import os
import cv2
import numpy as np
import imutils
import modules.database as db


def plot_this(title, image):
    """
    Only for Jupyter Notebook. Moved to show.py in modules
    """
    pass

def match_template(template, image):
    """
    Matches just one template to any supplied image returns the match percentage and the location,
    as well as the shape (h,w) of the template image
    """
    #needs a grayscale image, template
    (t, maskLayer) = cv2.threshold(src = template,
        thresh = 200,
        maxval = 255,
        type = cv2.THRESH_BINARY)
    # Find shape (resolution) of template and match it in the scan
    h, w = template.shape[:2]
    result = cv2.matchTemplate(image, maskLayer, cv2.TM_CCOEFF_NORMED)
    #https://stackoverflow.com/questions/36040630/how-to-interpret-matchtemplate-output-opencv-python
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result) # max_loc for best match probably
    return (max_val, max_loc, h, w)

def draw_rectangle(image, param, color):
    """
    Draws a rectangle around a given template
    color needs to be an array (0,255,0) for green, (255,0,0) for red
    """
    overlay = image.copy()
    # transparency value
    alpha = 0.3
    # creates an copy of the image and makes an opaque box in the area and then makes it transparent
    cv2.rectangle(overlay, param[1], (param[1][0] + param[3], param[1][1] + param[2]),color, -1)
    #cv2.rectangle(image, param[1], (param[1][0] + param[3], param[1][1] + param[2]), color, 3)
    # adds the overlay to the original image
    cv2.addWeighted(overlay, alpha, image, 1 - alpha, 0, image)
    # for more info https://www.pyimagesearch.com/2016/03/07/transparent-overlays-with-opencv/


def crop_this(image, param):
    #param is (max_val, max_loc, h, w) else is false
    image = image[param[1][1]:param[1][1]+param[2], param[1][0]:param[1][0]+param[3]]
    return image



def read_templates(folder, image, diet, pt_info):
    """
```

```
    Reads the templates folder and calls match_template, which returns the best match for each template.
    It then:
    - Calls draw rectangle, around each template in the folder
    - Crops the template on the image
    """
    entries = os.listdir('data/templates/{}'.format(folder))


    for filename in entries:
        template = cv2.imread("data/templates/{}/".format(folder)+filename)
        template_gray = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)
        match_param  = match_template(template, image)
        # match_param is (max_val, max_loc, h, w) else is false

        item = crop_this(image, match_param)

        if (is_checked(item)):
            draw_rectangle(image, match_param, (0,255,0))
            print ("{} is checked".format(filename))
            # filename  eg: Steamed Rice.jpg
            db.update(filename, diet, pt_info)
        else:
            draw_rectangle(image, match_param, (0,0,255))
            print ("{} is not checked".format(filename))


def is_checked(item):
    check = cv2.imread("check.jpg")
    gray_check = cv2.cvtColor(check, cv2.COLOR_BGR2GRAY)
    match_param  = match_template(check, item)
    menu_check = item.copy()

    menu_check = crop_this(menu_check, match_param)

    kernel = np.ones((5,5),np.uint8)
    modval=100
    (t, maskLayer) = cv2.threshold(src = check,
            thresh = modval,
            maxval = 170,
            type = cv2.THRESH_BINARY)
    erosion = cv2.erode(maskLayer,kernel,iterations = 1)
    (t2, maskLayer2) = cv2.threshold(src = menu_check,
            thresh = modval,
            maxval = 170,
            type = cv2.THRESH_BINARY)
    erosion2 = cv2.erode(maskLayer2,kernel,iterations = 1)
    print()
    print (np.mean(erosion), np.mean(erosion2))
    if np.mean(erosion) > np.mean(erosion2*1.04):
        return True
```

14.5 ocr.py

```
import cv2
import pytesseract

# Set Tesseract Executable location
```

```
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

#text = pytesseract.image_to_string(cv2.imread('test/ocr.jpg'))

# print(text)

def read(image):
    return pytesseract.image_to_string(image)
```

14.6 document.py

```
import docx
import redis
import os


# FOR REDIS
redis_host = "localhost"
redis_port = 6379
redis_password = ""
prev_item=""


filename = "patient_list.docx"
def read():
    """
    Reads the document and returns list of patients, skips where name is blank

