City St George's University of London

BSc (Hons) Computer Science

IN3007 - Individual Project

Academic Year: 2024-25

Exploring the Feasibility of AI-Driven Receipt Scanning for Food Tracking

Fatima Ahmed

Student ID 210008985

ABSTRACT 5
CHAPTER 1 INTRODUCTION 6
1.1 Problem Statement 6
1.2 Understanding food waste and behavioural gaps 7
1.3 How can receipt data be used to track food items automatically? 8
1.2 Primary Objectives 10
1.4 Anticipated Beneficiaries and impact 10
1.5 Outline of Work Performed 10
1.5.1 OCR Tool Evaluation 10
1.5.2 Receipt Image Preprocessing 10
1.5.3 OCR Text Extraction 11
1.5.4 Rule-based Text Extraction 11
1.5.5 Fuzzy Matching for matching to database 11
1.5.6 Interface 11
1.6 Assumptions and Scope Limitations 11
CHAPTER 2 OUTPUT SUMMARY 12
CHAPTER 3 CONTEXT 14
3.1 Introduction 14
3.2 Current Systems 14
3.3 Introduction to OCR 14
3.3.1 Preprocessing 15
3.3.2 Regular Expressions 15
3.3.3 post-processing 15
3.3.4 String Matching 16
3.3.5 Relevant studies 16
3.4 Web Development: Flask, HTML, CSS and javascript 16
3.5 Conclusion 16
CHAPTER 4 METHOD 17
4.1 Software Engineering Method 17
4.1.2 Incremental Development Approach (Sprint-Based Methodology) 17
4.2 Management Tools 18
4.2.1 System Preparation , tools and software's used 18
4.3 Dataset collection 19
4.4 Project Requirements and analysis 19
4.4.1 Functional and Non-Functional Requirements 20
4.4.1.1 Use Case Diagram 21

4.4.1.2 ERD Diagram 21
4.4.1.3 State Diagram 21
4.4.1.4 User Interface Design 21
4.5 Testing and Evaluation 22
4.5.1 Sprint 1 22
4.5.2 Sprint 2 22
4.5.3 Sprint 3 22
4.5.4 Sprint 4 23
4.5.5 Sprint 5 23
4.5.1 Legal, Social, Ethical and Professional Considerations 23
4.6 Interface Implementation 24
4.6.1 Flask Framework 24
4.6.2 Front-End Interface 24
4.6.3 Database integration 24
4.7 Reused material 25
4.8 user testing 25
4.9 Project Deviations 25
CHAPTER 5 RESULTS 25
5.1 Dataset Collection 25
5.2 Recording of Results 25
5.3 Sprint 1 results 25
5.4 Sprint 2 results 38
5.5 Sprint 3 results 70
5.6 Sprint 4 results 73
5.6.1 Design 73
5.6.1.1 Use case diagram 73
5.6.1.2 Entity Relationship Diagram 73
5.6.1.3 State Diagram 73
5.6.1.4 User Interface Design 74
5.6.2 Project Setup 75
5.6.2.1 Dependencies 75
5.6.2.1 Folder Structure and File Organization 75
5.7 Implementation 76
5.7.1 Flask Web Application Development 76
5.7.1.2 Flask routing 76
5.7.1.3 OCR and AI Pipeline integration 76

5.7.1.4 Fuzzy Matching and Database Logic 78
5.7.1.5 Jinja2 Templating and HTML rendering 79
5.7.2 HTML Pages and UI Functionality 79
5.7.2.1 Index page 80
5.7.2.2 Expiry Notifications system 80
5.7.2.3 Manual input entry – fallback 81
5.7.2.2 Pantry Page 82
5.8 Sprint 5 results 84
5.8.1 OCR extraction on varied receipts 84
5.8.2 edge detection exploration 85
5.9 User testing 88
CHAPTER 6 CONCLUSION 89
6.1 Objectives and research questions revisited 89
6.2 Limitations and Reflections 89
6.3 Implications for Commercialization and Future Work 90
6.4 Legal, Social, Ethical and professional issues 90
6.5 Personal Reflections 91
Appendix A - PDD 92
Appendix B – Reused Summary 106
Appendix C – Receipt Dataset 120
Appendix D – Ai pipeline Sprints folder and interface 122
Appendix E – comparison of OCR Tools 123
Appendix F – Updated Objectives and Research questions 126
Appendix G – Calculating OCR accuracy 128
Appendix H – Key Deviations summary 130
Appendix I – Web development With Flask 132
Appendix J – Regex Filtering 134
Appendix K – String matching additional research 136
Appendix L– Fuzzy matching method 139
Appendix M – User Testing Results 139
Appendix N - Requirements 145
Appendix O – Research Questions and Finding 150
References 154

ABSTRACT

Food waste continues to be a global issue, where households currently waste the most food and so to manage this, food tracking apps were developed to help people manage their inventory however the main issue with the apps is that it, relies on updating each item manually one by one / scanning each items barcode which can be very long and tiring , hence making it inconvenient for users therefore leading users abandoning the apps. therefore, there is a growing need for AI to effortlessly automate a user's pantry with minimal user effort . This project explores the feasibility of using ai to manage a user's pantry where users can automatically update their inventory using a receipt using OCR to read the image and extract the food items and match to a food database to get the expiry date and update their pantry. The report details the methods applied across the project's lifecycle, including all the testing and techniques to then developing the simple interface. The project then concludes with evaluation results and discussions of the findings of the feasible test.

CHAPTER 1 INTRODUCTION

1.1 Problem Statement

Food waste is a global issue with significant environmental implications. In the UK alone, households contribute approximately 6.6 million tonnes of total food waste annually (Dray,2021). A significant portion of this waste is avoidable and often results from poor inventory tracking-users simply forget what they have bought or let items expire unnoticed.

This is a problem, so in response to this, pantry management apps have been developed however, their core limitation lies in how they expect users to engage with them.

Most systems rely on **manual input or barcode scanning**, placing the responsibility on users to consistently log each item they purchase. Whether through typing in food names and expiry dates or scanning items one-by-one, the process is repetitive, time-consuming, and often abandoned after initial use. Despite good intentions, users rarely maintain the habit of updating their pantry daily. There is no automation, just input forms and a growing sense of inconvenience. These systems do not automate whereas receipts contain a complete list of food items bought and so could serve as a key object to effortless inventory logging, so ai-powered solutions have a potential to fulfil automation. An Ai-powered pipeline can turn a simple user action by simply taking a picture of a receipt into a complete pantry update. The flowchart in Figure 1 illustrates this traditional process.

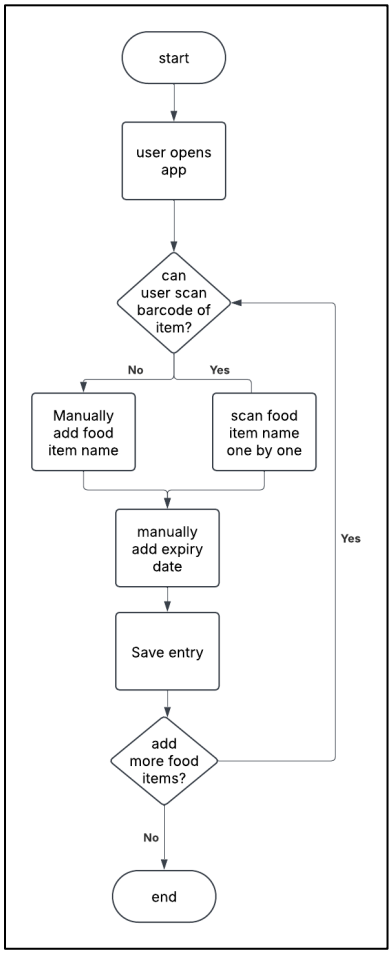


Figure 1 *current process for manually adding food items to food waste apps*

1.2 Understanding food waste and behavioural gaps

Food waste is a complex issue and preventing it at the household level requires a deeper understanding of the everyday behaviours that lead to it. Research shows that food waste does not result from a single action, but rather from a combination of behaviours across planning, shopping, storage, preparation, and consumption (Quested et al., 2011). These behaviours are shaped by household routines, limited awareness of expiry dates, and a lack of visibility into current food inventory. Often, by the time food is discarded, the opportunity for intervention has already passed, highlighting the need for **early, low-effort interventions** that fit naturally within people's daily habits.

To understand these behaviours, multiple research methods have been employed. These include food waste diaries, in-home ethnographic studies, and consumer questionnaires, surveys. Each of these methods contributes insight into waste generation, consumer attitudes, and behavioural patterns. The findings from these approaches have been used to develop a conceptual framework to illustrate how and when food waste occurs (Quested et al., 2011), as shown in **Figure 1.1**.

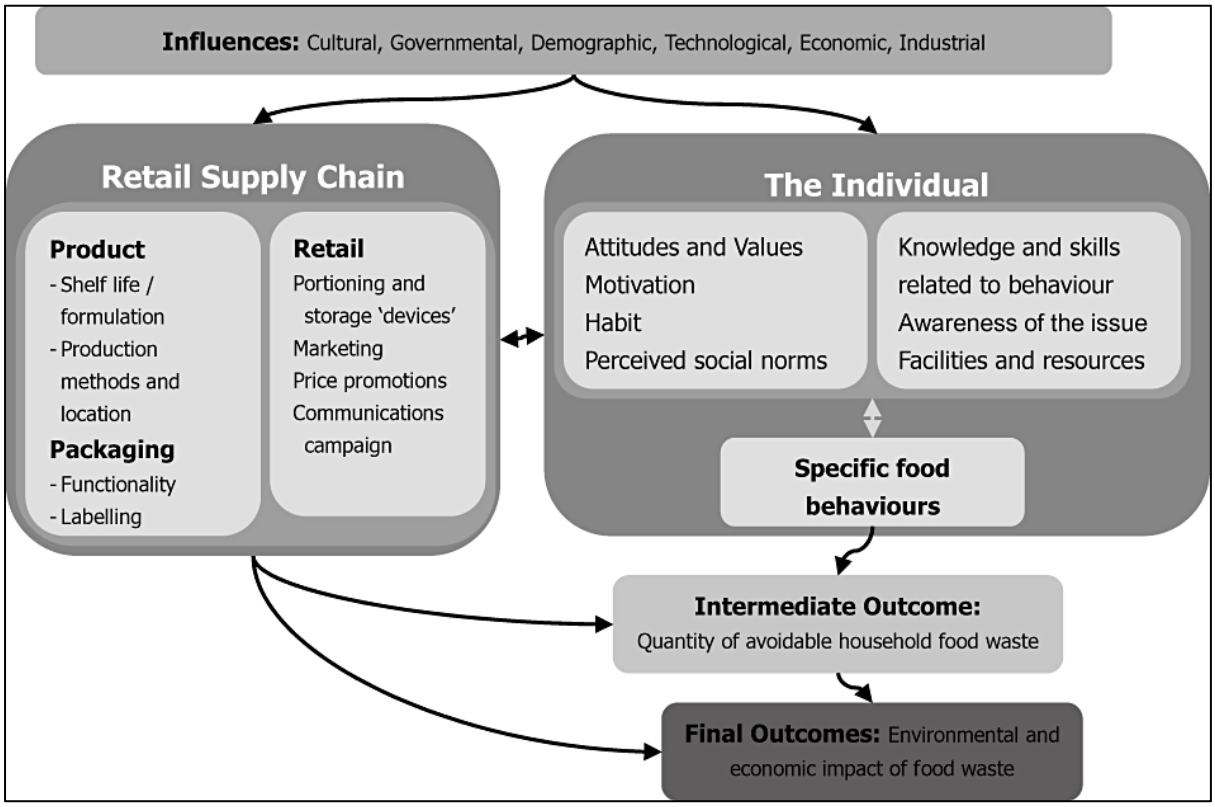


Figure 1.1 *conceptual framework to understand prevention of food waste*

The framework suggests that the decision to throw away food is typically the result of earlier missed opportunities-such as overbuying, poor planning, or forgetting about perishable items. However, despite the value of this research, many of the data collection methods are subject to limitations. Food diaries, for example, often lead to under-reporting by as much as 40%, and may influence participants behaviour simply by making them more aware of their waste (Quested et

al., 2011). Similarly, surveys may not accurately capture routine and habitual actions, which poses a challenge for long-term food waste reduction strategies.

Given these behavioural insights, it becomes clear that expecting users to monitor their food inventory consistently and manually is unrealistic. Systems that rely on manual input, such as typing in expiry dates or scanning individual items are often abandoned because they are not well-aligned with real-world behaviours. Effective food waste interventions must therefore minimise user effort and integrate smoothly into existing routines, especially those immediately following shopping.

This project responds to that need by proposing an automated, behaviourally aligned method of food tracking, rather than requiring the user to self-monitor. This approach reflects the conclusions of WRAP's research: that sustainable outcomes start with making better behaviours easier ,not harder to maintain by using automation at the point of purchase and turning data from the receipt into insights.

1.3 How can receipt data be used to track food items automatically?

Receipts typically contain a structured record of grocery purchases, including product names, prices, brand names, and other useful details that can be leveraged to track food items. With recent advancements in artificial intelligence, particularly within the field of computer vision, It is now possible to extract this information directly from receipt images. A key technique used for this purpose is **Optical Character Recognition (OCR)**, which enables systems to identify printed text, even from low-quality scans and varying font styles (Vina, 2024). This is particularly important as receipt formats and print quality often differ between supermarkets, making robustness a critical requirement for automated processing in the future. Furthermore, using receipt data aligns with everyday shopping habits and requires minimal change from users.

Figure 1.2 *example receipt image*

1.2 Primary Objectives

The primary objective of this project is to evaluate the feasibility of using AI-driven receipt scanning for automated food tracking. The system extracts the food items from shopping receipts, matches the items against a food database to retrieve it's predicted expiry date so it automatically updates the users app reducing manual input, and to achieve this I had sub-objectives however these have changes from my initial plan (refer to Appendix A) which led to a more focused approach in particular the changes has been highlighted (Refer to Appendix F for changes).

1.4 Anticipated Beneficiaries and impact

Beneficiaries	benefit
Households and food waste reduction app users	Automation through uploading/scanning receipts can reduce manual data entry, making pantry management easier so more motivation to continue using the app to reduce food waste
Developers of Food tracking and Pantry Management Apps	Insights from this poc can help developers integrate the Ai features into existing applications to enhance user engagement
Retailers	Implementing this technology in retail apps could provide consumers with an effortless way to track their purchases and manage expiry dates, thereby improving customer satisfaction and help reduce food waste

1.5 Outline of Work Performed

To address the project objectives and answer the research questions the following work has been done

1.5.1 OCR Tool Evaluation

A review of existing OCR tools was conducted comparing each to select which tool to use which was guided by comparative analysis (refer to Appendix E)

1.5.2 Receipt Image Preprocessing

Receipt images were pre-processed to improve the quality of text for OCR using Python. Several techniques were used which was discussed in chapter 4 with results shown in chapter 5

1.5.3 OCR Text Extraction

Tesseract OCR was used as the tool to extract text from the pre-processed receipt images using Python.

1.5.4 Rule-based Text Extraction

Rather than using advanced NLP techniques, the project implemented a rule-based approach using regular expressions which the author learnt through literature review to extract food item names from the OCR output and removing unrelated content.

1.5.5 Fuzzy Matching for matching to database

The extracted food items were matched against a food database, (created with SQLite), which contained expiry dates using a rapidFuzz library. A threshold was set to ensure at least 70% of the extracted food items were able to fetch the expiry date for the matched item.

1.5.6 Interface

A user-friendly web app was developed using Flask which allowed users to upload a receipt image to automatically update their pantry with the matched food item with their expiry date.

1.6 Assumptions and Scope Limitations

Several assumptions were made during the development of the project to keep the implementation manageable within the given time frame and resource constraints:

1) **Receipt Format Consistency**
   it was assumed the receipts followed a consistent layout with food items appearing on separate lines however in reality there are variations such as loyalty card lines, discounts, which impacts the accuracy of item extraction

2) **Receipt quality and background**
   Although various receipt format and background was tested, It was limited to one receipt type in the simple interface to show how it would look with one perfect working example.

3) **Single Language Support**
   The development of this project was under the assumption that all receipts would be printed in English despite this tesseract does support multiple languages.

4) **Food item identification only**

   Although the initial plan included extracting retailer name as well since products may have different expiry dates depending on from which brand it was purchased, the final implementation focused solely on identifying food item names to narrow the scope and ensure reliability of the core functionality of extracting food items to get the expiry date.

5) **Generic Expiry dates**
   The expiry dates associated with food items were retrieved from a database the author created where it was not influenced by context such as brand, packaging and storage methods which would be key factors in determining expiry dates neither using a real food database from brands.

CHAPTER 2 OUTPUT SUMMARY

2.1 Receipt Dataset	
Description and type	A set of real-world receipt images were used to evaluate the ai pipeline, where a controlled receipt was selected and the other varied receipts were used to assess robustness
Recipient	Author, academic supervisor
Usage	Used to validate OCR performance under realistic conditions and testing of system robustness
Link	Chapter 5.3 baseline testing , 5.8 technical testing (during sprint 1 and 5) , and Moodle submission sprints image file (.jpg images) and interface receipts file, and Appendix C
2.2 OCR, Preprocessing and Expiry Matching pipeline	
Description and type	Scripts performing OCR extraction, image preprocessing, regex filtering, fuzzy matching to database across 4 main files and database. 900 lines approx. reused (repeated due to testing different parameters) receiptscan.py 107 lines of code reused: edge detection code reused entirely to test different preprocessing technique for sprint 5
Recipient	Author, Supervisor, future researchers
Usage	Serves as testing.
Link	Refer to Appendix B for reused material, Appendix D on how to run the file that was submitted in Moodle, sprints folder for all the source code

2.3 Flask-based Web Application	
Description and type	A prototype web application source code developed using flask for users to upload receipt images to be processed through OCR, preprocessing techniques and fuzzy matching in the dashboard also displaying items expiring soon, with the food content shown in pantry table. The web application used a flask backend for python code, front end with dynamic rendering using jinja2 in HTML, CSS and JS, and a SQLite database where the codes from the sprint were reused approx. 15 lines. Extra 3 python files source code were used to support and test the ai driven receipt, test.py, init_db.py, and SQlite,sql

	test.py 15 lines of code: Script used to see the content of the databases to see if successfully created and queries being confirmed working , init_db.py 27 lines of code: initializes the SQLite food and pantry tables which is required for first time set up to connect with or if it's not working to set up again SQLite.sql : SQL script to populate the food database and pantry database
Recipient	Author, academic supervisor, future developers
Usage	Used as working proof of concept web app for using Ai powered receipt scanning to track food pantry.
Link	See chapter 5.6 , and Moodle submission interface folder and Appendix D to run

2.4 User testing	
Description and type	User testing questionnaire and result summaries
recipient	Author, Supervisor and future researchers
Usage	Used to understand and evaluate the user's opinions for the project.
Link	Appendix M for user testing results and additional submission artifacts for all user testing questionnaire results
2.5 Requirements	
Description and type	Created diagrams and tables with the core components and minimum system functionality
Recipient	Author, supervisor and future developers
Usage	Used to guide the implementation process and defining the system scope
Link	Chapter 4.4, Appendix N

CHAPTER 3 CONTEXT

3.1 Introduction

This chapter reviews existing research , tools relevant to the use of AI-driven receipt scanning for automating pantry management systems. This project requires familiarity with new technologies and concepts, as such extensive reading was required. Research papers had a large impact on how to best carry on the project, particularly in guiding the selection of methods for data processing and analysis. Key areas of focus include OCR for text extraction, NLP for regex patterns to capture food item names and techniques for linking extracted data to names in a database. This chapter highlights the current state of technologies and identifies the research gap this project aims to address to lay a foundation for the design choices made in developing this project.

3.2 Current Systems

Several existing systems in the market aim to help manage food tracking with some automation, but they primarily focus on barcode scanning and manual input, which has limitations in terms of item recognition accuracy and flexibility. One notable example is NoWaste, an app that helps users manage food expiry by managing their pantry. Although it offers similar functionalities, it requires manual input from users later.

3.3 Introduction to OCR

Manual data entry can be time-consuming and prone to errors, however with the development of OCR there has been a shift towards more automated systems. OCR has become a widely used tool for converting printed or handwritten text into editable digital text. Despite its widespread application, OCR faces challenges, including variations in text styles and difficulties in differentiating similar-looking characters, such as the digit "0" and the letter "o" (Patel et al., 2012). The accuracy of OCR can depend on preprocessing and segmentation techniques used to enhance the image quality and simplify the text for recognition (Patel et al., 2012).

While there are several OCR tools available (refer Appendix C), only a few are open-source and free for use. One of the most notable open-source OCR tools is Tesseract, which has become popular for recognizing text however, under various conditions, can struggle with different font sizes, orientations, and complex backgrounds (Patel et al., 2012).

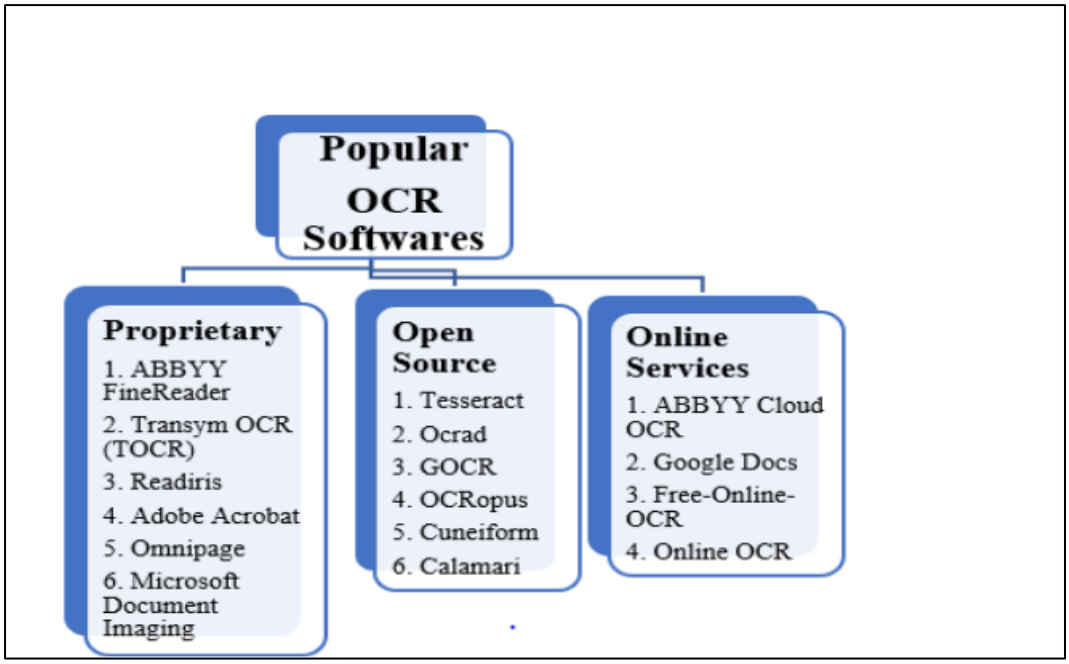


Figure 2 *Popular OCR software's*

3.3.1 Preprocessing

The effectiveness of OCR however depends on the preprocessing steps applied to the receipt image. Preprocessing techniques like noise reduction, thresholding, and image binarization play a crucial role in enhancing the accuracy of OCR by improving image clarity and distortions (Kaderabeck,2023) therefore Preprocessing methods are crucial to optimize accuracy to ensure the extracted data is reliable.

3.3.2 Regular Expressions

Receipts contain information in a recognizable pattern such as food items followed by totals. These patterns can be identified using regex, a method that matches text strings based on specific sequences such as "[0-9]" to match numbers (Kaderabek, 2023) as well as filtering information out , however the accuracy of this approach depends on the quality of the OCR output and so its effectiveness is limited to OCR errors or inconsistent formatting requiring additional post processing.