    Sample Output: [{'Room': '1', 'Patient Name': 'John Doe', 'Comments': 'Lactose Intolerant'},
                {'Room': '3', 'Patient Name': 'Jane Doe', 'Comments': 'Allergy: Tomato, Onions'}]
    """
    doc = docx.Document('data/{}'.format(filename))
    # https://stackoverflow.com/questions/27861732/parsing-of-table-from-docx-file
    table = doc.tables[0]
    # Data will be a list of rows represented as dictionaries
    # containing each row's data.
    data = []

    keys = None
    for i, row in enumerate(table.rows):
        text = (cell.text for cell in row.cells)

        # Establish the mapping based on the first row
        # headers; these will become the keys of our dictionary
        if i == 0:
            keys = tuple(text)
            continue

        # looks at the patent name, if empty it is skipped
        check_empty = row.cells[1]

        if check_empty.text:
            # Construct a dictionary for this row, mapping
            # keys to values for this row
            row_data = dict(zip(keys, text))
            data.append(row_data)
    return data
```

```
47
48
49
50  def write(prefix):
51      menu_names = [”Breakfast”, ”Lunch”, ”Dinner”]
52      r = redis.StrictRedis(host=redis_host, port=redis_port, password=redis_password, decode_responses=True)
53      filename = ”report.docx”
54      report = docx.Document()
55      keys = r.keys(”*:count”)
56      for i in menu_names:
57          if i != ”Breakfast”:
58              report.add_page_break()
59          report.add_heading(i, 0)
60          entries= os.listdir('data/templates/{}'.format(prefix+i+”/”))
61          for j in keys:
62              nicekeys = j.split(”:”)
63              if nicekeys[0] in entries:
64                  count = r.get(”{}”.format(j))
65                  getlist = nicekeys[0]+”:list”
66                  print(getlist, nicekeys[0])
67                  lister = r.lrange(nicekeys[0]+”:list”, -100, 100)
68                  print(lister)
69                  name = j.split(”.”)
70                  name = name[0]
71                  print(name,count)
72                  report.add_paragraph('{} {}:{}'.format(name, count, ', '.join(lister)))
73
74      report.save(filename)
```

14.7 dietcheck.py

```
1   import redis
2
3
4
5   # FOR REDIS
6   yredis_host = ”localhost”
7   redis_port = 6379
8   redis_password = ””
9   prev_item=””
10  r = redis.StrictRedis(host=redis_host, port=redis_port, password=redis_password, decode_responses=True)
11
12  def check(item, diet, info):
13      # Italian Chicken with Basil and Tomato.jpg General {'Room': '3', 'Patient Name': 'Jane Doe', 'Comments':
        ↪  'Allergy: Tomato, Onions'}
14      keys = r.keys(”Contains:*”)
15      print (”Checking Diet”)
16      for i in keys:
17          food_items = r.lrange(i, -100, 100)
18          name = i.split(”:”) # Contains:Tomato
19          item = item.split('.') # Vegetable Medley.jpg
20          # if 'Tomato' in 'Allergy: Tomato, Onions' and 'Italian Chicken with Basil and Tomato' in
            ↪  ”Contains:Tomato”
21          print (”testing if {} in {} and {} in {}”.format(name[1], info[”Comments”], item[0], food_items))
22          if name[1] in info[”Comments”] and item[0] in food_items:
23              return ” (No {})”.format(name[1])
24          else:
```

```
            return str('')
```

14.8 pdf.py

```
import os
import img2pdf

newlist=[]
def create():
    # https://stackoverflow.com/questions/27327513/create-pdf-from-a-list-of-images
    with open("debug.pdf", "wb") as f:
        debug = os.listdir('debug/')
        for i in debug:
            newlist.append('debug/'+i)