3.3.3 post-processing

Post-processing in OCR systems can significantly enhance the accuracy of the extracted data, especially after the use of REGEX , to clean text or as discussed in work by Bassil et al.(2023), using a dictionary-based post-processing method for correcting errors without the need for complex machine learning models, while lexicon-based methods can improve accuracy as demonstrated in Kazuhiro's(2023) research. Given its simplicity , dictionary-based is a more practical option for this project.

3.3.4 String Matching

There are many String matching techniques available to find similarities, in which fuzzy matching is a popular technique in use. Fuzzy matching techniques are useful for handling variations in data, especially dealing with OCR outputs which may still have spelling mistakes or misrecognized characters. The goal of fuzzy matching is to match strings that are almost the same using a threshold, which can be used to match to the food database to retrieve the expiry date. Fuzzy-token similarity is more effective than traditional exact matching techniques especially when working with imperfect or noisy data (Wang,Li and Fe, 2011). The research showed that fuzzy-token similarity, by considering both token and character-level matches significantly outperform standard token-based methods, providing better accuracy in string matching tasks by utilizing techniques like edit similarity, which accounts for insertion, deletion, or substitution between token pairs. Refer to Appendix K for additional information

3.3.5 Relevant studies

Kaerabek's(2023) study on OCR in capturing data from receipts provided valuable insights .The study explored the use of Tesseract OCR combined with **ImageMagick** to process receipts, showcasing its potential for reducing manual data entry efforts and improving the efficiency of data collection. Kaderabek (2023) emphasized the role of preprocessing techniques such as binarization, resolution adjustments, and skew correction, which significantly improved the text recognition accuracy of the OCR tool. However, despite these improvements, the study revealed challenges when it came to handling inconsistent receipt formatting and varying quality of receipts.

One of the critical observations in Kaderabek's (2023) work was that Tesseract OCR performed well in recognizing structured numerical data, such as costs and totals. However, it struggled with the recognition of food item descriptions due to their varying formats and the impact of poor image quality on the OCR accuracy. To address these limitations, the study used **Regular Expressions (REGEX)** to parse and categorize the extracted text, filtering out irrelevant data to improve the quality of the OCR output. While the study demonstrated the effectiveness of OCR for reducing manual data entry, it highlighted the need for Natural Language Processing (NLP) techniques to enhance the accuracy of item recognition, particularly for food items where inconsistencies in descriptions were common (Kaderabek, 2023).

3.4 Web Development: Flask, HTML, CSS and javascript

The receipt scanning web application will use **Flask** for backend development as the code will be implemented in python, enabling tasks like receipt uploads and OCR processing. The frontend will be built with **HTML**, **CSS**, and **JavaScript**, ensuring an interactive user experience.

3.5 Conclusion

This chapter has provided an overview of the background research and technologies that will aid the development of the system. It has highlighted the foundational knowledge that will guide the project's design and implementation discussed in later chapters.

CHAPTER 4 METHOD

4.1 Software Engineering Method

The project follows a research driven, agile-based development cycle to assess the feasibility of using AI to scan receipts and automatically update the users pantry. The project iteratively evolved through flexible sprints where it deviated at multiple points based on performance, experimentation outcomes and real-world constraints. The initial plan outlined in the PDD referring to appendix A proposed a 2- phase development model:

1) **Experimental phase** focusing on ocr and data preprocessing
2) **Report writing phase**- documenting findings, analyse results

However as the project progressed it became clear that a sprint-based approach was more effective in addressing unexpected challenges, such as:

- Extracting text from unclear receipts which were crumpled , faded, under noisy conditions and such.

4.1.2 Incremental Development Approach (Sprint-Based Methodology)

The system was developed in multiple sprints, with progress being dependent on the successful completion of the preceding stage, where experimenting with the preprocessing techniques could dramatically change results .

Sprint	Objective	Techniques Implemented	Expected Outcome
Sprint 1	Basic OCR Text Extraction	Tesseract OCR (after completing analysis of OCR tools)	Extract raw text from receipts (for baseline accuracy)
Sprint 2	Image Preprocessing + Regex filtering	Apply grayscale conversion, adaptive threshold, denoising ,gaussian blur, Edge detection. Creating regex patterns	Improved OCR accuracy on unclear receipts Extract food items while removing unnecessary text
Sprint 3	Database Matching for expiry dates	Using a text-matching algorithm fuzzy matching to link extracted food items with database	Fetching expiry dates
Sprint 4	Making Web app	Flask, HTML, CSS, JavaScript	Allows user to upload receipt and view their pantry.

Sprint 5 : technical testing to test all receipts in the working pipeline, and then testing another preprocessing technique

After Sprint 1, once a consistent and successful food item extraction was achieved from a single receipt, development moved forward with one receipt. As a proof of concept, this strategy prioritized technical feasibility before expanding to multiple receipts. The original plan did not consider making a web interface however it was built to verify feasibility through user interaction. Using multiple receipts was considered as a future extension after establishing results for one test case.

4.2 Management Tools

To manage the development phase multiple tools were used:

Tool	**Development Process**
OneDrive	Storing notes for evaluation purposes such as user testing
Jupyter notebook	Used in early-stage prototyping, particularly for preprocessing and ocr for each receipt to extract text.
GitHub	Source code

To maintain the code of the project, all were uploaded onto GitHub to allow regular tracking of changes across the sprints. Due to libraries and dependencies updating it could break the core functionality, so a txt file was maintained throughout development to document specific package versions in case of major changes or deprecations.

Backups of local database, test receipts were saved on external usb drives as well as synced to one drive.

4.2.1 System Preparation , tools and software's used

The project was developed using the following tools and software's listed :

Tool/ Library	**Purpose**
Flask	Backend web framework
Python	Main programming language for OCR,NLP, database interaction and web development
Visual Studio Code	IDE to write, test, manage python, html, CSS and JavaScript code
SQLite3	Local database for food items with expiry dates chosen as my database as it was simple and serverless to setup for a proof-of-concept system
pytesseract OCR	Text extraction from images
OpenCV / NumPy	Image preprocessing before OCR

Pillow (Pil)	Image handling
Rapid Fuzz	Fuzzy matching to map the food item name to those in the food database
Regex (re)	Parsing text
HTML/CSS/JavaScript	Front-end interface with styling and modal editing and interactivity
Figma	UI mock-ups for simple interface

A virtual environment was created to manage dependencies, to not affect the system's packages. All dependencies were installed using pip inside the virtual environment defined in requirements.txt.

The primary development environment was VScode as it has strong python support and a good file organization

4.3 Dataset collection

To support the feasible study, a small dataset of real-world shopping receipts were collected, and used on a controlled approach where one receipt with clear OCR results was used for the iterative development and experimentation across early sprints. Additional receipts with different qualities were held back for technical testing in the final sprint. Refer to Appendix C for dataset description.

4.4 Project Requirements and analysis

Understanding the system requirements was vital to establish the functionality of the feasible test as it explores the technical potential of automated food tracking through receipts by working with a controlled receipt then expanding to other receipt formats.

4.4.1 Functional and Non-Functional Requirements

iteration	Requirement ID	Requirement Description
Sprint 1 – Baseline OCR extraction		
	1.1	The system must extract raw text from a receipt using Tesseract OCR for the output to be analysed
	1.2	The system is tested on receipts with varying backgrounds to see which condition gives good text extraction
	1.3	Evaluate the accuracy of the OCR in extracting the food item names from the receipts using CER and WER
Sprint 2 – image Preprocessing Techniques and Regex Filtering		
	2.1	Explores whether grayscale, thresholding and blurring techniques improve OCR accuracy on receipts.
	2.2	System must re-run OCR after each preprocessing technique to assess improvements on text extraction
	2.3	Parameter tuning (kernal sizes, threshold values) should be explored to assess impact on OCR output quality
	2.4	The system should be able to process OCR output using regular expressions to filter out non-food data (e.g totals, prices, measurements)
Sprint 3-Database Matching		
	3.1	The system should be able to match the extracted food items against a local food database using fuzzy string matching
	3.2	The system is tested to evaluate how different fuzzy thresholds affect accuracy and false positives when matching food item to database to fetch expiry date
Sprint 4- Web App Integration		
	4.1	The prototype should integrate the tested OCR preprocessing, Regex filtering

		and matching database layers into a web application
	4.2	The web app should allow a user to upload a receipt and viewing extracted food items with its expiry dates automatically updated
	4.3	Interface should allow users to view and edit entries of their pantry
	4.4	Users should be able to manually add food item which then automatically gets the expiry date and updates user pantry if food item was not detected by OCR as fallback method
Sprint 5- Technical Testing		
	6.1	The system is evaluated using more receipt images with different qualities to assess robustness
	6.2	The system is tested to see if edge detection preprocessing improves OCR results

4.4.1.1 Use Case Diagram

The use case diagram was created to help visualise how a user may interact with the system in a future deployment scenario, although the system didn't involve real-time user uploads during the feasibility study, modelling how the user actions could be helped outline the system's potential scope, so this diagram allowed clearer planning of system expectations.

4.4.1.2 ERD Diagram

The ERD diagram helped form the database structure as from the analysis of the data, it was clear that the system needed to store 2 main types of entities: food items in a database with their predicted expiry dates and pantry items that were extracted from receipts so SQlite was chosen as the data model was simple and structures with no nested data. The relationships between data were minimal making SQLite the ideal database for feasibility testing .

4.4.1.3 State Diagram

The state diagram modelled a few core states the system goes through where an example being how a food item progresses through the pipeline, how the system handles both successful matches and unmatched item , how the expiry notification would be triggered, and how the manual fallback method could occur.

4.4.1.4 User Interface Design

The UI was designed using Figma to visualize the receipt scanning system before the implementation. Wireframes were created to show the core features of the system such as the

dashboard of the pantry, receipt upload and user inventory. These wireframes helped establish a minimal design prioritising clarity and usability to reflect the AI pipeline. The designs were later created using HTML, CSS and JavaScript for the Flask- web app.

4.5 Testing and Evaluation

Testing played a crucial role throughout the project's lifecycle, ensuring each output aligned with the project's objectives through meeting the sub objectives as well to validate feasibility at each point. Every sprint added a new functional layer to the pipeline, beginning with raw OCR extraction for baseline results leading to database matching, therefore, to manage this complexity each layer was isolated and tested independently before integrating it to a web app.

4.5.1 Sprint 1

The challenges outlined by Patel et al. (2012) in section 3.3, such as OCR's struggle with varied text styles and poor image quality set a clear sprint 1 phase, establishing a baseline OCR extraction test . These initial results were useful in determining whether OCR could handle the expected variety of receipt formats and qualities, as highlighted by Kaderabek (2023). Based on calculating the OCR accuracy (Martinez, 2025) on food items refer to Appendix G , the controlled receipt was chosen amongst the dataset to proceed to the next sprint. The accuracy rate of the OCR has major impacts for both usability and efficiency of the images as a higher accuracy means less human effort is needed in updating data therefore less manual input required, however a poor OCR accuracy would lead to inaccuracies like misspelt names and values leading to the automated process not being feasible as it would be unable to match to the food database to retrieve the expiry dates to update the user's pantry. With the results it provided insights into adjustments for the next stages in preprocessing and text filtering.

4.5.2 Sprint 2

Once a receipt demonstrated some good food item recognition using basic OCR, it became the controlled experimentation with the next sprint, image preprocessing and regex pattern. Various pre-processing techniques were applied, such as grayscale conversion, adaptive thresholding, gaussian blurring following Kaderabek's (2023) recommendations to improve OCR text clarity as discussed in section 3.3.1. After each transformation, OCR was re-applied to observe any changes in extraction quality. Parameters like kernel sizes and threshold values were also tuned iteratively to see its role in text extraction and how it affected the results. Once the controlled receipt confirmed good results, the other receipts were kept for technical testing for later to ensure feasibility of the system was first achieved before assessing robustness. After this the focus shifted to text filtering using regular expressions (refer to appendix J for more information) As described by Kaderabek (2023) in section 3.3.2, regex was essential for filtering out irrelevant data. Test outputs from the processed OCR text was analysed to determine how effectively non-food data was removed (such as Total, Store Name) whilst keeping food items. However, the receipt format was different for some food items such as some having * or / before the food name so these findings guided regex refinement to accurately capture the text without symbols and unnecessary data such as measurements (150g). The post-processing stage was guided by the research in 3.3.1 where methods like dictionary-based correction was explored to fix abbreviations found in receipts.

4.5.3 Sprint 3

The third sprint, database matching was done using fuzzy matching techniques following the research on fuzzy-token similarity (Wang, Li, and Fe 2011) in section 3.3.4 and Appendix L. At

this point, the extracted food item names were tested for successful linkage against the food database. The system needed minor variations in spelling for example (“FTG pineapple” vs “ pineapple”) therefore the threshold used aimed to maximise correct matches whilst minimising incorrect matches (refer to Appendix K for more information).

4.5.4 Sprint 4

The fourth sprint involved integrating the validated components into a functional web app using the flask framework (refer to Appendix G) Using all the working functional layers into a full working pipeline to evaluate a potential system usability in a real-world scenario, where users can upload receipt images, view their food items that are automatically stored in a pantry with it’s expiry dates as well as getting expiring notifications in the app so users know what to use up first.

4.5.5 Sprint 5

After the successful implementation of the web app, Sprint 5 focused on technical testing using a wider variety of receipts, as discussed in Section 3.5 Relevant Studies, where Kaderabek (2023) emphasized challenges with inconsistent receipt formatting and varying receipt qualities , where earlier stages focused on one controlled receipt to validate feasibility, this sprint revisited previously stored receipts to evaluate different receipts to see how well the pipeline handled variability which included receipts with darker backgrounds, faded ink , noise and preprocessing techniques were reapplied and adjusted as necessary to optimize OCR accuracy. Edge detection methods were applied to assess whether background played a role in extracting text to improve character visibility. While this phase was primarily exploratory, it helped define the range of challenges that could be imposed to the system and if food item names could be retrieved from those conditions. This phase provided a broader understanding of the system’s robustness as before this it was important to establish a working proof of concept on a receipt that could successfully extract food items, filtering irrelevant data as well as matching against database. This selective focus allowed each layer - OCR, image preprocessing, Regex filtering and fuzzy matching to be developed and tested. Once feasibility was clearly demonstrated , technical testing was done to assess robustness.

All these sprints collectively contributed to refining both the technical capability and usability of the system , where from the first sprint to the final addressing each key layer of the receipt scanning system. By validating this against varied conditions and evaluating output at each stage, this helped the project explore the feasibility of receipt scanning as a solution for automated food tracking where it supports users in managing pantry items more efficiently, providing a way to reduce waste, save time and integrate into their routine.

4.5.1 Legal, Social, Ethical and Professional Considerations

Although the primary focus of this project was to assess the technical feasibility of AI-driven receipt scanning, legal, social, ethical, and professional considerations were also carefully addressed throughout the system’s design and development.

From a legal perspective, the project adhered to data protection principles by ensuring that no personal or sensitive data from receipt images was retained or stored. During the OCR and preprocessing stages, techniques such as regular expression filtering were applied to exclude

non-food elements, including payment details or customer information, thereby aligning with GDPR guidelines for handling potentially sensitive data.

From an ethical and social standpoint, the possibility of OCR inaccuracies was recognised as a potential risk. Incorrectly extracted food items could result in misleading expiry information, which may influence users' food decisions. To mitigate this, manual override functions was implemented, allowing users to review, edit, and correct any entries as needed.

In accordance with professional standards, the system was developed and refined through iterative sprint cycles. Each development stage was carefully evaluated to ensure that the solution posed no harm to users and maintained transparency and reliability in its functionality.

4.6 Interface Implementation

4.6.1 Flask Framework

Flask was chosen as the back end for this project due to its lightweight nature and ease of integration with python-based code. Whilst initially not familiar with flask, it was easy to implement due to its ability to build and integrate the pipeline quickly making it ideal for this project.

4.6.2 Front-End Interface

The user interface was built using HTML ,jinja2 to render dynamic pages from flask and styled with CSS. With a solid familiarity with these languages, the author was able to develop a simple, user-friendly interface to simulate user experience, ensuring that the core features worked as intended.

4.6.3 Database integration

Two database options were considered to manage data. SQLite was chosen over SQL due to being server-less and easy to set up (Wisniowski, 2022) which is enough for the proof-of-concept nature of this project handling small data.

SQLite	SQL
A self-contained, **serverless** database system that stores data in a simple file.	A traditional relational database system used in larger applications **requiring a server** to manage queries.
Simple to set up, making it lightweight, easy to integrate	Requires setup of database server – can be **complex , time-consuming**
All data stored in a **single file**, can be run with scripts in file	Server manages and run queries
Well suited for **small sized applications** like proof of concept where, only handling small data	Suitable for **large-scale applications** with a lot of data
Requires **regular** database maintenance	**Minimal** maintenance needed as file-based approach

Figure 4.5 SQL vs SQLite

4.7 Reused material

Throughout the projects development several resources were reused and adapted to support the implementation of the system where it included open-source libraries, online tutorials, and pre-existing code that aligned with the goals of the feasibility system. Refer to Appendix B for the documentation.

4.8 user testing

To test the usability and value of this project, user testing was conducted to understand the users current tracking behaviours, attitudes towards food waste and their experience interacting with the system where each session included a short interview with then testing the system.

4.9 Project Deviations

Several changes were made to the original plan due to various challenges and time constraints. These deviations were essential to maintain the focus of the project whilst ensuring its feasibility. Refer to Appendix H for the overview of the key deviations and reasons

CHAPTER 5 RESULTS

This chapter will present the outputs produced in more detail and further explore the mentioned tasks in the methods chapter if needed.

5.1 Dataset Collection

The receipts used in this project was collected by capturing real-world receipt images during shopping trips which aligns with the approach discussed in 3.2.1 and 3.5, which emphasized the variability in receipt formats, ink quality and background conditions by testing real receipts, the project was able to assess how well the OCR system could handle the complexities of real-world data. This served as the primary source of data for testing instead of a predefined or controlled dataset to allow the project to simulate real-time challenges a user may face whilst scanning receipts with mobile app.

5.2 Recording of Results

This section will describe the results that came from sprints mentioned in chapter 4 explaining the aim of each sprint with its results as well as from chapter 4.5 during the testing and evaluation.

5.3 Sprint 1 results

Aim:	Baseline OCR extraction
Methods:	Tesseract OCR ran on receipt, calculating **CER :** evaluates the accuracy of the receipt as a whole and food item names extracted from the receipts based on recognizing individual characters for accurate food item identification. **WER:** evaluates accuracy of extracting information from the receipt as well as correct food item names from the receipts based on recognizing whole words in the food item names.

Receipt one

Receipt Output

1 OEE WE'D OH Os COHEN js Aime Fate «a
fe hetw rs 1 OW OH eb eR FoF
a ie ae fae
WO) Ra tes wes 1 Uy 0
|b) a Tarai o
ast nb th ty +4
4
: . – - es PLO E ML HONE bm &
Mae! Rae! Case Sa Say a Naif a OS « . Z Py ee >] cod J Trier re
Raph Mae 6 Ga Mar > Tua $e 0 Ni a) ar tiles Ay - ve os tom hbk ors Le ;
Fe (G4 Rae be Gan! Ree (ca aw Nae a VLG ph eae Yeah ey ee Nga) liad mt ot i Be pi
Si Raa 0 Gar VG ae a na ay . var sa) Vp ae! saad Jab ated á ad Naa) WB 1 ee AF
be eh G6 8 I eh
Die Path 4 Ae
1 et aso
ak Ry 8 Gr eee
Wee Sir Cae te) Gaeta ae on ap
See EL OE Ree ey Nae Ne sie
vas ya tao ao hee ea nae ae Cad
EN EC GaN Ob Gay a OS Cae Ge a ae
LON-Hoxton Street
Enter « VAT NO, GB350396892
survey; |id].co,uk/haveyoursay
YOU Can win £100 of Lid] Vouchers.
MUltiv Tropic
Cheese € Singles
Neapolitan Ice Cream
Foz OvenChip Homest
W Sliced White Rolls
Carrier Bag
TOTAL
CARD
Fa PSN es ee rac
LT aa we
NGS ha Bes ae
ER ON YS NS a
SRS ge Fp ieee = teh By
AD iar or
OP aS 5 Be = ai as 5 ap --
EIDE GS 1 OS LTS Se fi
PEE ECE § AS tthe Te Ps
pee ee ee eee
wT > re OW ee WC sp
GT cM oo oO
ON ORE OM & panty cae Siege Tae ae
ne a ee ne
CUSTOMER COPY* - PLEASE RETAIN RECEIPT
Date: 24/02/25 Time: 10:59346
MID: #4*95014 TID: #1404
TRNS NO: UK201177019134055251
DEBIT MASTERCARD PAAR EEEREDS 94
4000000004 1010
Contact less SALE
Amount £8.11
Verification Not Required
APPROVED AUTH CODE 192304
PLEASE DEBIT ACCOUNT WITH TOTAL SHOWN
VAT RATE DALES £ VAT £
A Oh 4,27 0.00
B 20 3,64 0.64
MT
VOOM LOL ELON ELE
IM tO (eee ee

	OCR Accuracy rate on entire receipt
CER	1.50
WER	2.94

Actual food items	Extracted food items
Multiv Tropic Drink	MUltiv Tropic
Cheese Singles	Cheese € Singles
Neapolitan Ice Cream	Neapolitan Ice Cream
Froz OvenChip Homest	Foz OvenChip Homest
W Sliced White Rolls	W Sliced White Rolls

	OCR Accuracy rate on food items
CER	0.41
WER	0.76

The accuracy of this was evaluated by comparing the WER and CER for both the entire receipt and food item names alone where they revealed significant challenges due to factors such as the complexity of text layout, background (visual noise, patterns),contrast between them.

For the entire receipt, the OCR system had a high error rate with WER 2.94 and CER 1.50 which indicates that the system struggled with accurately recognizing diverse information contained in the receipt, including store details transaction codes, amounts, VAT however when the system just focused on the food items, the performance significantly improved with a WER of 0.76 and CER of 0.41. The lower error rates show that the food items text was easier to process despite minor misreading's of certain characters it demonstrated better accuracy with names which are more consistent in structure. The food items were relatively easier for the OCR system to handle, despite some minor misreading's The study by Kaderabek (2023) suggested that these variations could lead to reduced OCR accuracy, which is evident in the baseline sprint results, where high WER and CER were observed due to challenges such as glare, reflections, and noisy backgrounds. These challenges suggest the need for enhanced preprocessing techniques such as contrast enhancement, background removal to improve OCR accuracy. Whilst this showed promising results for food item extraction, further refinements are necessary where in real-world environments background noise and receipt layouts vary.

Receipt two

Receipt Output

Sainsbury's
Good food for all of us
ISLINGTON ST JOHN STREET LOCA
03830 013 7177
Sairisbury's Supermarkets Ltd
33 Ho)}born London ECIN 2HT
www. sainsburys. co.uk
Vat Number ; 660 4948 36
*EVIAN S/C 750ML £1,806
*GRENADE WHITE OREO £2.00
FTG PINEAPPLE i606 1,40
*xCDBY SLT CRML FNGER £1.50
4 BALANCE DUE £6.65
Visa DEBIT £6.65
contactiess »)))
{Icc] FOI IK KR 7B
AID: AQOOVONGO3 1010
PAN SEQUENCE: 00
MERCHANT : *xxKGG04
AUTH CODE: 003187
TID: «xxx 1110
Cardholder Device Verified
CHANGE £0.00
REAR EKA RIK KKK AAR KR KK KAKA KKK KKK KK MAK KKK i
MY NECTAR SUMMARY
[OC] x*Kx xxKK xxxx *x*5032
POINTS EARNED ON £6.65
PREVIOUS POINTS BALANCE 299
POINTS EARNED 6
NEW POINTS BALANCE 305
YOUR POINTS ARE WORTH £1.92
KRKKKKEKKKARKKKEKKRKKEREKKRAKRKEKKRKKEKKKKEEKAK
Check the Nectar app or nectar.com to see
any bonus points you might have collected.

xxexx For a chance to win **«x-»
100,000 Nectar points
please tell us how we did at
lettuce~know. com
ERKKAKKKERRKKKKRARAREARKAKKKKK

PLEASE KEEP FOR YOUR RECORDS
PUBLISHED TERMS AND CONDITIGNS APPLY

wh

#3852 = 17:18:59 O5MAR2025
$4280 R62 ,

you for your visit.

OCR Accuracy rate on entire receipt	
CER	0.27
WER	0.6

Actual food items	Extracted food items
*EVIAN S/C 750ML	*EVIAN S/C 750ML
*GRENADE WHITE OREO	*GRENADE WHITE OREO
FTG PINEAPPLE 160G	FTG PINEAPPLE i606
*CDBY SLT CRML FINGER	*xCDBY SLT CRML FNGER

OCR Accuracy rate on food items	
CER	0.087
WER	0.54

The OCR accuracy for Receipt Two and food items was significantly better than Receipt One, where the condition of the receipt had an impact on the OCR system's performance. Receipt Two had high contrast between the black text and the white background, with a clearer layout compared to the first receipt. This contributed to a better OCR performance overall.

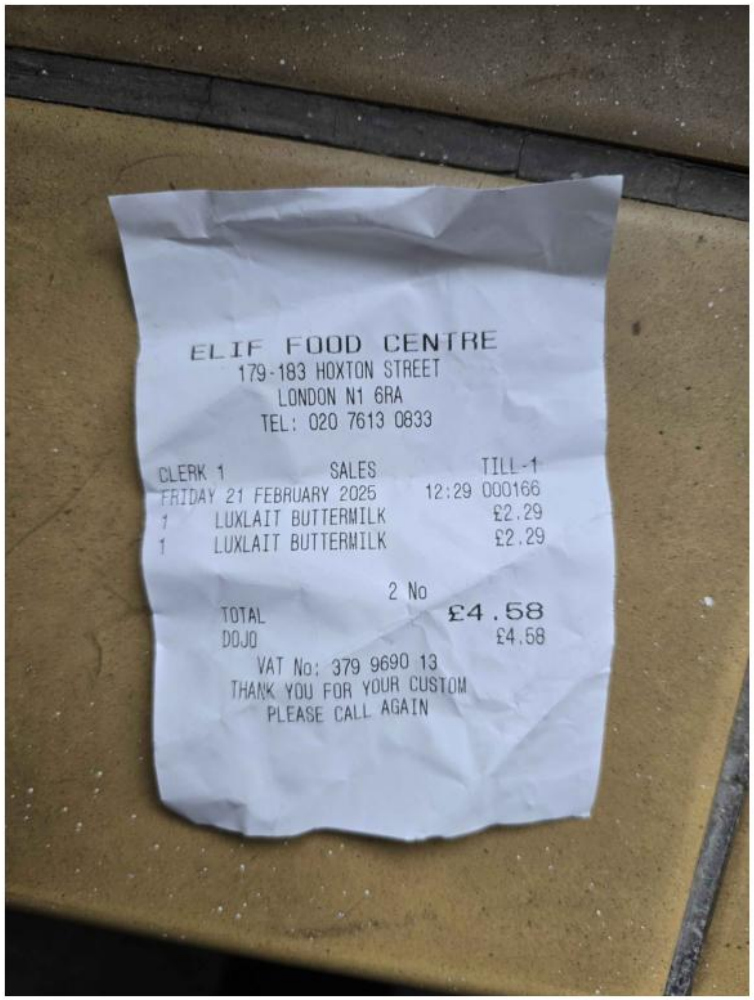
For the entire receipt, the CER of 0.27 and WER of 0.6 indicate that while there were some misinterpretations, the error rates were still relatively low for a full receipt that contains varied data types, such as store details, transaction information, and product prices. A CER of 0.27 suggests that the OCR system accurately recognized most characters, with a small percentage of errors. Achieving a CER below 1.0 is considered a strong result, showing that the system was able to handle the diverse text on the receipt effectively. The WER of 0.6 means that 60% of the words were correctly recognized, with the remaining words misinterpreted. These results suggest that while the OCR system may need some improvements in word-level recognition, it was still largely able to maintain the structure and context of the receipt, especially for less complex sections.

For food item extraction, the OCR accuracy was impressive, with a CER of 0.087 and WER of 0.54. The low CER of 0.087 indicates that almost all the characters were correctly recognized in the food item names, which is an excellent result, especially since OCR systems often struggle with similar-looking characters (such as "6" vs "G"). However, the WER of 0.54 suggests that a small percentage of words were still misinterpreted. The OCR system performed exceptionally well at recognizing food item names, with only minor character-level misinterpretations, such as "FTG PINEAPPLE 160G" being misread as "FTG PINEAPPLE i606".

Based on these results, while the OCR system performed well overall, improvements in character-level accuracy, particularly for numbers and alphanumeric confusion, could enhance

the food item extraction process. Techniques such as preprocessing to enhance contrast or focus on improving character recognition (especially for digits like 6 vs G) could further improve the overall accuracy of the system leading to a better WER.

Receipt three



Receipt Output

e_IF FOOD CENTHE

179-183 HOXTON STREET

LONDON Ni 6RA

TEL: 020 7613 0833

CLERK 1 SALES Tce

FRIDAY 21 FEBRUARY 2020 12:29 0001660

LUXLAIT BUTTERMILK £2.29

LUXLAIT BUTTERMILK £2.29

2 No

TOTAL S468

OUD £4.58

VAT No: 379 9690 13

THANK YOU FOR YOUR CUSTOM

PLEASE CALL AGAIN

	OCR Accuracy rate on entire receipt
CER	0.17
WER	0.53

Actual food items	Extracted food items
LUXLAIT BUTTERMILK	LUXLAIT BUTTERMILK
LUXLAIT BUTTERMILK	LUXLAIT BUTTERMILK

	OCR Accuracy rate on food items
CER	0.0
WER	0.0

Despite the receipt being slightly crumpled and having a darker background, the OCR accuracy was notably high. The CER for the entire receipt was 0.17, indicating that 83% of the characters were correctly recognized. This is an excellent result, suggesting that the OCR system was effective in handling the text recognition, even with the receipt's condition. The simpler format of the receipt, with clear separations between food items, prices, and totals, made it easier for the system to distinguish between different types of text. Additionally, the absence of barcodes, compared to other receipts, contributed to improved recognition accuracy. The WER of 0.53, indicating that only 47% of the words were misinterpreted, suggests that the OCR system could correctly extract most of the text but had some difficulty with specific words, likely due to the creases and slight distortions in the printed text.

For the food items, the OCR system achieved perfect accuracy with a CER and WER of 0.0, meaning 100% of the characters and words were extracted correctly without any errors. This demonstrates that the system was highly effective in recognizing food item names, as the text was clearly printed. To further improve accuracy, preprocessing techniques such as noise removal could help mitigate the challenges posed by any physical imperfections in the receipt. Overall, these results highlight that even with some minor distortions, the system can still achieve strong performance with well-structured text like food item names with simple receipt structures.

Receipt four

Receipt Output

TESCO

4 AD 4 AD

South Plaza Isle Of Dogs Express
Any questions please visit
www. tesco.com/store-|ocator

VAT Number: GB 220 4302 31

Warburtons Dan ish Sliced smal |

White Bread 400g

TOTAL:
Card

Cl ubcardspoints- earned:

Clubcard points balance:

Visa Vep]

ALL: AQO00OOgA= UL

Number : ARK RKI RR RTOS

Pan sequence no: m0

Authorisation code: OZ38g

Merchant: xxK^ASOS

OC

6AM3-1JEZ-3044-268x
208

23/02/2025 14:48 Store: 6289 Checkout

OCR Accuracy rate on entire receipt	
CER	0.24
WER	0.60

Actual food items	Extracted food items
Warburtons Danish Sliced Small White Bread 400g	Warburtons Dan ish Sliced smal | White Bread 400g

OCR Accuracy rate on food items	
CER	0.08
WER	0.57

The OCR performance for the Tesco receipt showed promising results with a CER of 0.24 and WER of 0.60 for the entire receipt. Despite some misreading's likely caused by the receipt's condition, such as fading or creases in the text. The high WER indicates that there were some word-level inaccuracies, possibly due to non-standard fonts or misalignment of text in the creased areas. For the food items, the OCR system performed much better, with a CER of 0.08 and WER of 0.57. Despite a few misinterpretations like "Warburtons Danish" being extracted as "Warburtons Dan ish," the system was able to recognize the food items accurately with low error rates. These results highlight that while the OCR system can efficiently handle structured data like food items, further preprocessing techniques, such as enhancing contrast or noise reduction, could help improve character-level accuracy and ensure even better food item extraction.

Receipt five

Receipt Output

TESCO

ae eS a a a

lington

Ons please VISIT

ANY? quest
WWW tesc o.com /s tol e-] oca T Or

VAT Number: GB 220.4302.31

REPRINTED REGEIPT
_PRINTED® 22/02/2025 12: 39

ee co Classic ie 4 Coffee
2006
REPRINTED RECEIPT
PRINTED 22/02/21 025.1239

TOTAL.

Clubeard noints earned:

Visa Debit

AID: A0000000031010
Number : KKKKKKKKKKKK 1394
Pan sequence no: 0
Sithorisation code: 024095
Merchant: x*xx 4809

22/02/2025 12:37 Store:-2747

	OCR Accuracy rate on entire receipt
CER	0.33
WER	0.91

Actual food items	Extracted food items
Warburtons Danish Sliced Small White Bread 400g	Warburtons Dan ish Sliced smal | White Bread 400g

	OCR Accuracy rate on food items
CER	0.32
WER	1.0

In conclusion, the OCR system showed reasonable character-level accuracy for both the entire receipt and food items but faced challenges with word-level recognition. The high WER for the entire receipt and food items suggests that the receipt condition (creasing, fading) impacted the system's ability to extract accurate text. To improve accuracy, preprocessing techniques like contrast enhancement, noise reduction could help enhance the overall performance, particularly for word extraction and food item recognition

Receipt Two was selected for Sprint 2 due to its high OCR accuracy, better receipt condition, and ability to handle structured data effectively. The low error rates provide a strong base for further improvement, particularly in character-level recognition.

5.4 Sprint 2 results

Aim:	Sprint 2 – image Preprocessing Techniques and regex pattern
Methods:	Applying : - Grayscale Conversion - Gaussian Blur - Adaptive Thresholding To focus on extracting the food items properly and price for regex patterns

Before Starting the preprocessing stage, it was noted that food items on the receipts are often aligned with their prices refer to figure 5.1 . This presented an opportunity to enhance the text extraction process by incorporating price matching within the regular expression (regex) patterns. Given that each food item is typically followed by its cost, this alignment made it easier to capture both pieces of information using a single regex pattern.

Figure 5.1 *Receipt image food item with price*

For Preprocessing the first step started with grayscale conversion, as discussed in chapter 3, preprocessing is essential to enhance the OCR's recognition accuracy. The research (Kaderabek, 2023) highlighted the importance of preprocessing methods like noise reduction and image binarization to ensure clarity for text recognition. Grayscale conversion was chosen as the initial step to simplify the image and reduce the complexity of the receipt, allowing the OCR tool to focus solely on the text, unaffected by the colours.

Grayscale Conversion

Original Receipt

Grayscale

Initial Ocr Accuracy on Food Item with Price

CER	0.099
WER	0.29

Items to extract	Extracted Items
*EVIAN S/C 750ML £1.80	*EVIAN S/C 750ML £1,806
*GRENADE WHITE OREO £2.00	*GRENADE WHITE OREO £2.00
FTG PINEAPPLE 160G £1.35	FTG PINEAPPLE i606 1,40
*CDBY SLT CRML FINGER £1.50	*xCDBY SLT CRML FNGER £1.50

Grayscale Ocr Accuracy on Food Item with Price

CER	0.079
WER	0.36

Items to extract	Extracted Items
*EVIAN S/C 750ML £1.80	*EVIAN S/O 750ML £1.80
*GRENADE WHITE OREO £2.00	AGRENADE WHITE OREO £2.00
FTG PINEAPPLE 160G £1.35	FTG PINEAPPLE i160 £1,399
*CDBY SLT CRML FINGER £1.50	*CDBY SLT CRML FNGER £1.50

Following the grayscale conversion the accuracy for the item extraction slightly improved but there were still problems with extracting the prices and food items perfectly. Despite this the Preprocessing step helped reduce the complexity of the receipt image by removing colours which made it easier for the OCR engine to focus on extracting relevant information however challenges remain particularly with misread characters and price misinterpretation. The results however align with the findings from chapter 3 where research emphasised that image quality such as colour affects OCR performance and so further enhancements in terms of background noise reduction is required.

After these blurring techniques were tested as discussed in the context chapter, preprocessing is essential for improving clarity and accuracy of OCR outputs especially when dealing with noisy images that may have varying levels of contrast and resolution, therefore, to improve the results the author experimented with three blurring techniques : Gaussian Blur, Median Blur, and Bilateral Filtering. Each of these techniques aim to reduce noise whilst preserving important features like edges. Furthermore, the kernel size was also tested to see its role in how it improved the OCR result.

Gaussian Blur

Gaussian Blur with Kernel (1,1)

Gaussian Blur with Kernel (3,3)

Gaussian Blur with Kernel (5,5)

Gaussian Blur with Kernel (7,5)

Gaussian Blur OCR Accuracy on food item with price

Kernel size	1,1	3,3	5,5	7,5
WER	0.357	0.285	0.21	0.214
CER	0.079	0.0495	0.0495	0.0297

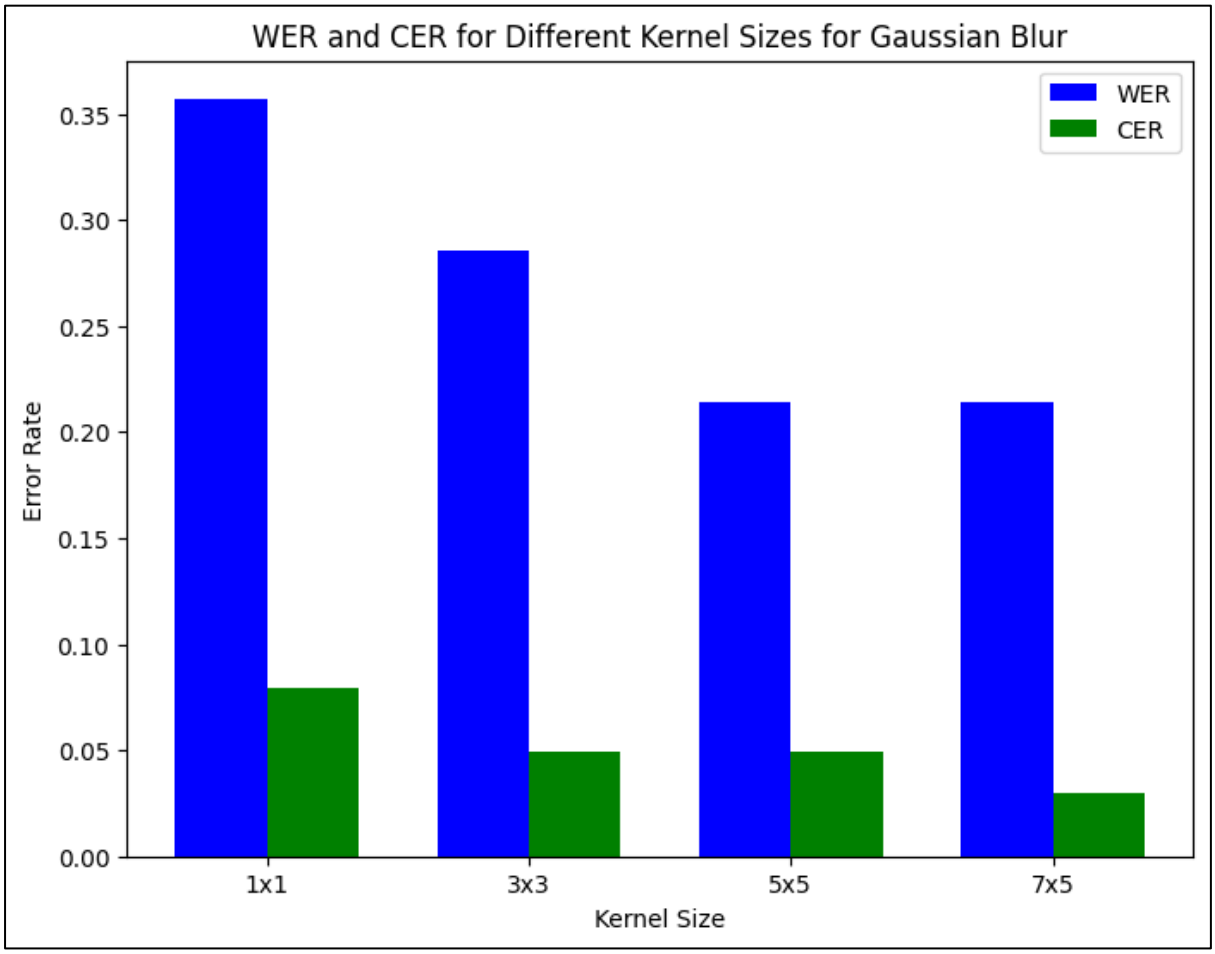


Figure 5.2 Gaussian (matrix) blur with different kernel sizes table

Gaussian Blur was applied with different kernel sizes to evaluate its ability to smooth the image while maintaining the detail for accurate character recognition. The larger kernels resulted in greater noise reduction but sometimes blurred the edges of text leading to lower accuracy but produced the best OCR accuracy results as reflected by the WER and CER of kernel size 7, 5. From the table and figure 5.2, we can see that as the WER decreases as the kernel size increases from 1,1 to 5,5 but then slightly increases at 7,5 which suggests that while increasing the size at a certain point, it may blur out too much detail affecting the OCR's ability to distinguish the words. The CER improves with increasing the Kernel size, where the highest gave the most accurate character recognition which means the smaller kernels may not smooth the image enough to reduce noise leading to less accurate character recognition. This finding aligns with the insights from the context chapter where optimizing preprocessing techniques such as blurring is essential for improving OCR.

Median Blur

Median Blur 1

Median Blur 3

Median Blur 5

Median Blur OCR Accuracy on food item with price

Kernel size	1	3	5
WER	0.357	0.214	0.143
CER	0.079	0.069	0.05

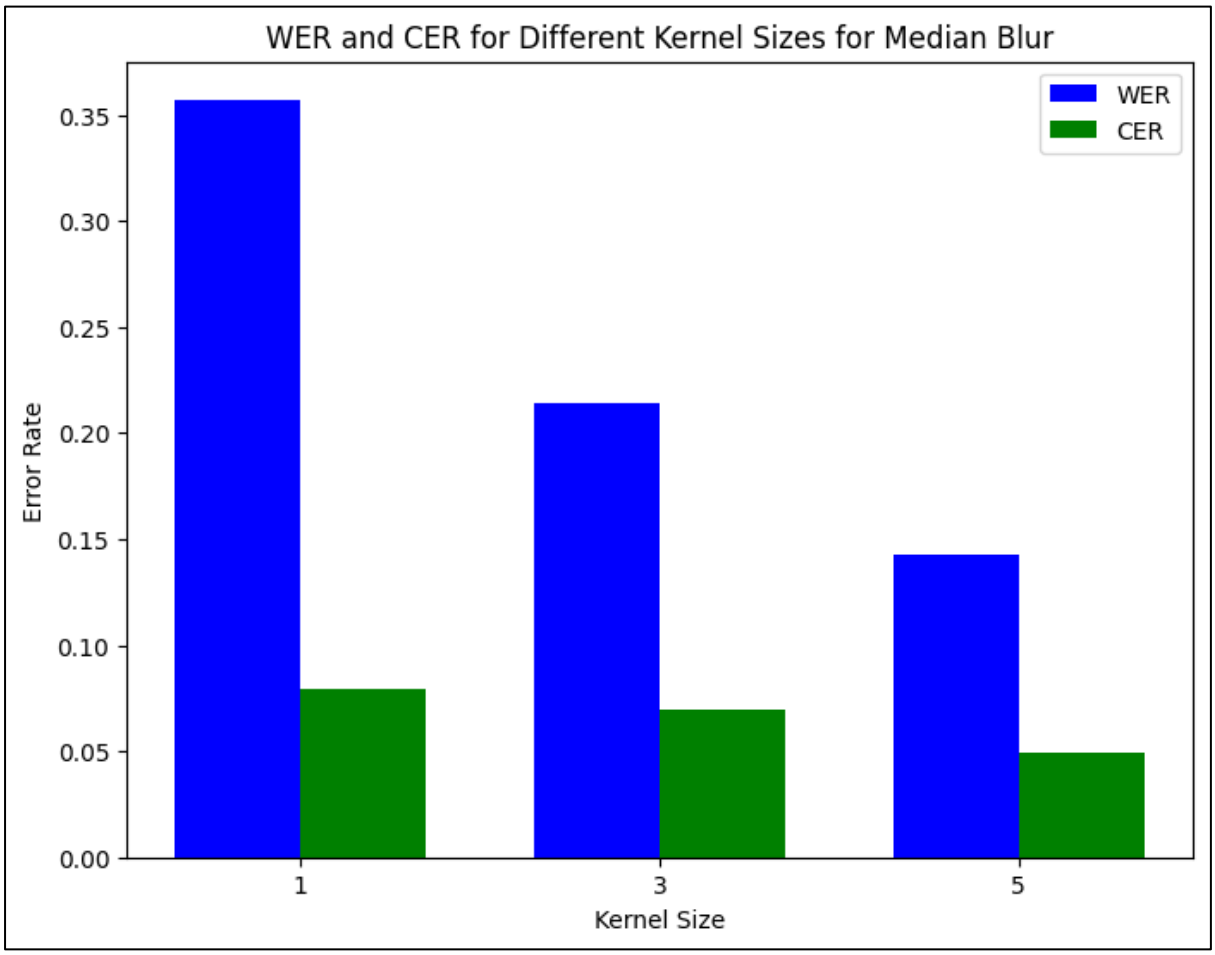


Figure 5.3 Median blur with different kernel sizes table

Following the table and figure 5.3, the larger kernel sizes improve the performance of OCR by reducing noise as it the image is more smoothed which makes the text clearer. The smallest kernel has the worst performance in both WER and CER which means there is some noise in the image that is affecting the accuracy of the OCR system so applying a blur with 5 or 3 helps remove this leading to better results. Kernel size 5 is the most effective for improving OCR accuracy as it has the lowest WER and CER. These results also highlight the importance of choosing an appropriate kernel size to enhance OCR accuracy.