        f.write(img2pdf.convert(newlist))
```

References

A box detection algorithm for any image containing boxes. (n.d.). https://medium.com/coinmonks/a-box-detection-algorithm-for-any-image-containing-boxes-756c15d7ed26. ((Accessed on 09/06/2019))

Build your first redis hello world application in python | opensource.com. (n.d.). https://opensource.com/article/18/4/how-build-hello-redis-with-python. ((Accessed on 11/24/2019))

Davison, O. C., Berutti, V. F., Holenstein, B. D., Holenstein, P. J., & Hoffmann, T. E. (2012, December 11). *Inferential self-registration of imperfect omr forms.* Google Patents. (US Patent 8,331,740)

Enterprise Apps Tech News. (n.d.). *What ITIL 4 means for AI and automation in IT service management.* https://www.enterprise-cio.com/news/2019/mar/11/what-itil-4-means-ai-and-automation-it-service-management/. ((Accessed on 11/18/2019))

Food ordering system for patients | alpha hros. (n.d.). https://www.alphaebm.com/hospital-food-service-software-uae.html. ((Accessed on 09/03/2019))

Forbes. (2016, September). *Doctors wasting over two-thirds of their time doing paperwork.* https://www.forbes.com/sites/brucelee/2016/09/07/doctors-wasting-over-two-thirds-of-their-time-doing-paperwork/#5c3f48d95d7b. ((Accessed on 11/18/2019))

How to crop an image in opencv using python - stack overflow. (n.d.). https://stackoverflow.com/questions/15589517/how-to-crop-an-image-in-opencv-using-python. ((Accessed on 11/24/2019))

How to display a matplotlib rgb image - pyimagesearch. (n.d.). https://www.pyimagesearch.com/2014/11/03/display-matplotlib-rgb-image/. ((Accessed on 11/24/2019))

How to use opencv imshow() in a jupyter notebook — quick tip. (n.d.). https://medium.com/@mrdatainsight/how-to-use-opencv-imshow-in-a-jupyter-notebook-quick-tip-ce83fa32b5ad. ((Accessed on 11/24/2019))

Indeed.com. (n.d.). *(dietary aide job description examples.*

Omr vs. ocr – difference and comparison – diffzi. (n.d.). https://diffzi.com/omr-vs-ocr/. ((Accessed on 09/03/2019))

opencv - how to draw a rectangle around a region of interest in python - stack overflow. (n.d.). https://stackoverflow.com/questions/23720875/how-to-draw-a-rectangle-around-a-region-of-interest-in-python. ((Accessed on 11/24/2019))

Opencv python : rotate image without cropping sides - stack overflow. (n.d.). https://stackoverflow.com/questions/43892506/opencv-python-rotate-image-without-cropping-sides/47248339. ((Accessed on 11/24/2019))

PyImageSearch. (2016, October). *Bubble sheet multiple choice scanner and test grader using omr, python and opencv.* https://www.pyimagesearch.com/2016/10/03/bubble-sheet-multiple-choice-scanner-and-test-grader-using-omr-python-and-opencv/. ((Accessed on 11/19/2019))

python-docx — python-docx 0.8.10 documentation. (n.d.). https://python-docx.readthedocs.io/en/latest/. ((Accessed on 11/24/2019))

python - how do you create different variable names while in a loop? - stack overflow. (n.d.). https://stackoverflow.com/questions/6181935/how-do-you-create-different-variable-names-while-in-a-loop. ((Accessed on 11/24/2019))

python - increase dpi of matplotlib .show() in jupyter notebook - stack overflow. (n.d.). https://stackoverflow.com/questions/51937381/increase-dpi-of-matplotlib-show-in-jupyter-notebook?rq=1. ((Accessed on 11/24/2019))

python - parsing of table from .docx file - stack overflow. (n.d.). https://stackoverflow.com/questions/27861732/parsing-of-table-from-docx-file. ((Accessed on 11/24/2019))

Python tutorial: Dictionaries. (n.d.). https://www.python-course.eu/python3_dictionaries.php. ((Accessed on 11/24/2019))

Python | working with .docx module - geeksforgeeks. (n.d.). https://www.geeksforgeeks.org/python-working-with-docx-module/. ((Accessed on 11/24/2019))

Rescueomr: batch optical mark recognition without foresight. (n.d.). https://www.thregr.org/~wavexx/software/RescueOMR/. ((Accessed on 09/03/2019))

Taylor, G. S. (2004, May 25). *Method of optical mark recognition.* Google Patents. (US Patent 6,741,738)

Template matching — opencv 2.4.13.7 documentation. (n.d.). https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html. ((Accessed on 09/06/2019))

tesseract-ocr/tesseract: Tesseract Open Source OCR Engine (main repository). (n.d.). https://github.com/tesseract-ocr/tesseract. ((Accessed on 11/24/2019))

The Independent. (2018, May). *Nurses made to 'choose between paperwork and patient care' because of staff shortages, RCN warns.* https://www.independent.co.uk/news/health/nurse-nhs-staff-patients-paperwork-bureaucracy-healthcare-funding-crisis-a8347611.html. ((Accessed on 11/18/2019))

Transparent overlays with opencv - pyimagesearch. (n.d.). https://www.pyimagesearch.com/2016/03/07/transparent-overlays-with-opencv/. ((Accessed on 11/24/2019))

Zitova, B. (2019). Mathematical approaches for medical image registration.