Bilateral Filter

Bilateral Filter 9, 75, 75

Bilateral Filter 9, 75, 35

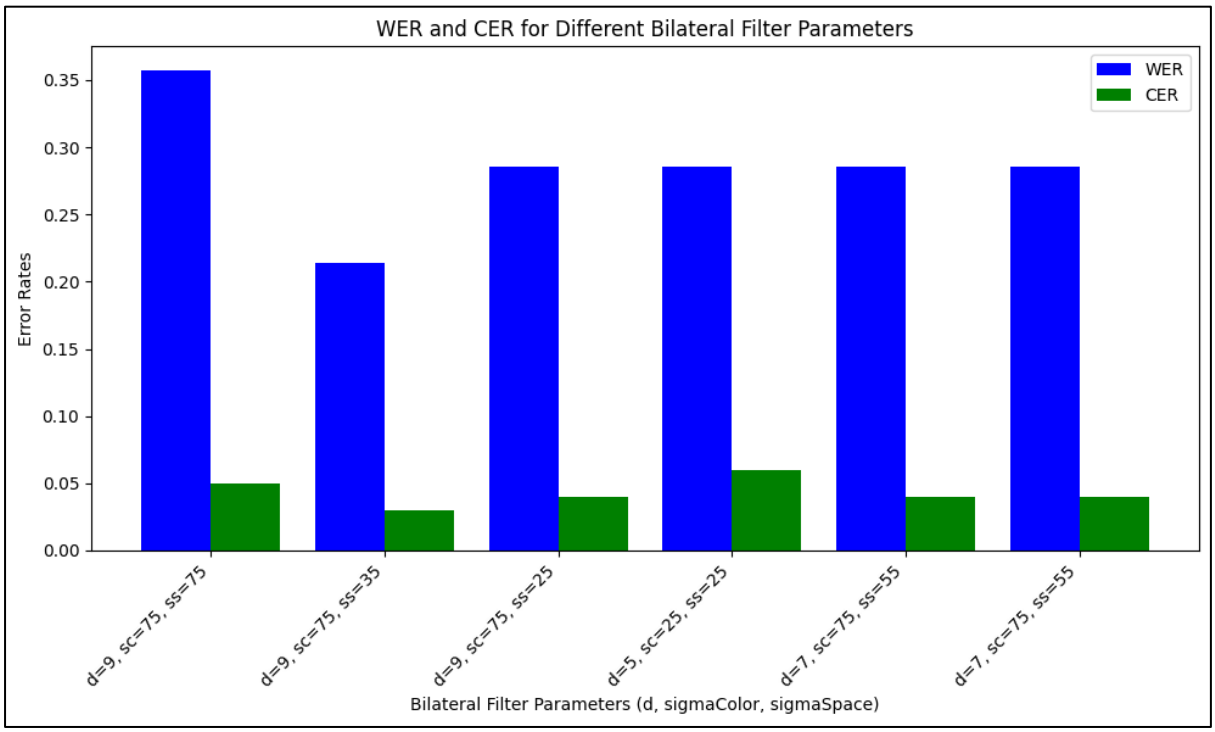
Bilateral Filter Applied 9, 75, 25

Bilateral Filter 7, 75, 55

Bilateral Filter 5, 25, 25

Bilateral Filter OCR Accuracy on food item with price

Parameters	9, 75, 75	9, 75, 35	9, 75, 25	7, 75, 55	5, 25, 25
WER	0.357	0.214	0.2857	0.2857	0.2867
CER	0.0495	0.069	0.0396	0.0396	0.0594



The bilateral filter is a edge preserving smoothing technique that can reduce image noise whilst preserving boundaries and involves three parameters:

Parameters	**What it's for**
d (diameter)	Diameter of each pixel neighbourhood
sigmaColor	Filters sensitivity to colour differences
aigmaSpace	Filters sensitivity to spatial distances

The WER being the highest with parameters 9, 75 , 75 indicates that the filter caused significant word mismatches suggesting that a large neighbourhood and high sigmacolor led to over-smoothing which blurred the boundaries between words however when it was reduced to 9, 75, 35 and 9, 75, 25 it was more effective in preserving the words whilst reducing noise. This also showed an improvement in CER meaning it also helped enhance character level recognition accuracy without reducing overall smoothing . The CER values however remained quite consistent across the other parameters showing that small parameters and adjusted sigma values provide an effective balance between reducing noise and maintain the edges for OCR. These tests therefore explore different parameters to see the impact on the ocr text extraction building upon the discussion of image preprocessing discussed in the Context chapter.

After the blurring techniques were evaluated, thresholding methods were explored as part of the preprocessing pipeline. As outlined in the context chapter, thresholding plays a crucial role in converting grayscale images into binary images to simplify the data for OCR processing. The author experimented with three types of thresholding : Simple Thresholding, Adaptive Thresholding, and Otsu's Thresholding . These methods were evaluated to assess their impact on OCR performance with images with varying contrast or background noise.

Simple Thresholding

THRESH_BINARY

THRESH_BINARY_INV

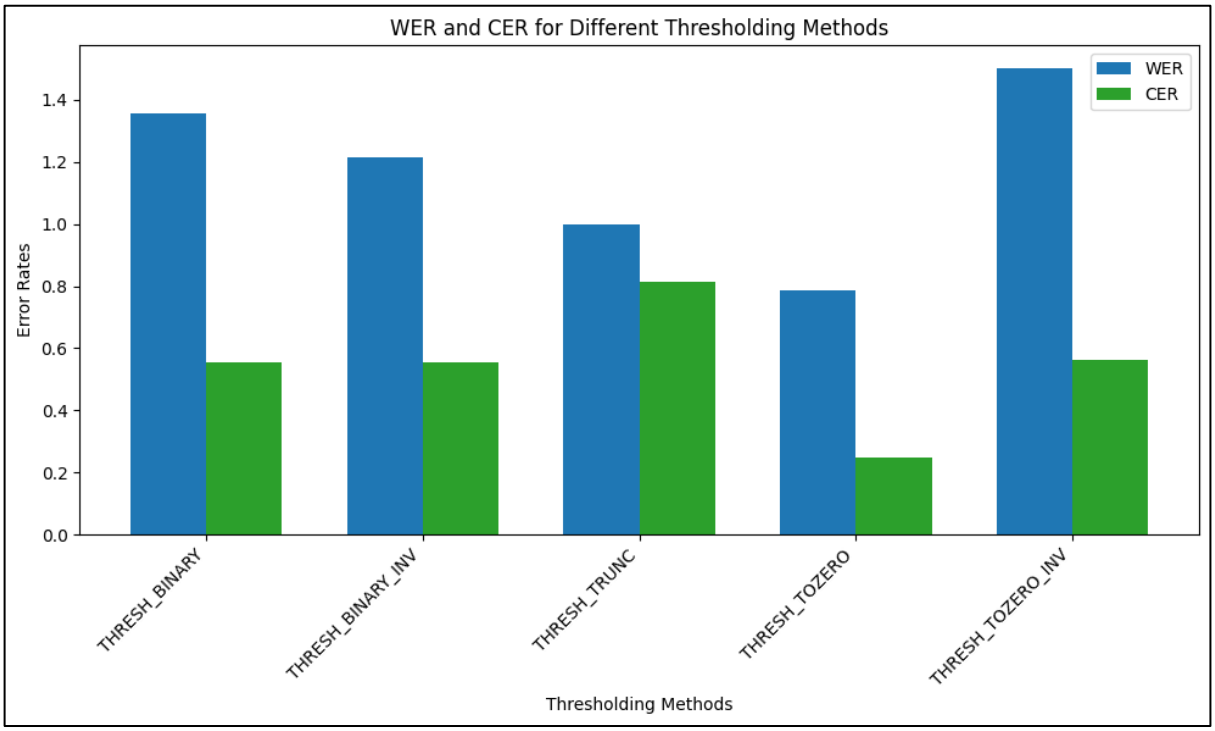
THRESH_TRUNC

THRESH_TOZERO

THRESH_TOZERO_INV

Simple Thresholding on food item with price

	THRESH_BINARY	THRESH_BINARY_INV	THRESH_TRUNC	THRESH_TOZERO	THRESH_TOZERO_INV
WER	1.357	1.214	1.00	0.7857	1.50
CER	0.554	0.554	0.812	0.2475	0.564



Through these results, THRESH_BINARY and THRESH_BINARY_INV resulted in a higher WER which means the basic method of binarization converting all pixels to black or white is not optimal for this receipt image as the contrast is removing essential details that OCR needs to detect the words. THRESH_TRUNC on the other hand has a lower WER as the brightness of the pixels are not fully white which may help preserve some information whilst reducing noise. THRESH_TORENZO however performs the best in reducing WER and CER indicating its more effective in managing this receipt with a little contrast. However overall THRESH_TOZERO_INV has the highest error rate in word recognition as the background is inverted to black which is almost indistinguishable from the text leading to poorer recognition accuracy.

Otsu Thresholding

Original Image

Otsu Thresholding

Otsu Thresholding Accuracy on Food Item with Price

WER	0.5
CER	0.059

Items to extract	Extracted Items
*EVIAN S/C 750ML £1.80	*EVIAN S/O 750ML £1.80
*GRENADE WHITE OREO £2.00	*GRENADE WHITE OREO £2 00
FTG PINEAPPLE 160G £1.35	FTG PINEAPPLE i60G £1.35
*CDBY SLT CRML FINGER £1.50	*CABY SLT CRML FNGER £1.56

Following the initial experiments with simple thresholding, Otsu's Thresholding was tested to automatically calculate an optimal threshold based on the image. The method performed better in some cases by enhancing the clarity of the receipt's text resulting in a reduced WER of 0.5 and CER of 0.059, This technique avoided the need for manually choosing a threshold value making it a good method for automated thresholding in image preprocessing however there were some incorrect extractions which suggests the performance is limited when working with images that may contain less noise or less distinct contrasts leading to partial extractions and some misinterpretations of the text therefore further refinements or additional preprocessing steps may be necessary.

Adaptive Thresholding

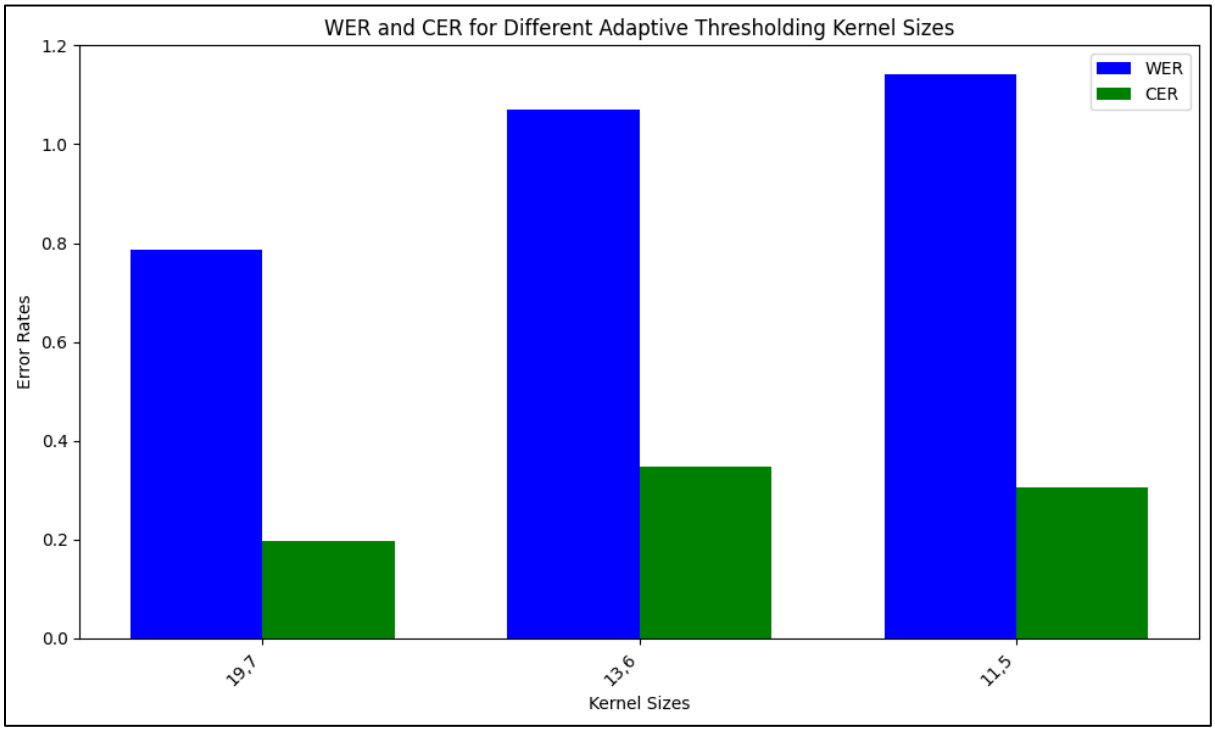
Adaptive thresh with (19,7)

Adaptive thresh with (13,6)

Adaptive thresh with (11,5)

Adaptive Thresholding OCR Accuracy on food item with price

Kernel size	19 , 7	13,6	11,5
WER	0.7857	1.071	1.143
CER	0.198	0.3465	0.3069



Adaptive thresholding was able to make the text in the receipt images more contrasted which helped highlight the characters however for this receipt it also led to some noise in the image affecting the OCR output. The noise caused by the smaller kernel sizes made it difficult for the OCR system to correctly recognize text, leading to higher error rates. The results suggest that adaptive thresholding with larger kernel sizes (e.g 19,7) gives better OCR accuracy. However further techniques such as gaussian blur could improve the performance by reducing background noise while maintaining text clarity therefore was tested next to see if it could improve the accuracy scores.

Based on the previous blurring techniques, the best result parameters for the blur filters were used in adaptive thresholding for each kernel size evaluated before to see its effect on the OCR accuracy scores.

Adaptive Thresholding with Gaussian Blur

Adaptive Thresholded Image for OCR with Gaussian blur 19,7

Adaptive Thresholded Image for OCR with Gaussian blur 11,5

Adaptive Thresholded Image for OCR with Gaussian blur 13,6

Sainsbury's

Good food for all of us

ISLINGTON ST JAMES STREET LOCAL
0330 013 7177

Sainsbury's Supermarkets Ltd
33 Holborn London EC1N 2HT
www.sainsburys.co.uk
Vat Number : 660 4548 36

*EVIAN S/C 750ML	£1.80
*GRENADE WHITE CHED	£2.00
FTG PINEAPPLE 160G	£1.35
*CRBY SLI CRML FNGER	£1.50
4 BALANCE DUE	£6.65
Visa DEBIT	£6.65

contactless

[ICC]	************5478
AID:	A0000000031010
PAN SEQUENCE:	00
MERCHANT:	****5904
AUTH CODE:	003187
TID:	****1110

Cardholder Device Verified

CHANGE £0.00

****

MY NECTAR SUMMARY

[C] **** **** **** ***5032

POINTS EARNED ON	£6.65
PREVIOUS POINTS BALANCE	299
POINTS EARNED	6
NEW POINTS BALANCE	305
YOUR POINTS ARE WORTH	£1.52

****

Check the Nectar app or nectar.com to see any bonus points you might have collected.

***** For a chance to win *****
100,000 Nectar points
please tell us how we did at
lettuce-know.com
****

PLEASE KEEP FOR YOUR RECORDS
PUBLISHED TERMS AND CONDITIONS APPLY

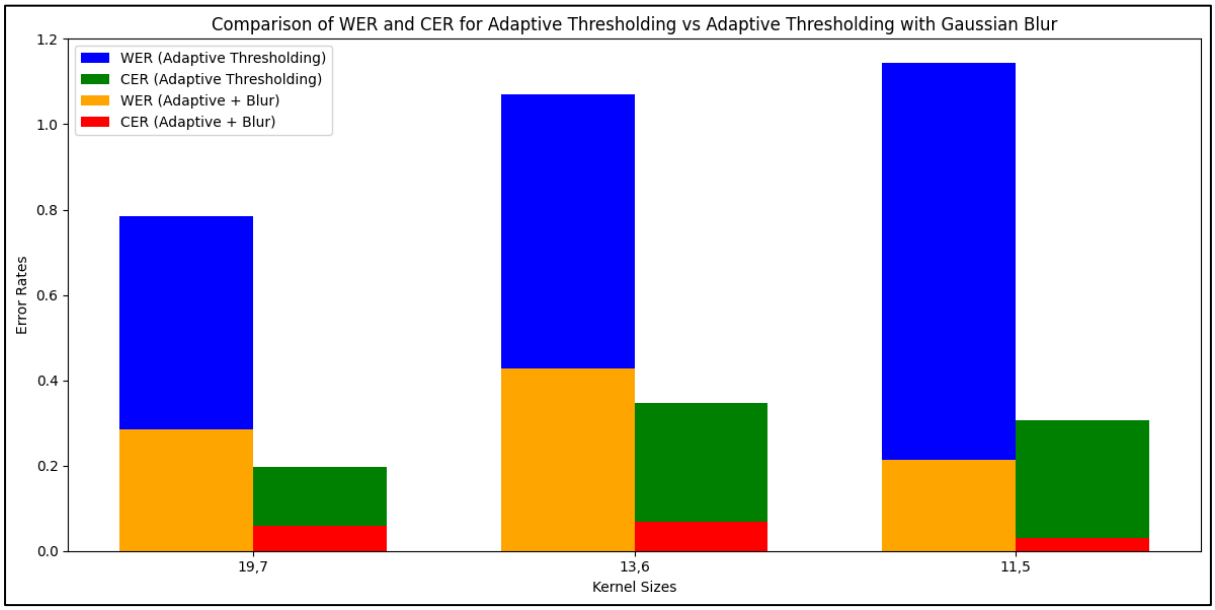
629426008238520060325

C3 #3852 17:18:59 05MAR2025
S4280 R62

Thank you for your visit.

Adaptive Thresholding with Gaussian Blur

Kernel size	19 , 7	13,6	11,5
WER	0.2857	0.4286	0.2143
CER	0.0594	0.0693	0.0297



This graph compares the performance of adaptive thresholding against with gaussian blur with the initial kernel sizes used at first during adaptive thresholding and it shows overall gaussian blur reduced both WER and CER of all the kernel sizes and is most evidently shown with the lowest kernel size of 11,5 as it is significantly lower which means gaussian blur worked really well with 11,5 despite its initial very high WER rate which may make it the optimal choice for this receipt case as it provides a good balance of noise reduction without losing too much detail unlike the larger kernels as it may blur too much despite its initial lower WER without blur.

Adaptive Thresholding with Median Blur

Adaptive Thresholded with median blur 19,7

Sainsbury's

Good food for all of us

TSLINGTON ST JOHN STREET LOCAL
0330 013 7177

Sainsbury's Supermarkets Ltd
33 Holborn London EC1N 2HT
www.sainsburys.co.uk
Vat Number : 660 4548 36

*EVIAN S/C 750ML	£1.80
*GRENADE WHITE ONEO	£2.00
FTG PINEAPPLE 160G	£1.35
*CDBY SLT CRML FNGER	£1.50
4 BALANCE DUE	£6.65
Visa DEBIT	£6.65

contactless

[ICC] **********	5478
AID:	A0000000031010
PAN SEQUENCE:	00
MERCHANT:	****5904
AUTH CODE:	003187
TID:	****1110

Cardholder Device Verified

CHANGE £0.00

MY NECTAR SUMMARY

[C] **** **** **** ***5032

POINTS EARNED ON	£6.65
PREVIOUS POINTS BALANCE	299
POINTS EARNED	6
NEW POINTS BALANCE	305
YOUR POINTS ARE WORTH	£1.52

Check the Nectar app or nectar.com to see any bonus points you might have collected.

***** For a chance to win *****
100,000 Nectar points
please tell us how we did at
lettuce-know.com

PLEASE KEEP FOR YOUR RECORDS
PUBLISHED TERMS AND CONDITIONS APPLY

62942800623852000503025

C3 #3852 17:18:59 05MAR2025
S4280 R62

Thank you for your visit.

Adaptive Thresholded with median blur 13,6

Sainsbury's

Good food for all of us

TSLINGTON ST JOHN STREET LOCAL
0330 013 7177

Sainsbury's Supermarkets Ltd
33 Holborn London EC1N 2HT
www.sainsburys.co.uk
Vat Number : 660 4548 36

*EVIAN S/C 750ML	£1.80
*GRENADE WHITE ONEO	£2.00
FTG PINEAPPLE 160G	£1.35
*CDBY SLT CRML FNGER	£1.50
4 BALANCE DUE	£6.65
Visa DEBIT	£6.65

contactless

[ICC] **********	5478
AID:	A0000000031010
PAN SEQUENCE:	00
MERCHANT:	****5904
AUTH CODE:	003187
TID:	****1110

Cardholder Device Verified

CHANGE £0.00

MY NECTAR SUMMARY

[C] **** **** **** ***5032

POINTS EARNED ON	£6.65
PREVIOUS POINTS BALANCE	299
POINTS EARNED	6
NEW POINTS BALANCE	305
YOUR POINTS ARE WORTH	£1.52

Check the Nectar app or nectar.com to see any bonus points you might have collected.

***** For a chance to win *****
100,000 Nectar points
please tell us how we did at
lettuce-know.com

PLEASE KEEP FOR YOUR RECORDS
PUBLISHED TERMS AND CONDITIONS APPLY

62942800623852000503025

C3 #3852 17:18:59 05MAR2025
S4280 R62

Thank you for your visit.

Adaptive Thresholded with median blur 11,5

Sainsbury's

Good food for all of us

TSLINGTON ST JOHN STREET LOCAL
0330 013 7177

Sainsbury's Supermarkets Ltd
33 Holborn London EC1N 2HT
www.sainsburys.co.uk
Vat Number : 660 4548 36

*EVIAN S/C 750ML	£1.80
*GRENADE WHITE ONEO	£2.00
FTG PINEAPPLE 160G	£1.35
*CDBY SLT CRML FNGER	£1.50
4 BALANCE DUE	£6.65
Visa DEBIT	£6.65

contactless

[ICC] **********	5478
AID:	A0000000031010
PAN SEQUENCE:	00
MERCHANT:	****5904
AUTH CODE:	003187
TID:	****1110

Cardholder Device Verified

CHANGE £0.00

MY NECTAR SUMMARY

[C] **** **** **** ***5032

POINTS EARNED ON	£6.65
PREVIOUS POINTS BALANCE	299
POINTS EARNED	6
NEW POINTS BALANCE	305
YOUR POINTS ARE WORTH	£1.52

Check the Nectar app or nectar.com to see any bonus points you might have collected.

***** For a chance to win *****
100,000 Nectar points
please tell us how we did at
lettuce-know.com

PLEASE KEEP FOR YOUR RECORDS
PUBLISHED TERMS AND CONDITIONS APPLY

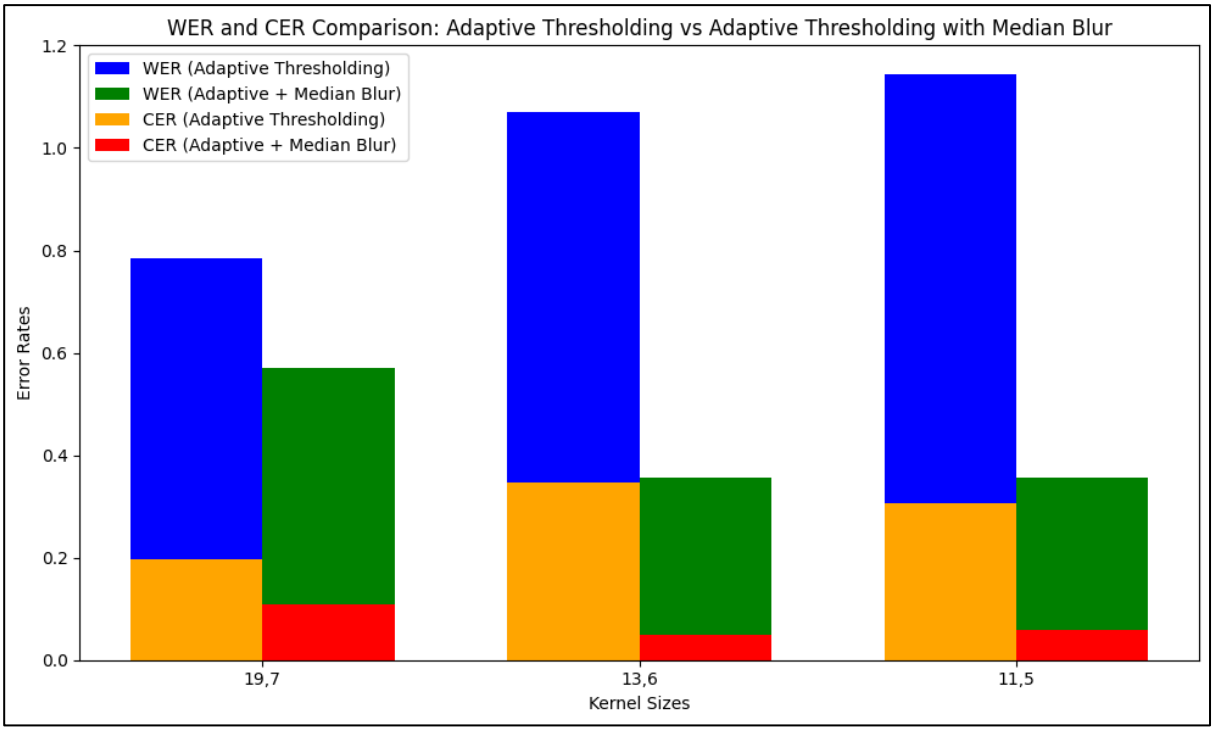
62942800623852000503025

C3 #3852 17:18:59 05MAR2025
S4280 R62

Thank you for your visit.

Adaptive Thresholding with Median Blur

Kernel size	19 , 7	13,6	11,5
WER	0.5714	0.3571	0.357
CER	0.1089	0.0495	0.0594



From this chart, Adaptive Thresholding with Median blur produces better results than adaptive thresholding alone across all kernel sizes, where 19,7 proves to be the best choice for WER as the larger kernels provide better results for identifying words despite the small increase in CER where the goal is to extract food items and their prices its more important to ensure the OCR correctly identifies individual items rather than incorrect words.

Adaptive Thresholding with Bilateral filter

Adaptive Thresholding with Bilateral Filter 19,7

Sainsbury's
Good food for all of us
151 Holborn St Jon Ward Lond
0330 013 7777

Sainsbury's Supermarkets Ltd
33 Holborn London EC1N 2HT
www.sainsburys.co.uk
Vat Number : 660 4548 36

*EVIAN 5x0 75dCL	£1.80
*BREHANE WHITE BULB	£2.40
FTG PINEAPPLE 1b66	£1.35
*CDBY SLT CRML FNGLR	£1.50
4 BALANCE DUE	£6.65
Visa DEBIT	£6.65

contactless

[ICC] ***********5476	
AID:	A0000000031010
PAN SEQUENCE:	00
MERCHANT:	****5904
AUTH CODE:	003187
TID:	****1110

Cardholder Device Verified

CHANGE £0.00

MY NECTAR SUMMARY

[C] **** **** **** ***5032	
POINTS EARNED ON	£6.65
PREVIOUS POINTS BALANCE	299
POINTS EARNED	6
NEW POINTS BALANCE:	305
YOUR POINTS ARE WORTH	£1.52

Check the Nectar app or nectar.com to see any bonus points you might have collected.

***** For a chance to win *****
100,000 Nectar points
please tell us how we did at
lettuce-know.com

PLEASE KEEP FOR YOUR RECORDS
PUBLISHED TERMS AND CONDITIONS APPLY

C3 #3852 17:10:59 05MAR2025
S4280 R62

Thank you for your visit.

Adaptive Thresholding with Bilateral Filter 13,6

Sainsbury's
Good food for all of us
151 Holborn St Jon Ward Lond
0330 013 7777

Sainsbury's Supermarkets Ltd
33 Holborn London EC1N 2HT
www.sainsburys.co.uk
Vat Number : 660 4548 36

*EVIAN 5x0 75dCL	£1.80
*BREHANE WHITE BULB	£2.40
FTG PINEAPPLE 1b66	£1.35
*CDBY SLT CRML FNGLR	£1.50
4 BALANCE DUE	£6.65
Visa DEBIT	£6.65

contactless

[ICC] ***********5476	
AID:	A0000000031010
PAN SEQUENCE:	00
MERCHANT:	****5904
AUTH CODE:	003187
TID:	****1110

Cardholder Device Verified

CHANGE £0.00

MY NECTAR SUMMARY

[C] **** **** **** ***5032	
POINTS EARNED ON	£6.65
PREVIOUS POINTS BALANCE	299
POINTS EARNED	6
NEW POINTS BALANCE:	305
YOUR POINTS ARE WORTH	£1.52

Check the Nectar app or nectar.com to see any bonus points you might have collected.

***** For a chance to win *****
100,000 Nectar points
please tell us how we did at
lettuce-know.com

PLEASE KEEP FOR YOUR RECORDS
PUBLISHED TERMS AND CONDITIONS APPLY

C3 #3852 17:10:59 05MAR2025
S4280 R62

Thank you for your visit.

Adaptive Thresholding with Bilateral Filter 11,5

Sainsbury's
Good food for all of us
151 Holborn St Jon Ward Lond
0330 013 7777

Sainsbury's Supermarkets Ltd
33 Holborn London EC1N 2HT
www.sainsburys.co.uk
Vat Number : 660 4548 36

*EVIAN 5x0 75dCL	£1.80
*BREHANE WHITE BULB	£2.40
FTG PINEAPPLE 1b66	£1.35
*CDBY SLT CRML FNGLR	£1.50
4 BALANCE DUE	£6.65
Visa DEBIT	£6.65

contactless

[ICC] ***********5476	
AID:	A0000000031010
PAN SEQUENCE:	00
MERCHANT:	****5904
AUTH CODE:	003187
TID:	****1110

Cardholder Device Verified

CHANGE £0.00

MY NECTAR SUMMARY

[C] **** **** **** ***5032	
POINTS EARNED ON	£6.65
PREVIOUS POINTS BALANCE	299
POINTS EARNED	6
NEW POINTS BALANCE:	305
YOUR POINTS ARE WORTH	£1.52

Check the Nectar app or nectar.com to see any bonus points you might have collected.

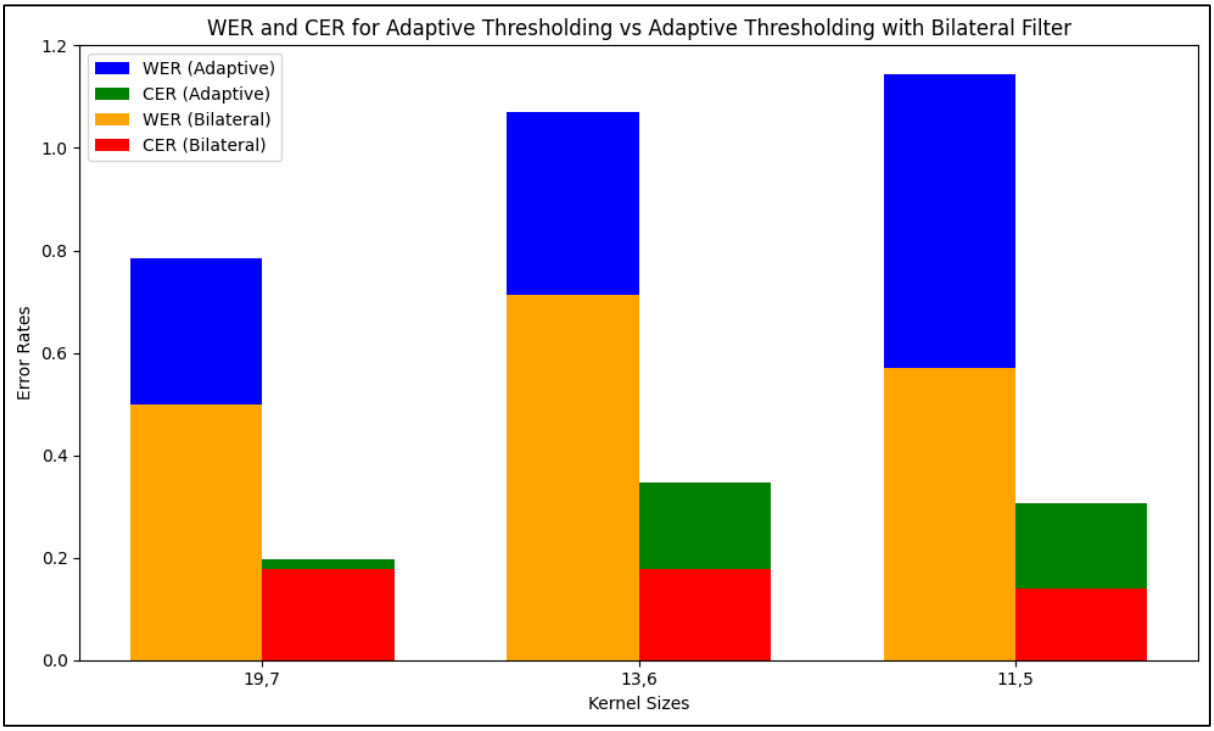
***** For a chance to win *****
100,000 Nectar points
please tell us how we did at
lettuce-know.com

PLEASE KEEP FOR YOUR RECORDS
PUBLISHED TERMS AND CONDITIONS APPLY

C3 #3852 17:10:59 05MAR2025
S4280 R62

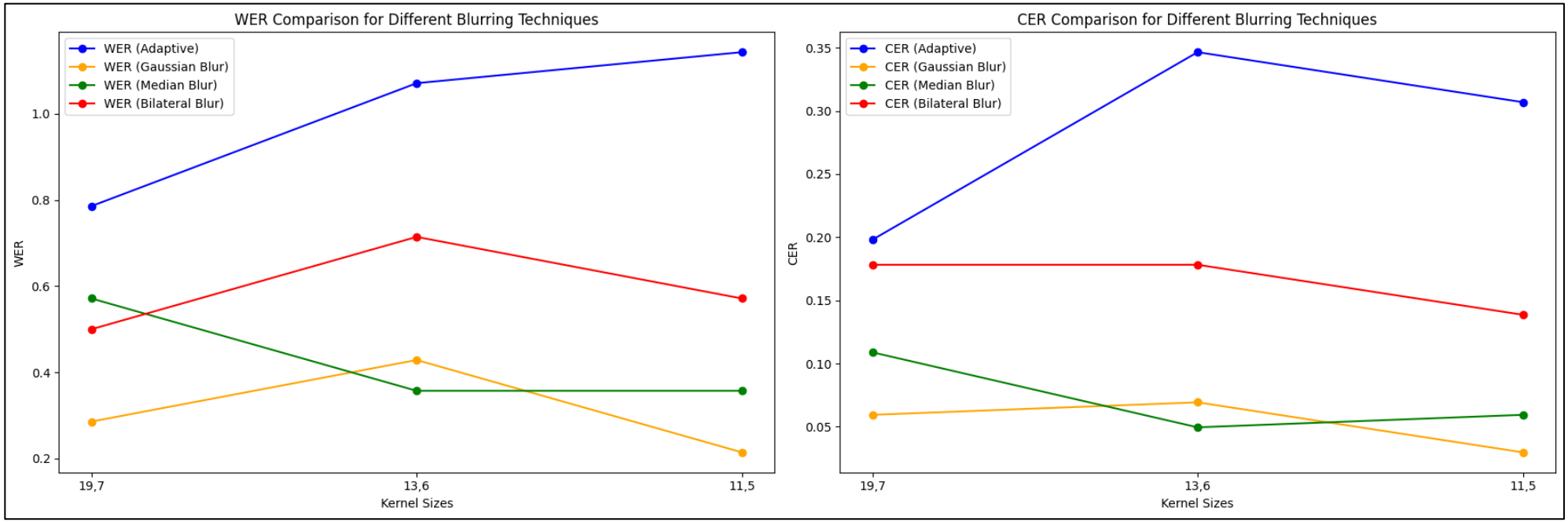
Thank you for your visit.

	Adaptive Thresholding with Bilateral filter		
Kernel size	19 , 7	13,6	11,5
WER	0.5	0.7143	0.5714
CER	0.1782	0.1782	0.1386



The results show that applying adaptive thresholding with a bilateral filter leads to lower WER and CER especially with 13,6 and 11,5 showing drastic improvements which suggests that preprocessing techniques like this are crucial for improving OCR accuracy.

Overall comparison



Whilst comparing all the different blurring techniques combined with adaptive thresholding for OCR accuracy, Gaussian blur with a kernel size of 11,5 provided the most promising results with an exceptionally low WER indicating very well recognition of food items and a low CER, so the individual characters were also well recognised which suggests for this receipt case gaussian blur helped clean up the image effectively reducing noise while preserving important text details making this combination the optimal choice for this system.

Regex Pattern

In section 3.3.2, the context chapter emphasized the use of regular expressions to filter out irrelevant text after OCR preprocessing. Referring to figure 5.1, to improve the OCR extraction custom regex patterns where created to extract necessary information relating to the food items whilst capturing with the price tag to follow its structure, and excluding unnecessary information for this feasible test such as the measurements of food items and the price when extracted as part of post processing as mention in section 3.3.2, how Post-processing in OCR systems can significantly enhance the accuracy of the extracted data, especially after the use of regex patterns.

Initial Receipt

Receipt

Sainsbury's
Good food for all of us
TSLINGIGN ST JGHN STREET LOCAI
0330 013 7177

Sainsbury's Supermarkets Ltd
33 Holborn London ECIN 2HT
www. sainsburys .co.uk
Vat Number : 660 4548 36

*EVIAN S/C 750ML £1.80
*GRENADE WHITE OREO £2.00
FTG PINEAPPLE i60G £1.35
*CDBY SLT CRML FNGER £1.56
4 BALANCE DUE £6.65

Visa DEBIT £6.65
contactless }}}}
LICC] xxxxxxxxxxxx5478

AID: AQ000006031010
PAN SEQUENCE : 00
MERCHANT : xxx x5904
AUTH CODE: 03187
TID: *x*x% 1110

Cardholder Device Verified
CHANGE £0.00
KXAN KANKEK KEKE KKRKRA KEE KRAKEKRK KER ERE RRR KEKE
MY NECTAR SUMMARY
(C] xwx% xxkx xxxx #x%5092
POINTS EARNED ON £6.65
PREVIOUS POINTS BALANCE 299
POINTS EARNED 6
NEW PGINTS BALANCE 305
YOUR POINTS ARE WORTH £1.52
KARKARKAKARNKKARKKAREGRERKKRRARRRERERKEKERNE
Check the Nectar app or nectar.com to see
any bonus points you might have collected.
xaxax For a chance to win x«««s
100,000 Nectar points
please tell us how we did at
lettuce-know. com
TRKKEKRRERRRRRRKRRKKRRERKEK
PLEASE KEEP FOR YOUR RECORDS
PUBLISHED TERMS AND CONDITIGNS APPLY
itl
6294280062385 200050325.
C3 #3852 17:18:59 O5MAR2095
$4280 R62
'Thark you for your visit.

	Regex pattern
Focusing on extracting all text aligned with a number for the price	pricePattern = r'([0-9]+\.[0-9]+)'

This pattern captures numbers in a typical decimal format found in receipts.

Desired Output	**Extracted Items**
EVIAN S/C	*EVIAN S/C 750ML £1.80 *GRENADE WHITE OREO £2.00 FTG PINEAPPLE i60G £1.35 *CDBY SLT CRML FNGER £1.56 4 BALANCE DUE £6.65 Visa DEBIT £6.65 CHANGE £0.00 POINTS EARNED ON £6.65 YOUR POINTS ARE WORTH £1.52
GRENADE WHITE OREO	
FTG PINEAPPLE	
CADBURY SALTED CARAMEL FINGER	

This initial simple pattern extracts all the food items however contains irrelevant details as well as unnecessary symbols (*) with mixed upper and lower cases and so another pattern was created to manage these complexities

	Updated Regex pattern
Focusing on extracting food item matching to food item and price	Pattern = r'^(.*?)\s*£(\d+[.,]\d{2})$'
Skipping common words in receipt structure	'balance due', 'visa debit', 'total', 'point', 'change'

Dictionary of abbreviations	abbreviations = {
	"CDBY": "Cadbury",
	"SLT": "Salted",
	"CRML": "Caramel",
	"FNGER": "Finger",
	}

Replace "i" with "1" when followed by digits and "G"	r'\bi(\d+G)\b'
Match and remove measurement units from the item names	r'\s*\d+[A-Za-z]+'

Desired Output	Extracted Items
EVIAN S/C	EVIAN S/C GRENADE WHITE OREO FTG PINEAPPLE CADBURY SALTED CARAMEL FINGER
GRENADE WHITE OREO	
FTG PINEAPPLE	
CADBURY SALTED CARAMEL FINGER	

5.5 Sprint 3 results

Aim:	Sprint 3 – Database Matching
Methods: **Token_sort_ratio** **Token_set_ratio** **Partial_token_sort_ratio** **ratio**	Applying : Fuzzy matching techniques to match the food item to food database.

In this sprint, the objective was to perform database matching by linking the extracted food item names to the food database using the best fuzzy string-matching technique. Here a custom food database is created with predicted expiry dates and then matched with fuzzy matching.

Food database

Item name	**Expiry date**
evian	2025-12-31
grenade white oreo	2025-05-15
pineapple	2025-05-01
pineapple pizza	2025-06-01
cadbury salted caramel finger	2025-05-10
cadbury finger	2025-05-10
apple pie	2025-05-10
apple	2025-05-10
chocolate oreo	2025-05-10

Example food items extracted
evian s/c
Grenade white oreo
ftg pineapple
Cadbury salted caramel finger
apple

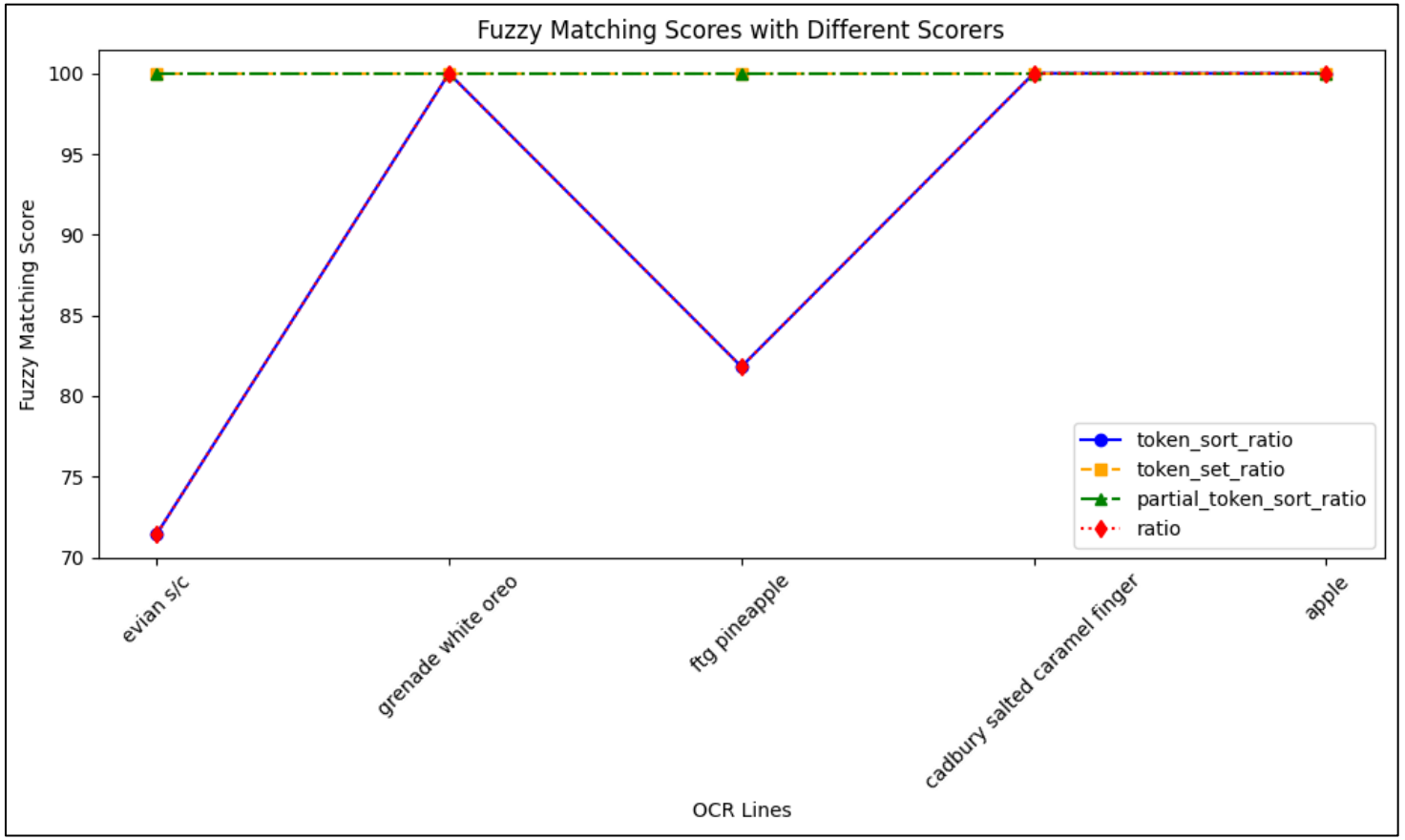


Figure 5.3 *Fuzzy matching scores*

Despite the promising results of token set ratio and partial token sort ratio further breakdown of accuracy scores reveal they are incorrectly matched and in fact token sort ratio and ratio does better. Token sort ratio displayed flexibility by correctly matching OCR extracted variations which is a realistic scenario for a OCR based system dealing with varied item extractions therefore is a important feature as it demonstrates the functions robustness in recognizing food items that are written with slight variations. Below is a table breakdown of the matches with its percentage rate.

Accuracy

Key:

	Correctly matched food item with correct percentage
	Correctly matched with wrong percentage
	Incorrectly matched

Fuzzy matching techniques			
Technique	Food item	Matched with	Percentage
Token_sort_ratio	evian s/c	evian	71.428
	grenade white oreo	grenade white oreo	100.0
	ftg pineapple	pineapple	81.818
	cadbury salted caramel finger	cadbury salted caramel finger	100.0
	apple	apple	100.0
Token_set_ratio	evian s/c	evian	100.0
	grenade white oreo	grenade white oreo	100.0
	ftg pineapple	pineapple	100.0
	cadbury salted caramel finger	cadbury salted caramel finger	100.0
	apple	apple pie	100.0
partial_token_sort_ratio	evian s/c	evian	100.0
	grenade white oreo	grenade white oreo	100.0
	ftg pineapple	pineapple	100.0
	cadbury salted caramel finger	cadbury salted caramel finger	100.0
	apple	pineapple	100.0
ratio	evian s/c	evian	71.428
	grenade white oreo	grenade white oreo	100.0
	ftg pineapple	pineapple	81.818

	cadbury salted caramel finger	cadbury salted caramel finger	100.0
	apple	apple	100.0

To summarise the results, it's important to acknowledge that OCR derived text often contains inaccuracies, abbreviations due to variations in printing or formatting on receipts as discussed in chapter 3.3.4. Therefore perfect matching of every word is not always feasible or necessary for the system's success. While both token sort ratio and ratio perform well, since ratio would be more sensitive to any changes in word order , Token_sort_ratio was implemented in the system.

5.6 Sprint 4 results

Aim:	Sprint 4– Web app
Creating a user friendly web app integrating the ai pipeline from previous sprints	Applying : All previous sprints

5.6.1 Design

The following section will be about my diagrams created for my requirements refer to Appendix N

5.6.1.1 Use case diagram

To understand how end users could interact with the prototype system, a use case diagram was designed this diagram models the primary interactions of this proof of concept

5.6.1.2 Entity Relationship Diagram

Informed by section 4.4.1.3, the database structure includes two databases, Pantry and food database with minimal data relationships needed for the feasible testing to support this as user accounts and multiple pantries were out of scope.

5.6.1.3 State Diagram

To visualize the ai pipeline, a state diagram was created to help guide the logic of the web app.

5.6.1.4 User Interface Design

The UI was designed in Figma as a prototype and then later implemented using flask, HTML/ CSS (referring to section 4.6.2) a minimalist dashboard layout was chosen to prioritize clarity and keeping it easy to test the feasibility of AI scanning.

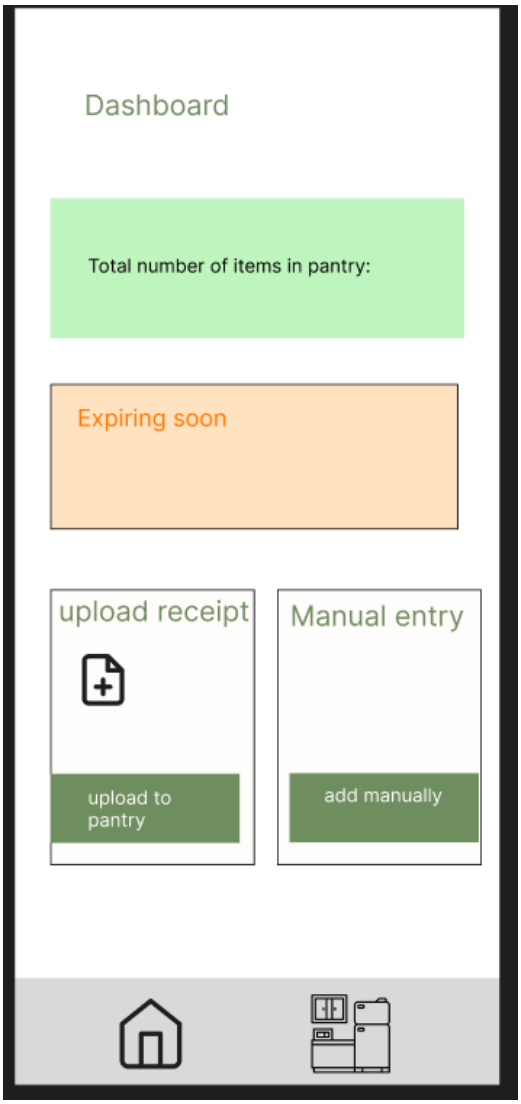


figure 5.4.1 *Home page GUI*

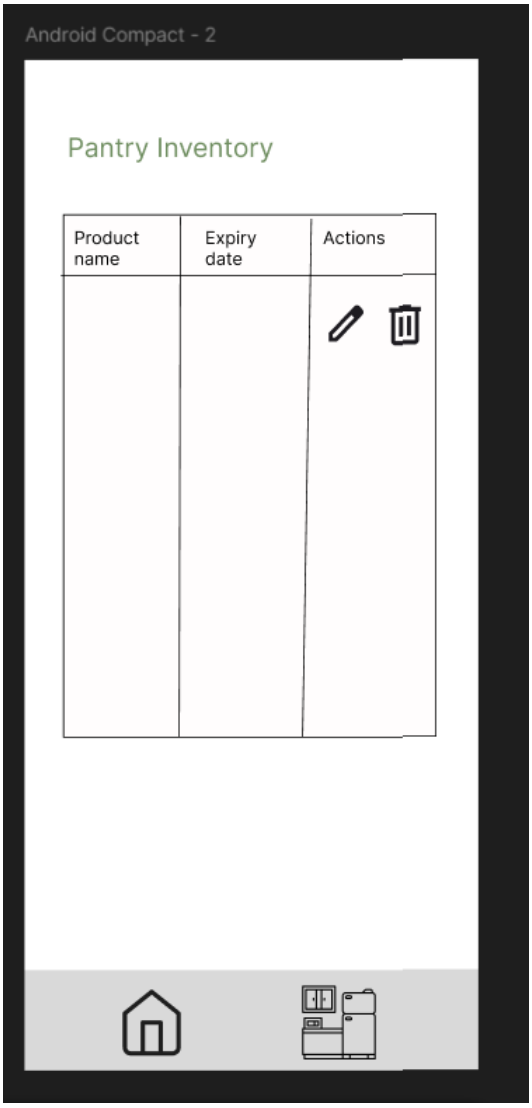


figure 5.4.2 *Pantry page GUI*

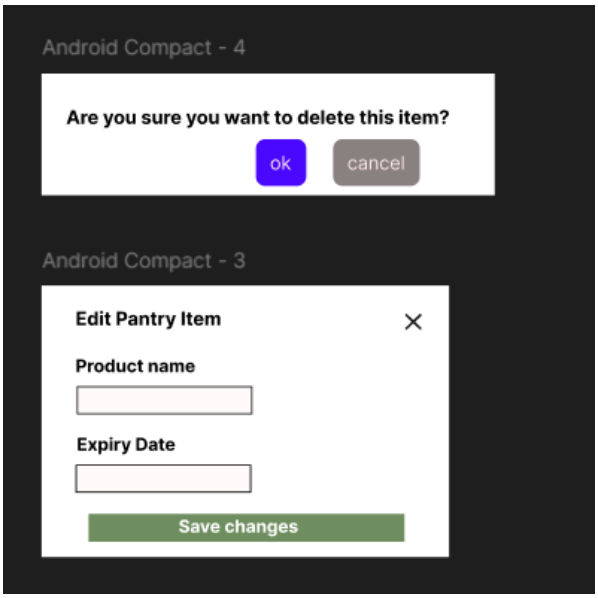


figure 5.4.3 *Edit and Delete in Pantry view GUI*

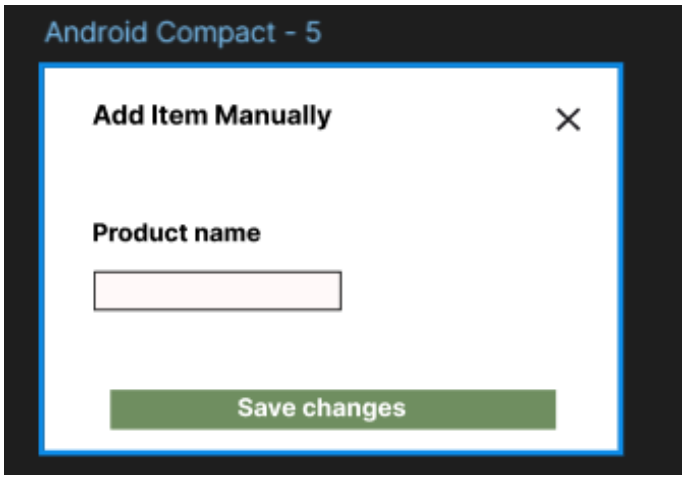


figure 5.4.4 *adding items manually in index page GUI*

5.6.2 Project Setup

5.6.2.1 Dependencies

As outlined in section 4.2.1, the Flask- based system was developed using a series of python libraries essential for implementing the OCR pipeline, and database integrations. These dependencies were installed and kept within a virtual environment and documented in a requirements.txt file to support future testing.

5.6.2.1 Folder Structure and File Organization

The folder structure was set up to reflect clarity between the static assets, template files, and data storage shown in figure 5.4

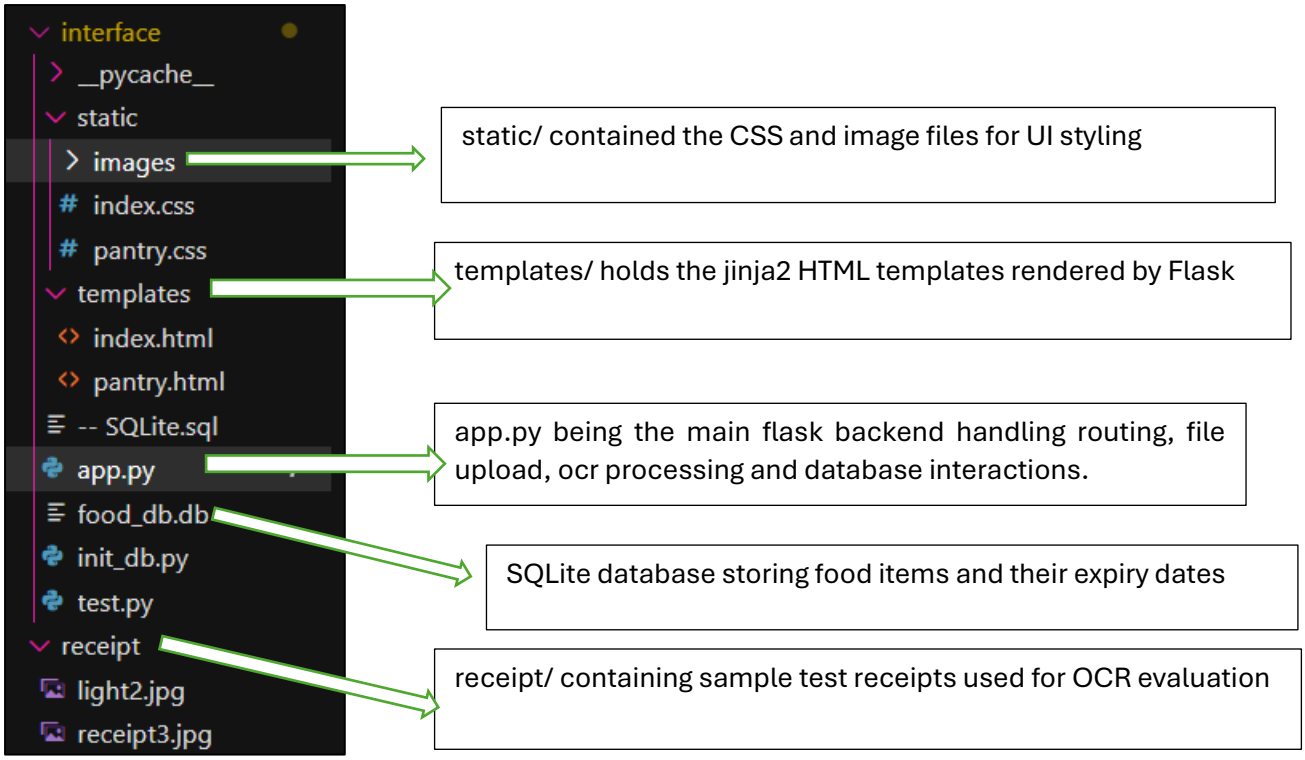


Figure 5.4 *Folder structure*

5.7 Implementation

5.7.1 Flask Web Application Development

The system was developed using Flask, a python micro web framework (Pythonbasics.org, 2021) to build a web app and connect the AI pipeline with the user interface ,meeting the functional requirements from section 4.4.1, allowing simulated user interactions where each view was rendered using jinja templates and data dynamically passed using Flask route logic therefore serving as the backbone of the web apps interactivity.

5.7.1.2 Flask routing

The routing was handled and configured in app,py as detailed in Appendix F, it was used to map URLS to specific Python functions through @app.route() where each route returned dynamic HTML views using render_template(), populated with data from the SQLite database using jinja2 (see section) for example:

```
return render_template('pantry.html', pantry_contents=pantry_contents)
```

5.7.1.3 OCR and AI Pipeline integration

Once a user uploads a receipt in the index page, the backend reads the image using pytesseract for OCR after applying OpenCV preprocessing steps, including grayscale conversion, Gaussian blur and adaptive thresholding to enhance text clarity in the code snippet shown below.

```
def extract_food_items_from_receipt(image_stream):
    img = Image.open(image_stream)

    # Converting to grayscale using PIL
    gray_img = img.convert('L')

    # Converting PIL image to NumPy array for OpenCV processing
    gray_cv = np.array(gray_img)

    #applying Gaussian Blur
    blurred_cv = cv2.GaussianBlur(gray_cv, (7, 5), 0)

    #applying Adaptive Thresholding
    adaptive_thresh = cv2.adaptiveThreshold(
        blurred_cv, 255,
        cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
        cv2.THRESH_BINARY,
        11, 5
    )
  ocr_text = pytesseract.image_to_string(Image.fromarray(adaptive_thresh), config="--psm 4")
```

As discussed in section 4.5, Regex was then applied to extract the food items and prices from the OCR output, below showing the code snippet

```
# Splitting OCR text into lines and removing empty ones.
 lines = [line.strip() for line in ocr_text.splitlines() if line.strip()]


 # Regex pattern to extract food name and price.
 pattern = re.compile(r'^[\*]?(.*?)\s*£(\d+[.,]\d{2})$', re.IGNORECASE)


 # Keywords to filter out key words.
 filter_keywords = ['balance due', 'visa debit', 'total', 'point', 'change','amount']
 food_items = []
 for line in lines:


   if any(keyword in line.lower() for keyword in filter_keywords):
     continue


   match = pattern.search(line)
   if match:
     item_name = match.group(1).strip()
     # if i instead of 1 in measurement convert to 1
     item_name = re.sub(r'\bi(\d+G)\b', r'1\1', item_name)
       for abbr, full_name in abbreviations.items():
        item_name = item_name.replace(abbr, full_name)
     # Clean the product name by expanding abbreviations, removing measurements.
      item_name = item_name.lower()
     item_name = remove_measurements(item_name)
     food_items.append(item_name)
```

5.7.1.4 Fuzzy Matching and Database Logic

To support the automated expiry predictions, the system relied on a local SQLite database which had two tables:

	Databases
FoodDatabase	containing predefined food item names with its associated expiry dates

Pantry	A dynamic table updated with items extracted from the receipt along with their matched expiry dates.

As mentioned in section 4.6.3 and 4.4.1.3, the FoodDatabase was manually populated during development to simulate the expiry logic to allow fuzzy matching to retrieve and assign expiry values accurately during the receipt upload process.

Extracted food items after preprocessing were then matched against a local food database using the fuzz.token_sort_ratio function which was chosen during sprint 3 with a matching threshold set at 70 as figure 5.3 shows after around 70% it recognised all the food items, so this was used as the threshold, see below the code snippet.

```
def match_food_item(ocr_item, food_db_items, threshold=70):
    match = process.extractOne(ocr_item, [item[0] for item in food_db_items], scorer=fuzz.token_sort_ratio)
```

On a successful match it was matched and inserted into the pantry table using:

```
cursor.execute("INSERT INTO Pantry (name, expirydate) VALUES (?, ?)", (item_name, expiry_date))
```

Which became visible in the Pantry UI.

5.7.1.5 Jinja2 Templating and HTML rendering

As described in section 4.6 and Appendix G, using flasks render_Template() which supports jinja2. This was used to loop through data structures to embed python into the HTML, below is a example code where the index.html displayed the total pantry items. The system calculates this by querying the pantry database which was then passed into the front-end shown below

```
<p>Total Items in Pantry: <strong>{{ total_items }}</strong></p>
```

Total_items was calculated in python by:

```
pantry_contents = get_pantry_contents()

total_items = len(pantry_contents)
```

The number updates automatically without needing to refresh the database manually.

5.7.2 HTML Pages and UI Functionality

5.7.2.1 Index page

This was the primary dashboard where it allowed users to upload a receipt and view total pantry items, notifications for items expiring soon (Set at 30 for testing) and a simple form to submit image files. Data was passed from flask to the front end using dynamic variables such as {{ total_items}} (see section 5.7.1.5 for the code) and {{notifications}} to render the information using Jinja2

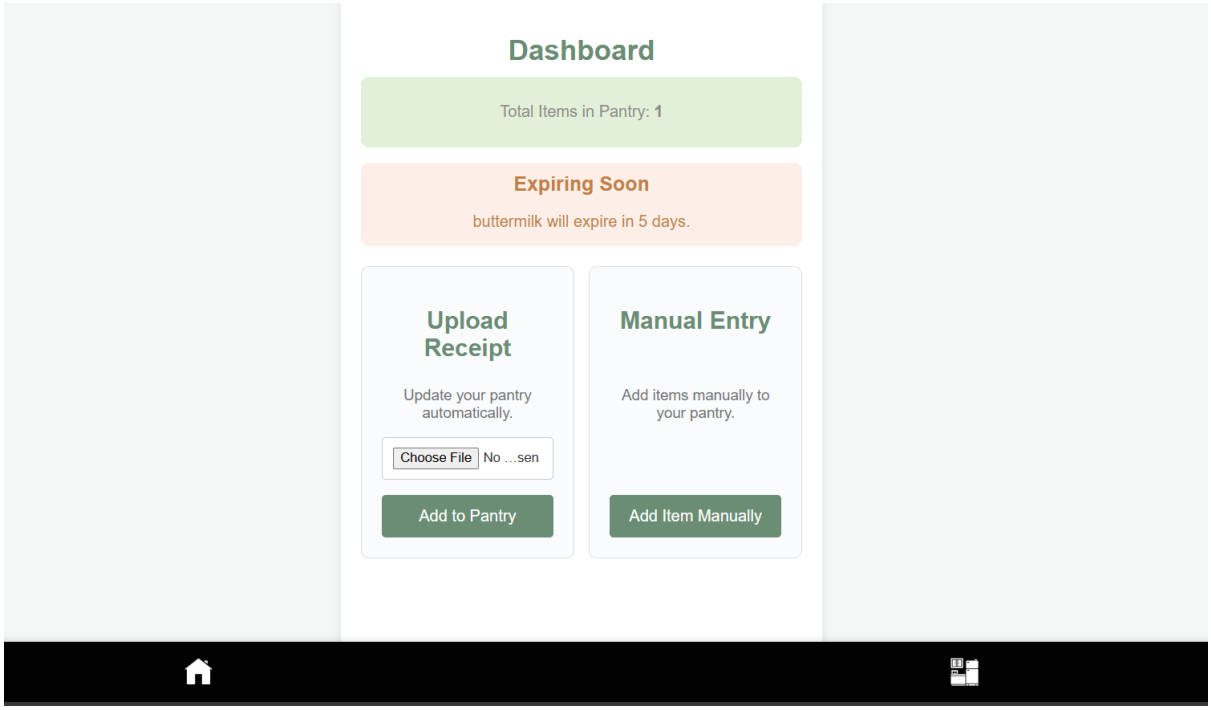


Figure 5.5 Dashboard UI on web browser

5.7.2.2 Expiry Notifications system

To help users prioritize what to use to reduce waste, a notification system was implemented to display items expiring soon (which was set to 30 days for the testing). A function def get_expiring_notifications(*threshold*=30) compared the expiry date of each pantry item with the current date using the code below

```
expiry_date = datetime.strptime(row['expirydate'], '%Y-%m-%d').date()
if today <= expiry_date <= threshold_date:
    notifications.append((row['name'], days_left))
```

Which was then dynamically rendered in the index page using jinja shown in the code below

```
{% if notifications %}
<div class="notifications">
  <h2>Expiring Soon</h2>
  <ul>
    {% for note in notifications %}
      <li>{{ note }}</li>
    {% endfor %}
  </ul>
</div>
{% endif %}
```

This feature aligned with the projects goal of supporting better pantry management and minimizing food wastage, key motivation points discussed in section 1.2.

5.7.2.3 Manual input entry – fallback

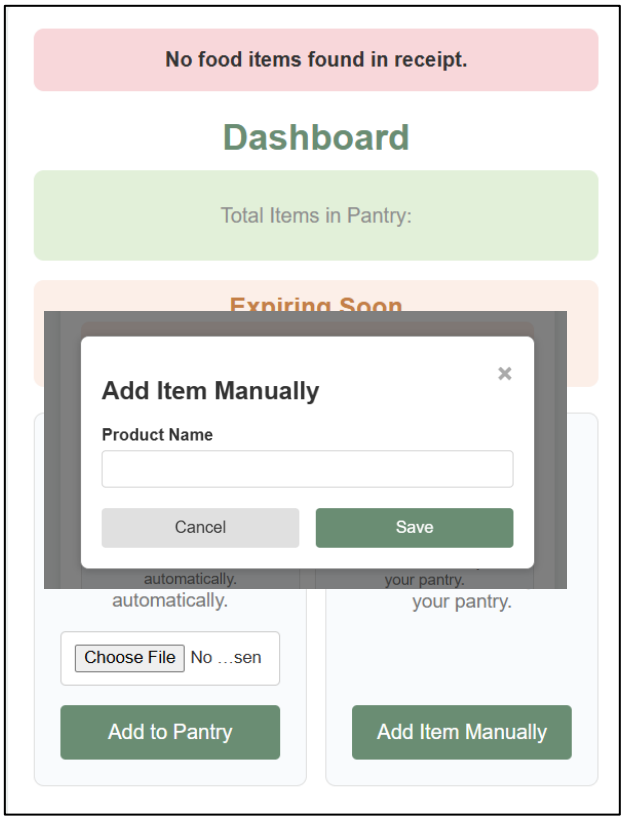


Figure 5.6 *error message for food items not found and manual input entry*

As a fall back method, if the ocr cannot detect food items, a error message would show and users can simply add the product name manually where the database would match it to the food database to retrieve the expiry date and update pantry.

5.7.2.2 Pantry Page

The Pantry page displayed a complete list of pantry items with its food name and expiry date as well as a edit button next to it that triggers a model to update name or expiry date, and a delete button.

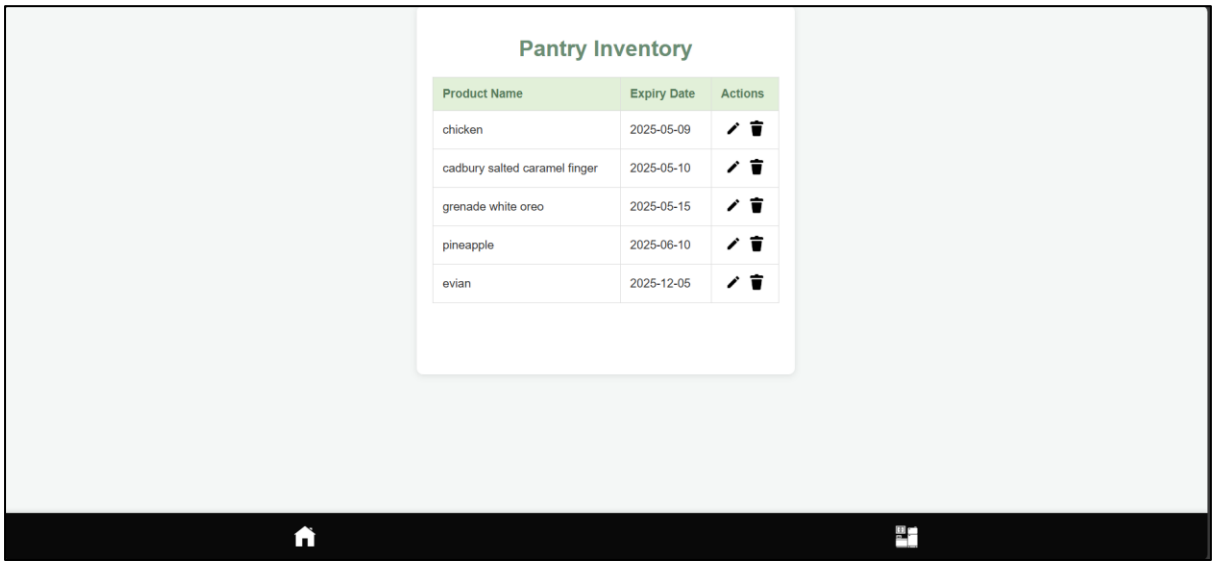
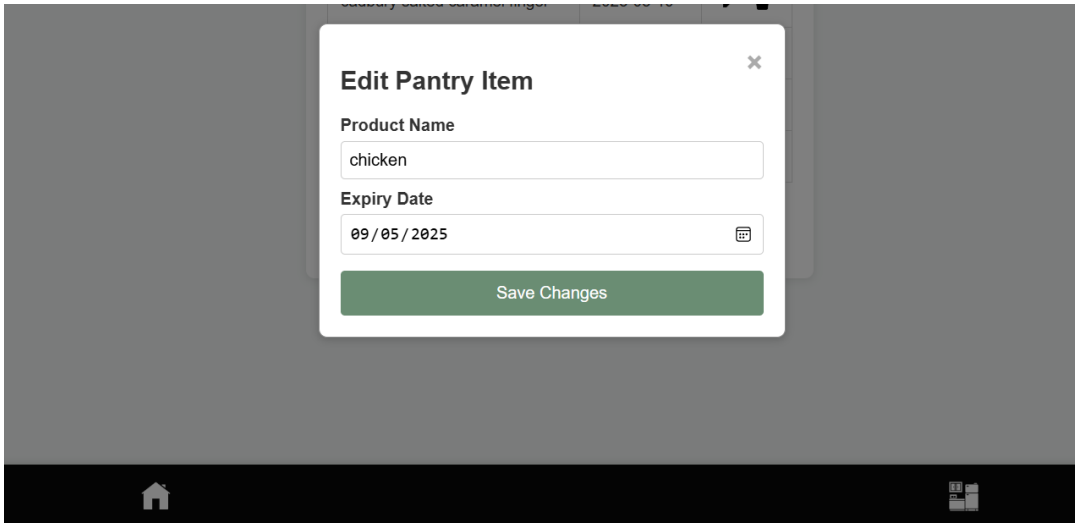


Figure 5.6 *Pantry UI on the web browser*

The HTML modals used JavaScript to allow users to update entries or delete entries and were implemented to simulate real world management where users might spot OCR mistakes such as label names or incorrect expiry dates and correct them easily.



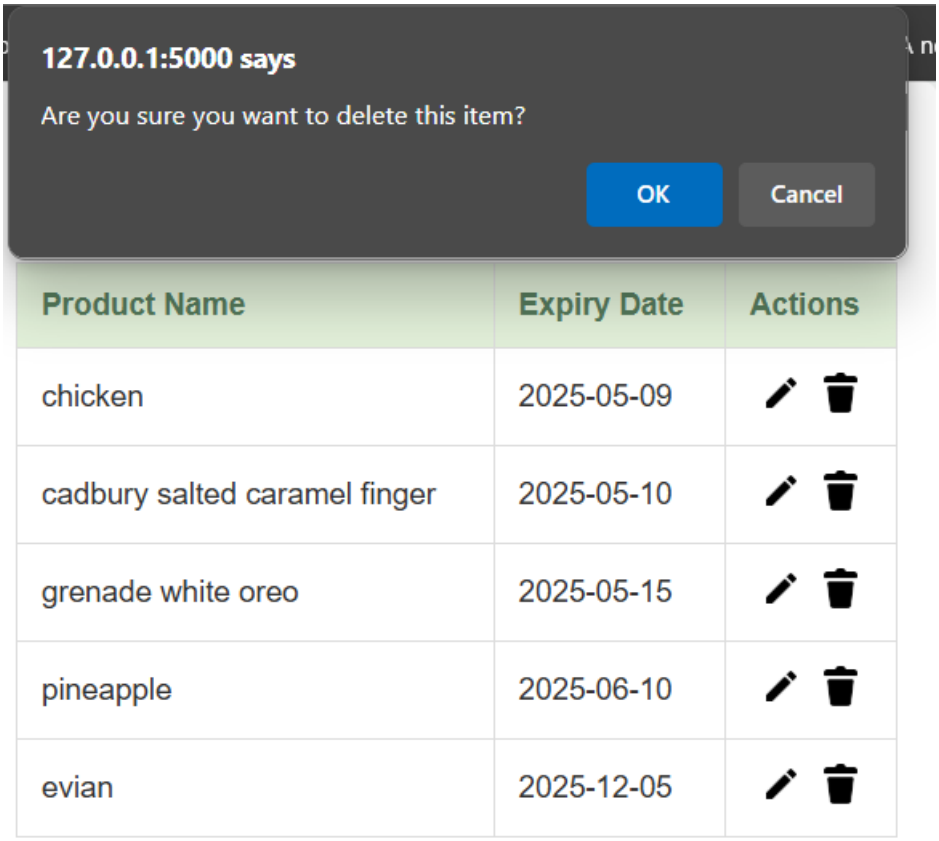


Figure 5.7 *edit and delete modal pop ups on web browser*

Each Item row in the pantry UI was created using jinja2 for loop, below is a code snippet

```
<h1>Pantry Inventory</h1>
  {% if pantry_contents %}
    <table>
      <tr>
        <th>Product Name</th>
        <th>Expiry Date</th>
        <th>Actions</th>
      </tr>
      {% for row in pantry_contents %}
{% endfor %}
```

Sprint 4 successfully completed the integration of the AI pipeline into a interactive usable prototype, validating the pipeline outlined in the methodology chapter. The next sprint was then exploring the robustness and feasibility of the scanning pipeline.

5.8 Sprint 5 results

Aim	Sprint 5– Technical testing
explore the robustness and feasibility of the receipt scanning pipeline on a broader range of real-world receipts.	Applying : -edge detection

Following the successful integration and demonstration of the AI-driven OCR receipt scanning system in sprint 4, sprint 5 focused on technical testing to assess robustness and generalizability across the receipts used in sprint one and to explore limitations and gather insights for future improvements.

5.8.1 OCR extraction on varied receipts

All receipts tested during sprint one were uploaded to the flask web app in the index page as described in chapter 4, it extended by uploading varied receipt tests. No additional preprocessing was done manually; the existing pipeline was used to process each receipt.

Upon uploading the other test receipts, no food items were successfully extracted, and no pantry updates were made. This shows the OCR module struggled to extract clean structured text from noisy receipt images, and therefore, fuzzy matching couldn't identify and match food items against the food database. This means that the pipeline was only functional and feasible for controlled, clean receipts it was not robust against common receipts without additional preprocessing enhancements therefore, the pipelines accuracy heavily depended on the receipt quality making it not fully feasible at this stage.

5.8.2 edge detection exploration

Although additional preprocessing enhancements were not integrated into the pipeline during Sprint 6, an experimental exploration of edge detection and receipt auto-cropping was tested through reusing code publicly available that can be seen in appendix B which aimed to detect receipt boundaries and apply four-point perspective transforms to straighten and crop receipts which were some issues observed during technical testing. Due to time constraints, edge detection was not incorporated into the testing environment therefore the functional testing results do not consider the auto-cropping enhancements however future work could focus on integrating such preprocessing steps to improve the robustness of this POC.

Edge detection example

Figure 5.8 *Edge detection applied*

Edge detection with Gaussian blur

Kernel size	5,5
Extracted text	1 Warburtons Danish Sliced Smal] £1.40

The experimentation demonstrated the effectiveness of applying edge detection and perspective transformation before applying OCR as it was able to almost successfully detect the food item . edge detection was tested which was able to almost successfully get the food items however these findings reinforce the importance of integrating adaptive preprocessing techniques such as auto-cropping , and other techniques mentioned in chapter 3 and 4. With such improvements, the pipeline could be more robust and usable however more testing would be required with the other receipts using edge detection to see how the quality of the receipt effects detecting the edges.

5.9 User testing

After technical testing, user evaluation on a few participants was conducted to determine how effective the prototype was and if it aligned with the real word behaviours and whether it could feasibly support users in reducing food waste. The findings highlight that majority of the users who used manual methods to track food items struggled with knowing the expiry and the key causes were forgetfulness and overbuying. The users found the interface clear, and easy to use with features such as seeing expiring soon notifications and editing capabilities. These results reinforce the project's idea that automated food tracking is more behaviourally compatible than manual input systems and shows positive reviews on the ai tool that could help them reduce waste and plan meals better with minimal additional effort. Refer to appendix M for the results.

CHAPTER 6 CONCLUSION

6.1 Objectives and research questions revisited

The primary objective of the project was to evaluate the feasibility of using AI-driven receipt scanning to automate food inventory tracking by extracting food items from receipts with expiry dates. Overall, the objectives were successfully achieved however with the refinements during development mentioned in section 1.2 to fulfil the goal. While the original goal included comparing multiple OCR tools, the project focused on Tesseract following a literature-based evaluation in chapter 3 and appendix E, which identified it as the most suitable option to use for the POC implementation. Similarly, although advanced NLP methods were considered , a simpler rule-based approach using regular expressions and fuzzy matching were used as it achieved strong performance on the controlled receipt as seen in the results chapter during sprint 2 in section 5.4, it was able to extract the food items successfully. Although extracting retailer names was initially planned, this was omitted during development to maintain a focused scope on extracting food items only to assess the feasibility. A matching technique was then used to link the extracted items to a manually populated food database with its expiry dates as described during sprint 3 and 4 in chapter 4. A flask-based web app was built to integrate all the layers to demonstrate the full pipeline to meet the objectives.

All research questions addressed in chapter 1 was addressed and a full summary is provided in Appendix O

These conclusions therefore build upon the research context discussed in chapter 3 where a gap was identified in applying automation to food inventory management. As the project developed a working prototype capable of extracting food items from a controlled receipt, matching them to a food database to retrieve its predicted expiry dates , fulfilling its main aims outlined in chapter 1 demonstrating the possibility of using automation compared to existing systems discussed in Chapter 3 which rely primarily on manual entry or barcode scanning so therefore this work advanced the field by demonstrating some feasibility of OCR-based automation for pantry management. In fulfilling these objectives and research questions it was able to provide practical insights on how this can be used for food tracking applications providing a foundation for future research and development.

6.2 Limitations and Reflections

Although the system achieved its core objectives, several limitations were observed during testing from the results shown in chapter 5, where Ocr performance was highly sensitive to the quality of receipt images where folds, shadows, poor lighting significantly reduced accuracy.

Additionally expiry dates in the food database were manually assigned ignoring brand and packaging or storage variations which could lead to inaccurate expiry predictions in read -world use.

A broader reflection of the concept of a system to be “good enough” is needed as occasional ocr errors or expiry date errors may be acceptable for personal use however may not be tolerable for commercial settings. Serious user testing would be needed to determine the acceptable error

rates as trust is critical and if users encounter too many errors or misleading expiry dates, the system risks losing credibility to therefore not being used and abandoned. Overall, whilst this prototype validates the technical approach it is not yet fully reliable for real world deployment as further research would be needed to address these limitations to ensure it is feasible.

Overall with the help of reusing material (refer to appendix B) it allowed the author to save time which was critical to the success of this proof of concept as reusing Tesseract OCR for text extraction saved a lot of time instead of developing a custom OCR which may be difficult in such short time. Following the online examples and documentations, they helped the author adapt regex patterns and applying image processing techniques to improve the OCR accuracy well instead of manually editing images, these libraries helped speed up the development of the prototype. Altogether all these materials enabled the author to focus on validating the core logic of the system rather than rebuilding components which would require more extensive research and time , therefore this made the project more efficient, allowing the project to be technically feasible with the control receipt within the timeframe.

6.3 Implications for Commercialization and Future Work

Although a functional POC was developed the system is not yet ready for commercialization as several significant areas of future work is necessary for this to be feasible. First as seen in chapter 5, preprocessing In sprint 5 where edge detection was tested could be fully integrated to better handle poor receipt conditions. Error corrections through post processing through dictionary-based methods could use machine learning models to further address OCR inaccuracies. The current flask web application discussed during section 5.6 would need to be adapted for mobile users to reflect real-word user behaviour and would also need to adapt to varied camera qualities and lighting conditions. User testing must also be done to understand the acceptable levels of error and the technical improvements. The most important area concerns handling expiry date accuracy since its currently static. Future systems would need dynamic expiry prediction through potentially training machine learning models on historical receipt datasets or integrating external product databases or supermarket APIS to correctly fetch the expiry dates. Without this improvement the risk of incorrect pantry entries would undermine user trust which would be a barrier in adopting this pipeline.

6.4 Legal, Social, Ethical and professional issues

In my PDD, the project was classified under category 1 meaning it posed no direct risk to third parties as the PoC was focused on demonstrating a feasible technical concept rather than deploying a usable product

The legal concerns were focused on GDPR compliance and the risk of processing sensitive data from receipts and this became clear that the OCR could pick up payment types therefore regex filtering was implemented as seen in chapter 4.5 to remove non-food items and personally identifiable information from the processed output. As I obtained my own receipt dataset and uses those receipts, if the system developed beyond a PoC, users data should be anonymized to comply with GDPR.

The social impact was to help reduce food waste so, as mentioned in the PDD by helping users manage expiry dates to reduce food waste however whilst fuzzy matching testing (see sprint 3

results in chapter 5) some methods added the wrong food item which therefore also gets the wrong expiry date, therefore to mitigate this risk, the system had a manual override form to update the expiry date and food item name if it was not matched correctly.

Therefore overall the development of this PoC, prevents harm through misleading automation, and respects privacy , and future work would need to secure user information and cover more datasets of receipts to upheld the BCS code of conduct.

6.5 Personal Reflections

This project has enabled the author to gain new technical and project management skills, providing hands-on experience with Python libraries such as Tesseract, OpenCV, RapidFuzz, and Flask, supporting development as an aspiring full-stack web developer. Integrating OCR technology with web development provided valuable insights into using AI to automate real-world processes, strengthening skills in both backend and frontend development.

Managing the project through an Agile-inspired sprint structure allowed the author to plan, adapt, and maintain steady progress even when challenges arose. Given the tight development schedule, the sprint-based approach ensured that each layer of the AI pipeline was developed, tested, and integrated systematically.

One challenge faced during the project was the variability and noise in real-world receipt images, which significantly impacted OCR performance. In response, the author focused initial development on a single controlled receipt with clean formatting to validate the core system components. While this approach enabled delivery of a working proof-of-concept within a short timeframe, it delayed exposure to the complexities of real-world data.

If the project were to be repeated, more time would be dedicated to developing advanced preprocessing methods, and real-world receipt testing would begin earlier to identify OCR challenges sooner. A more balanced approach between controlled validation and varied real-world testing would improve system robustness.

Overall, the project strengthened the author’s technical foundation, problem-solving ability, and understanding of how to adapt research-driven software development to real-world constraints.

Appendix A - PDD

PDD **CONTENTS**

Cover Page 93

2. Project Statement 94

3. Project Objectives 94

Sub-objective 1: Compare Multiple OCR Tools for Receipt Text Extraction 95

Sub-objective 2: Evaluate Different NLP Techniques for Processing Extracted Receipt Text . 95

Sub-objectives 3: Implement an OCR Tool to Extract Food Item and Retailer Data from Receipts 95

Sub-objectives 4: Preprocess Extracted Receipt Data into a Structured Format 95

Project sub-objectives 5: Develop a Method for Matching Extracted Food Items to a Food Database with Predicted Expiry Date 95

Research Questions 96

Project Beneficiaries 96

Project plan 97

Risks affecting the project 99

Legal, social, ethical and professional considerations 100

Research Ethics checklist 101

Cover Page

Full name:	Fatima Ahmed
Project title:	Exploring the Feasibility of AI-Driven Receipt Scanning for Food Tracking

Module name	Individual Project IN3007
Title of degree:	BSc (Hons) Computer Science
University e-mail:	Fatima.ahmed.3@city.ac.uk
Project supervisor:	Nicolas Hine
Who proposed the project:	myself
proprietary interests in your project work and/or outside help:	No further arrangements for proprietary interests in the current project work and/or outside help
Project description:	The purpose of this Proof of Concept (PoC) is to determine the technical feasibility of using AI to extract data from shopping receipts, match products to a food database for the predicted expiry dates.
Promises for project acceptance:	Submitting all deliverables in accordance with academic and ethical guidelines
Word count:	1700 (excluding the cover page/sheet,ethics

2. Project Statement

Food waste is a significant issue in the UK, with severe economic and environmental implications. According to the Waste and Resources Action Programme (WRAP), the UK produced approximately 9.5 million tonnes of food waste in 2018, with 70% of this waste intended to be consumed by people. The largest contributor to this problem is households, responsible for 6.6 million tonnes (70%) of the total food waste (Dray, 2021).

Many people purchase groceries without a structured system for monitoring what they have, leading to expired food and unnecessary waste. Whilst some existing food-tracking apps existed to attempt to solve the issue, they rely on manual input of data and barcode scanning with limited product details or incomplete expiration tracking, making them inconvenient and ineffective for long-term use as it can be tiring to scan each item one by one to update the inventory leading to no motivation in using the apps.

Therefore a major gap in current solutions is the lack of automation in logging food purchases. While shopping receipts already contain detailed lists of purchased items, most apps fail to utilize this data effectively. Existing apps like NoWaste and Nosh attempt to track inventory but face issues like inaccurate data handling (List,2025) which highlights a clear opportunity to improve food tracking by leveraging AI-driven receipt scanning to automate the process , reducing manual effort and increasing accuracy

OCR has been widely used to extract text from documents but achieving 100% accuracy remains a challenge due to issues like uneven lighting, font variations and such and receipt scanning may have additional challenges since they are often crumpled, with inconsistent fonts so text extraction will be less reliable without proper preprocessing(Avyodri et al.,2022). Brisinello et al (2021) conducted experiments on Tesseract OCR and demonstrated that preprocessing significantly improves OCR accuracy suggesting that preprocessing is essential.

By testing different OCR tools, NLP techniques, and text-matching methods, this PoC aims to evaluate accuracy, efficiency, and potential challenges in automating food inventory tracking with shopping receipts.

3. Project Objectives

The main objective of this project is to evaluate the **feasibility** of using **AI-powered techniques** to extract data from shopping receipts , and assign predicted expiry dates based on a food database.

To meet my goal the following sub-objectives of my project shall:

Sub-objective 1: Compare Multiple OCR Tools for Receipt Text Extraction

Test: Evaluate at least **two OCR tools** (e.g., **Tesseract, Google Vision API**) for receipt text extraction.

Measurement: Compare OCR performance based on **accuracy (correctly extracted text) and error rate** using a sample of receipts.

Sub-objective 2: Evaluate Different NLP Techniques for Processing Extracted Receipt Text

Test: Compare at least **two NLP techniques** for cleaning and structuring OCR-extracted text.

Measurement: Assess effectiveness based on accuracy in identifying food items and retailer names.

Sub-objectives 3: Implement an OCR Tool to Extract Food Item and Retailer Data from Receipts

Test: Implement an OCR tool (e.g., Tesseract, Google Vision API) and evaluate its ability to extract food names.

Measurement: OCR should achieve at least **80% accuracy** in correctly identifying **food items**.

Sub-objectives 4: Preprocess Extracted Receipt Data into a Structured Format

Test: Convert raw OCR text into a structured dataset containing food item names, retailer, and other relevant attributes.

Measurement: Verify that extracted data is correctly formatted and structured for further processing.

Project sub-objectives 5: Develop a Method for Matching Extracted Food Items to a Food Database with Predicted Expiry Date

Test: Implement **text matching techniques** to link extracted food names to a structured database containing predicted expiry dates.

Measurement: At least **70% of extracted food items** should be successfully matched to database entries.

Research Questions

Is it technically feasible to use AI to extract data from shopping receipts, match products to a food database, and assign predicted expiry dates?

To break this down, the following questions will be addressed:

1. How accurately can AI-powered OCR extract food item names and retailer details from receipts?
2. Which NLP technique is most effective for processing and structuring OCR-extracted receipt text?
3. What is the best method for matching extracted food items to a food database?
4. What are the key technical challenges in using receipt scanning for food inventory automation?

Project Beneficiaries

Users of Food Waste Reduction Apps

Many existing apps (e.g., NoWaste, Nosh) require users to manually enter food items or scan barcodes, which is time-consuming and discourages consistent usage.

By automating food inventory input through receipt scanning, this project aims to reduce user effort and improve engagement, making it more convenient to track food and reduce waste.

Food Waste Reduction and Smart Pantry Apps developers

Food-tracking apps could integrate AI-powered receipt scanning to enhance user experience and user engagement.

This PoC will provide insights into how OCR and NLP can be applied to automate food inventory tracking, making food waste reduction apps more efficient and user-friendly.

Retailers

If feasible, Retailers could integrate this technology into their mobile apps, allowing customers to automatically track purchase with expiry dates and food tracking insights could increase brand loyalty.

Project plan

Since this project explores the technical feasibility of AI-powered receipt scanning, I will follow an incremental development approach. By first focusing on a single retailer's receipt format, I can test and refine my approach before expanding to other variations. This ensures manageable complexity and allows for iterative improvements.

The work plan has two main phases: the experimentation phase ,report writing phase. I will be coding in Python, utilizing libraries like Tesseract OCR for receipt scanning, and apply data cleaning techniques to process the extracted data focusing on extracting food items, retailer names, and use basic string matching to link the extracted items to a food database for predicted expiry dates.

I will begin by testing OCR on a sample set of receipts, focusing on a single receipt type. After evaluating OCR performance, I will proceed with the data preprocessing steps, such as handling misrecognized data. Once the text is cleaned, I will link the extracted food items to the food database, retrieving predicted expiry dates.

In the report writing phase, I will document all findings, challenges, and results from the experimentation phase. This will include details on preprocessing steps, OCR accuracy, matching results, and any anomalies observed. I will also evaluate the limitations of the current PoC and suggest areas for future improvement.

The report will serve as a summary of the PoC, demonstrating the technical feasibility of using AI to extract data from shopping receipts, match products to a food database for the predicted expiry dates and providing a clear path for further development.

Refer to the Gantt Chart in **Figure 1 for detailed project timeline**

Figure 1 Gannt Chart

Risks affecting the project

Risk	Low	Medium	High	Risk	Mitigation
OCR may not accurately extract receipt data		x		If OCR fails to recognize text correctly, the entire process of automating food tracking would not work which may make it unfeasible	Test multiple OCR tools and compare their accuracy
NLP may struggle with food item variations		x		Extracted data may not match the food database so it could lead to incorrect or no expiry predictions	Test with rule based-matching and AI-based NLP to compare performance
Receipt formats may not have the same layout for all the retailers		x		If the proof of concept only works for one retail it may not generalize to others	I can focus on one retail first to prove feasibility before testing on multiple retails
Limited access to receipt images for testing		x		Without enough receipts the OCR evaluation may be unreliable	I can use public datasets or collect my own sample
Ethical concerns	x			If receipts contain card details it may	Avoid storing sensitive data by

regarding data privacy				raise privacy concerns	filtering only food names

Legal, social, ethical and professional considerations

My project falls under Category 1 as it will serve as a proof-of-concept which focuses on evaluating the feasibility of using AI to extract food item data from receipts and predict expiry dates, therefore it is not intended for public deployment but it still requires consideration of legal, ethical and professional responsibilities to ensure compliance with best practices

From a legal standpoint, the project involves getting data from receipt images which may contain sensitive or copyright content so using public datasets and removing personal information like payment details to comply with data protection laws such as GDPR would be required.

In terms of social impact, the Poc aims to support food waste reduction apps, helping users track expiry dates more effectively however it's accuracy will influence user behaviour so if it is developed further, the system has to say the expiry dates are estimates and not guarantees to not mislead users .

Ethically, there may be bias in the ai models where it can struggle with certain receipt formats, food brands or receipts that are not in English which could lead to inaccurate data extraction so may have to test on diverse receipt formats to identify the biases and so the system should then allow manual corrections to improve accuracy.

From a professional perspective, This project aligns with the British computer society (BCS) code of conduct ensuring that it does not pose harm to individuals or organizations, respecting privacy and data protection laws, it's a Poc reducing the risk of misleading results. By integrating these considerations into the project, this study ensures that legal, social, ethical, and professional responsibilities are met while producing a PoC that is both technically feasible and aligned with best practices. This approach minimizes risks associated with data privacy, bias, and accessibility, while also addressing the potential societal impact of automating food inventory and expiry date tracking

Research Ethics checklist

Research Ethics Review Form for BSc and MSci Projects

Computer Science Research Ethics Committee (CSREC)

http://www.city.ac.uk/department-computer-science/research-ethics

Undergraduate students undertaking their final project in the Department of Computer Science must consider the ethics of their project work and ensure that it complies with research ethics guidelines and the law for data protection. In some cases, a project will need approval from an ethics committee before it can proceed. Usually, but not always, this will be because the student is involving other people ("participants") in the project.

To ensure that they give appropriate consideration to ethical issues, all students must complete this form and attach it to their project definition document (PDD). There are two parts:

PART A: Ethics Checklist. All students must complete this part.
The checklist identifies whether the project requires ethical approval and, if so, where to apply for approval.

PART B: Ethics Proportionate Review Form. Students who have answered "no" to all questions in A1, A2 and A3 and "yes" to question 4 in A4 in the ethics checklist must complete part B as well. The project supervisor or consultant has delegated authority to provide approval in such cases that are considered to involve MINIMAL risk. The approval may be ***provisional*** – *identifying the planned work with human end user participants as* likely to involve MINIMAL RISK. In such cases you must additionally seek ***full approval*** from the supervisor or consultant as the project progresses and details are established. You must obtain ***full approval*** in writing, before recruiting and engaging with human end users participants for your project.

A.1 If you answer YES to any of the questions in this block, your consultant/supervisor must have obtained approval for the project from an appropriate external ethics committee, and you need to have received written confirmation of this from him/her. Students cannot themselves apply for ethics approval in this case as the project is considered high risk". This type of research is not covered by City's process, and external approval from an appropriate institution is required.		*Delete as appropriate*
1.1	Does your research require approval from the National Research Ethics Service (NRES)?	**NO**
1.2	Will you recruit participants who are covered by the Mental Capacity Act 2005?	**NO**
1.3	Will you recruit any participants who are covered by the Criminal Justice System, for example, people on remand, prisoners and those on probation?	**NO**
A.2 If you answer YES to any of the questions in this block your consultant/supervisor must have obtained appropriate ethics committee approval		*Delete as appropriate*
2.1	Does your research involve participants who are unable to give informed consent? *For example, people who may have a degree of learning disability or mental health problem, that means they are unable to make an informed decision on their own behalf.*	**NO**
2.2	Is there a risk that your research might lead to disclosures from participants concerning their involvement in illegal activities?	**NO**
2.3	Is there a risk that obscene and or illegal material may need to be accessed for your research study (including online content and other material)?	**NO**
2.4	Does your project involve participants disclosing information about protected characteristics (as identified by the Equality Act 2010)? *For example: racial or ethnic origin; political opinions; religious beliefs; trade union membership; physical or mental health; sexual life; criminal offences and*	**NO**

2.5	Does your research involve you travelling to another country outside of the UK, where the Foreign & Commonwealth Office has issued a travel warning that affects the area in which you will study? *Please check the latest guidance from the FCO - http://www.fco.gov.uk/en/*	**NO**
2.6	Does your research involve invasive or intrusive procedures? *These may include, but are not limited to, electrical stimulation, heat, cold or bruising.*	**NO**
2.7	Does your research involve animals?	**NO**
2.8	Does your research involve the administration of drugs, placebos or other substances to study participants?	**NO**
A.3 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee or the Senate Research Ethics Committee (SREC), you must apply for approval from the Computer Science Research Ethics Committee (CSREC) through Research Ethics Online - https://researchmanager.city.ac.uk/. Depending on the level of risk associated with your application, it may be referred to the Senate Research Ethics Committee (SREC).		*Delete as appropriate*
3.1	Does your research involve participants who are under the age of 18?	**NO**
3.2	Does your research involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)? *This includes adults with cognitive and / or learning disabilities, adults with physical disabilities and older people.*	**NO**
3.3	Are participants recruited because they are staff or students of City, University of London? *For example, students studying on a particular course or module.* *If yes, then approval is also required from the Head of Department or Programme Director.*	**NO**
3.4	Does your research involve intentional deception of participants?	**NO**
3.5	Does your research involve participants taking part without their informed consent?	**NO**
3.5	Is the risk posed to participants greater than that in normal working life?	**NO**
3.7	Is the risk posed to you, the researcher(s), greater than that in normal working life?	**NO**
A.4 If you answer YES to the following question and your answers to all other questions in sections A1, A2 and A3 are NO, then your project is deemed to be of MINIMAL RISK. **If this is the case, then you can apply for approval through your supervisor under PROPORTIONATE REVIEW. You do so by completing PART B of this form.** **If you have answered NO to all questions on this form, then your project does not require ethical approval. You should submit and retain this form as evidence of this.**		*Delete as appropriate*
4	Does your project involve human participants or their identifiable personal data? *For example, as interviewees, respondents to a survey or participants in testing.*	**YES**

PART B: Ethics Proportionate Review Form

If you answered YES to question 4 and NO to all other questions in sections A1, A2 and A3 in PART A of this form, then you may use PART B of this form to submit an application for a proportionate ethics review of your project. Your project supervisor has delegated authority to review and approve this application under proportionate review. You must receive final approval from your supervisor in writing before beginning the planned research.

However, if you cannot provide all the required attachments (see B.3) with your project proposal (e.g. because you have not yet written the consent forms, interview schedules etc), the approval from your supervisor will be ***provisional***. You **must** submit the missing items to your supervisor for approval prior to commencing these parts of your project. Once again, you must receive written confirmation from your supervisor that any provisional approval has been superseded by with ***full approval*** of the planned activity as detailed in the full documents. **Failure to follow this procedure and demonstrate that final approval has been achieved may result in you failing the project module and/or result in an academic misconduct investigation.**

Your supervisor may ask you to submit a full ethics application through Research Ethics Online, for instance if they are unable to approve your application, if the level of risks associated with your project change, or if you need an approval letter from the CSREC for an external organisation.

B.1 The following questions must be answered fully. All grey instructions must be removed.		*Delete as appropriate*
1.1.	Will you ensure that participants taking part in your project are fully informed about the purpose of the research?	**YES**
1.2	Will you ensure that participants taking part in your project are fully informed about the procedures affecting them or affecting any information collected about them, including information about how the data will be used, to whom it will be disclosed, and how long it will be kept?	**YES**
1.3	When people agree to participate in your project, will it be made clear to them that they may withdraw (i.e. not participate) at any time without any penalty?	**YES**
1.4	Will consent be obtained from the participants in your project? Consent from participants **MUST** be obtained if you plan to involve them in your	**YES**

1.5	Have you made arrangements to ensure that material and/or private information obtained from or about the participating individuals will remain confidential?	**YES**

B.2 If the answer to the following question (B2) is YES, you must provide details		*Delete as appropriate*
2	Will the research be conducted in the participant's home or other non-University location? *If **YES**, you must provide details of how your safety will be ensured.*	**NO**

B.3 Attachments **ALL of the following documents MUST be provided to supervisors if applicable.** **All must be considered prior to final approval by supervisors.** **A written record of final approval must be provided and retained.**	***YES***	***NO***	***Not Applicable***
Details on how safety will be assured in any non-University location, including risk assessment if required (see B2)			**x**
Details of arrangements to ensure that material and/or private information obtained from or about the participating individuals will remain confidential (see B1.5) *Any personal data must be acquired, stored and made accessible in ways that are GDPR compliant.*	**x**		
Full protocol for any workshops or interviews**		**x**	
Participant information sheet(s)**	**x**		
Consent form(s)**	**x**		
Questionnaire(s)** *sharing a Qualtrics survey with your supervisor is recommended.*	**x**		
Topic guide(s) for interviews and focus groups**	**x**		
Permission from external organisations or Head of Department** *e.g. for recruitment of participants*			**x**

Appendix B – Reused Summary

			Sprint 1- 5
Material/ tool	Type	Source	Purpose
Tesseract OCR	Open source engine	GitHub - tesseract-ocr/tesseract: Tesseract Open Source OCR Engine (main repository)	Used for downloading the open – source engine to read the image and extract text.

Tesseract ocr with open cv	Adapting and reusing code	Python OCR: Extract Text from Images Easily
		
		Python OpenCV Tutorial
			Reading image:
			
			Code reused following the tutorial :
			# Open the image file
			image = Image.open('sample.jpg')
			
			# Perform OCR
			text = pytesseract.image_to_string(image)
			
			print(text)
			
			to read ocr text
			
			open cv was used as well for image preprocessing with computer vision, to save images for seeing how they looked after each technique,

			Sprint 2
OpenCV - methods	library	Open-source documentation OpenCV: Image Filtering	Used for understanding the techniques and parameters of using OpenCv for blurring, gaussian blur, bilateral filter, median blur etc For example, For median blur: **ksize** aperture linear size; it must be odd and greater than 1, for example: 3, 5, 7
Tesseract ocr with open cv	Adapting and reusing code	Python OpenCV Tutorial	**Preprocessing code :** img = cv2.imread('D:/image-1.png', cv2.IMREAD_GRAYSCALE this was reused and adapted following this tutorial.

Preprocessing technique – Gaussian Blur	Reusing techniques and	Python OpenCV – Smoothing and Blurring |	# Reading the image
			image = cv2.imread('image.png')
			
			# Applying the filter
			gaussian = cv2.GaussianBlur(image, (3, 3), 0)

	adapting	GeeksforGeeks	# Applying the filter
			medianBlur = cv2.medianBlur(image, 9)
			
			# Applying the filter
			bilateral = cv2.bilateralFilter(image, 9, 75, 75)
			
			This was reused and adapted with different values to test the kernels to see how it affected the receipt image

Preprocessing technique – Adaptive Thresholding	Adapting code from tutorial	OpenCV: Image Thresholding	ret,thresh1 = cv.threshold(img,127,255,cv.THRESH_BINARY) ret,thresh2 = cv.threshold(img,127,255,cv.THRESH_BINARY_INV) ret,thresh3 = cv.threshold(img,127,255,cv.THRESH_TRUNC) ret,thresh4 = cv.threshold(img,127,255,cv.THRESH_TOZERO) ret,thresh5 = cv.threshold(img,127,255,cv.THRESH_TOZERO_INV This code was reused to be used as a preprocessing technique , adaptive thresholding, during sprint 2 of the project to improve the extraction of food items. For example: *# Apply simple thresholding (BINARY)* ret, thresh1 = cv.threshold(img, 127, 255, cv.THRESH_BINARY) this code was adapted to test different kernels to see how it affected the receipt image: th2 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_MEAN_C,\ cv.THRESH_BINARY,11,2) th3 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_GAUSSIAN_C,\ cv.THRESH_BINARY,11,2)
			Sprint 2 - regex

Regex pattern – price matching	Adapting code	Automatically OCR'ing Receipts and Scans - PyImageSearch	**Regex pattern:**
			# define a regular expression that will match line items that include
			# a price component
			pricePattern = r'([0-9]+\.[0-9]+)'
			
			This code was adapted to capture my food items whilst using the price pattern
			But reused during initial testing

		Python Tutorial: Regular Expressions with Python - 2021 Regular expression syntax cheat sheet - JavaScript | MDN	My pattern final for extracting food items: ***Pattern = re.compile(r'^[*]?(.*?)\s*£(\d+[.,]\d{2})$', re.IGNORECASE)*** The tutorial was used to create patterns and adapted for example: 1. **^** matches the beginning of a string. 2. **$** matches the end of a string. 3. **\b** matches a word boundary. 4. ***\s: matches any whitespace characters such as space and tab****: matches any whitespace characters such as space and tab* 5. **\d** matches any numeric digit 6. **x?** matches an optional **x** character (in other words, it matches an **x** zero or one times). 7. **\ For characters that are usually treated specially** For example, "*" is a special character that means 0 or more occurrences of the character should be matched. **So together [*]? Means Optional * character is matched but not captured since () captures groups** ?(.*?) Makes it non-greedy (stop early when rest of pattern matches) Since food item names have spaces and then a price \s*£ Matches any spaces if there is or isn't incase OCR readings read it together as with space or without space and then £ as prices always begin with a £ 8. Example pattern: re.compile(r'^(\d{3})-(\d{3})-(\d{4})$') (\d{3}) = any numeric digit matching exactly 3 digits.

This was readapted to consider for example £2.50 where it could also be £20.50 so d+ meaning one or more, followed by [.,] meaning it could be £2.50 or £2,50 depending on OCR reading, \d{2} to be following exactly 2 digits after

re.compile(), will be cached. In other words, it saves the resulting regular expression object for reuse which makes it efficient when the expression will be used several times in a single program.

For this pattern:

item_name = re.sub(r'\bi(\d+G)\b', r'1\1', item_name)

To substitute for example I with 1 or to simply delete by substituting ‘’ , I followed this tutorial example:

Python re.sub() - Replace using Regular Expression

The syntax of re.sub() function is
re.sub(pattern, repl, string, count=0, flags=0)

the documentation said :

Parameter	Description
pattern	[Mandatory] The pattern which has to be found in the string.
repl	[Mandatory] The value which has to be replaced in the string in place of matched pattern.
string	[Mandatory] The string in which the replacement has to be done.
count	[Optional] The maximum number of pattern occurrences to be replaced.
flags	[Optional] Optional flags like re.IGNORECASE, etc.

So following the syntax sheet:
\n
Backreference: Where "n" is a positive integer. Matches the same substring matched by the nth capturing group in the regular expression (counting left parentheses,

I adapted this where I captured mine in group 1 and replaced it simply with a 1 where in r’1’\1, 1 is the literal and \1 is the group 1 capturing group.

		Python – Substituting patterns in text using regex | GeeksforGeeks	Deleting character code:
			re.sub(r'\b\d+\s*[a-zA-Z]+\b', '', *text*)
			
			To know how to delete characters so I can delete measurements I followed this tutorial after making my regex pattern to replace:
			
			# replacing every non-word character with a white space
			S[i] = re.sub(r"\W", " ", S[i])
			
			# replacing every digit character with a white space
			S[i] = re.sub(r"\d", " ", S[i])
			
			# replacing one or more white space with a single white space
			S[i] = re.sub(r"\s+", " ", S[i])
			
			# replacing alphabetic characters which have one or more
			# white space before and after them with a white space
			S[i] = re.sub(r"\s+[a-z]\s+", " ", S[i], flags = re.I)
			
			# substituting one or more white space which is at
			# beginning of the string with an empty string
			S[i] = re.sub(r"^\s+", "", S[i])
			
			# substituting one or more white space which is at
			# end of the string with an empty string
			S[i] = re.sub(r"\s+$", "", S[i])
			
			From this tutorial it shows that no matter what the regex pattern is to replace empty speech marks or ‘ ‘ can be used to delete strings and replaced with white space

			Sprint 3

Fuzzy matching	Adapting code following library	Fuzzy String Matching – A Hands-on Guide Fuzzy string matching in Python (with examples) | Typesense	Example from guide using the ratio method from the library to understand similarity with one string : Str1 = "Back" Str2 = "Book" Ratio = fuzz.ratio(Str1.lower(),Str2.lower()) print(Ratio) To understand many items and comparing to test this method in my code I adapted this by: my dictionary based food database and I went through with each of my extracted item so I to compare many items I used this example best_match = process.extractOne('joe r biden', str_list, scorer=fuzz.token_sort_ratio) print(best_match) this example shows to extract the best match to return one you use process.extractone() and then comparing with your list then the scorer so for my code: *for* fooditem *in* lines: match = process.extractOne(fooditem, food_database.keys(), *scorer*=fuzz.token_sort_ratio) using the process.extractOne() I pass the food item which is the ocr extracted item with the food_Database.keys() where in the dictionary the food items are stored and then the scorer following this strategy, I applied it to all the other scorers in the library to compare and see the best method for this prototype

			Sprint 4
Flask web app	Using guide To adapt Tutorial Parts reused	Welcome to Flask — Flask Documentation (3.1.x) How to Build a Flask Python Web Application from Scratch | DigitalOcean Getting started with Jinja Template | GeeksforGeeks	Using the documentation and the tutorial to set up flask from flask import Flask app = Flask(__name__) @app.route('/') def hello(): return 'Hello, World!' and then learning how to render html pages return render_template('index.html') and learning and adapting with the jinja template used from the tutorial: • {{ }} for expressions. • {# #} for comments (even multiline) inside the template. • {% %} for jinja statements (like loops, etc.) For example {{}} was used to render Pantry items in total: Total Items in Pantry: {{ total_items }} Where total items was the total number of items total_items = len(pantry_contents) {% %} Was used to show all the pantry contents for example

Forms using javascript	Tutorial Adapted and reused	How To Make a Modal Box With CSS and JavaScript
		
		Using W3schools for documentation of modals
			// Get the modal
			var modal = document.getElementById("myModal");
			
			// Get the button that opens the modal
			var btn = document.getElementById("myBtn");
			
			// Get the element that closes the modal
			var span = document.getElementsByClassName("close")[0];
			
			// When the user clicks on the button, open the modal
			btn.onclick = function() {
			  modal.style.display = "block";
			}
			
			// When the user clicks on (x), close the modal
			span.onclick = function() {
			  modal.style.display = "none";
			}
			
			// When the user clicks anywhere outside of the modal, close it
			window.onclick = function(event) {
			  if (event.target == modal) {
			   modal.style.display = "none";
			  }
			}
			
			Following this tutorial I adapted it to align with my form requirements
			
			var modal = document.getElementById("manualEntryModal");
			  var btn = document.getElementById("manualEntryBtn");
			
			  btn.onclick = function () {
			   modal.style.display = "block";
			  }
			
			  function closeModal() {
			   modal.style.display = "none";
			  }
			
			  window.onclick = function (*event*) {
			   if (*event*.target == modal) {
			    closeModal();
			   }

SQL with python	Reusing and adapting	Python SQLite3 Cursor Guide: Execute SQL Commands	The Author has experience with SQL in general with the commands however had to follow the documentation to connect with python code so the pantry table and food database is connected.
			
			import sqlite3
			
			# Create a database connection
			conn = sqlite3.connect('example.db')
			
			# Create a cursor object
			cursor = conn.cursor()
			
			# Create a table
			cursor.execute('''CREATE TABLE IF NOT EXISTS employees
			(id INTEGER PRIMARY KEY,
			name TEXT NOT NULL,
			salary REAL)''')
			
			# Commit the changes
			conn.commit()
			
			so this was reused but adapted to follow what the database was called.
			
			cursor.execute("SELECT * FROM employees WHERE id = 1")
			row = cursor.fetchone()
			print(f"Name: {row['name']}, Salary: {row['salary']}")
			
			This structure was used and adapted to return pantry data and such as the expiry dates

Preprocessing technique to detect receipt and extract information	Reusing and adapting code	Automatically OCR'ing Receipts and Scans - PyImageSearch	Debug command line arguments reused
			# construct the argument parser and parse the arguments
			ap = argparse.ArgumentParser()
			ap.add_argument("-i", "--image", required=True,
			help="path to input receipt image")
			ap.add_argument("-d", "--debug", type=int, default=-1,
			help="whether or not we are visualizing each step of the pipeline")
			args = vars(ap.parse_args())
			
			reused code to detect shape of receipt
			# load the input image from disk, resize it, and compute the ratio
			# of the *new* width to the *old* width
			orig = cv2.imread(args["image"])
			image = orig.copy()
			image = imutils.resize(image, width=500)
			ratio = orig.shape[1] / float(image.shape[1])
			
			# convert the image to grayscale, blur it slightly, and then apply
			# edge detection
			gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
			blurred = cv2.GaussianBlur(gray, (5, 5,), 0)
			edged = cv2.Canny(blurred, 75, 200)
			# check to see if we should show the output of our edge detection
			# procedure
			if args["debug"] > 0:
			cv2.imshow("Input", image)
			cv2.imshow("Edged", edged)
			cv2.waitKey(0)
			
			# find contours in the edge map and sort them by size in descending
			# order
			cnts = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,
			cv2.CHAIN_APPROX_SIMPLE)
			cnts = imutils.grab_contours(cnts)
			cnts = sorted(cnts, key=cv2.contourArea, reverse=True)
			
			# initialize a contour that corresponds to the receipt outline
			receiptCnt = None
			# loop over the contours
			for c in cnts:
			# approximate the contour
			peri = cv2.arcLength(c, True)
			approx = cv2.approxPolyDP(c, 0.02 * peri, True)
			# if our approximated contour has four points, then we can
			# assume we have found the outline of the receipt
			if len(approx) == 4:
			receiptCnt = approx
			break
			# if the receipt contour is empty then our script could not find the
			# outline and we should be notified

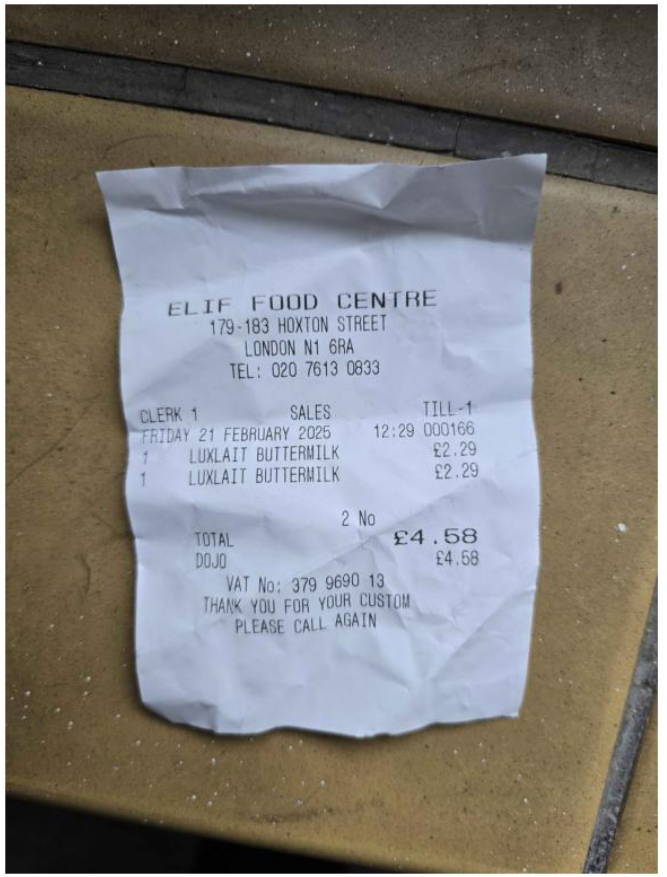
			if receiptCnt is None:
			raise Exception(("Could not find receipt outline. "
			"Try debugging your edge detection and contour steps."))
			
			# check to see if we should draw the contour of the receipt on the
			# image and then display it to our screen
			if args["debug"] > 0:
			output = image.copy()
			cv2.drawContours(output, [receiptCnt], -1, (0, 255, 0), 2)
			cv2.imshow("Receipt Outline", output)
			cv2.waitKey(0)
			# apply a four-point perspective transform to the *original* image to
			# obtain a top-down bird's-eye view of the receipt
			receipt = four_point_transform(orig, receiptCnt.reshape(4, 2) * ratio)
			# show transformed image
			cv2.imshow("Receipt Transform", imutils.resize(receipt, width=500))
			cv2.waitKey(0)
			
			This code was reused as a preprocessing method on automatically detecting edges to then apply ocr whilst exploring methods to test as a additional preprocessing method.

			Plots for results
Diagram drawings	Adapting and reusing	Matplotlib.axes.Axes.bar() in Python | GeeksforGeeks	Following this tutorial: **import** matplotlib.pyplot as plt **import** numpy as np data **=** ((30, 1000), (10, 28), (100, 30), (500, 800), (50, 10)) dim **=** len(data[0]) w **=** 0.6 dimw **=** w **/** dim fig, ax **=** plt.subplots() x **=** np.arange(len(data)) **for** i **in** range(len(data[0])): y **=** [d[i] **for** d **in** data] b **=** ax.bar(x **+** i ***** dimw, y, dimw, bottom **=** 0.001)

			ax.set_xticks(x + dimw / 2) ax.set_xticklabels(map(str, x)) ax.set_yscale('log') ax.set_xlabel('x') ax.set_ylabel('y') ax.set_title('matplotlib.axes.Axes.bar Example') I adapted it for my code to display my results for example: kernel_sizes = ['19,7', '13,6', '11,5'] wer_adaptive = [0.7857, 1.071, 1.143] cer_adaptive = [0.198, 0.3465, 0.3069] wer_bilateral = [0.5, 0.7143, 0.5714] cer_bilateral = [0.1782, 0.1782, 0.1386] fig, ax = plt.subplots(*figsize*=(10,6)) *# Bar width* bar_width = 0.35 index = np.arange(len(kernel_sizes)) bar1 = ax.bar(index - bar_width/2, wer_adaptive, bar_width, *label*='WER (Adaptive)', *color*='blue') bar2 = ax.bar(index + bar_width/2, cer_adaptive, bar_width, *label*='CER (Adaptive)', *color*='green') bar3 = ax.bar(index - bar_width/2, wer_bilateral, bar_width, *label*='WER (Bilateral)', *color*='orange') bar4 = ax.bar(index + bar_width/2, cer_bilateral, bar_width, *label*='CER (Bilateral)', *color*='red') *# Customize the plot* ax.set_xlabel('Kernel Sizes') ax.set_ylabel('Error Rates') ax.set_title('WER and CER for Adaptive Thresholding vs Adaptive Thresholding with Bilateral Filter') ax.set_xticks(index) ax.set_xticklabels(kernel_sizes) ax.legend() *# Display the plot* plt.tight_layout() plt.show() this example I used my own data and ocr accuracy values and used my own bar width size to make the image bigger.

Appendix C – Receipt Dataset

Receipts Collected



TESCO

South Plaza Isle Of Dogs Express
Any questions please visit
www.tesco.com/store-locator
VAT Number: GB 220 4302 31

1 Warburtons Danish Sliced Small White Bread 400g £1.40

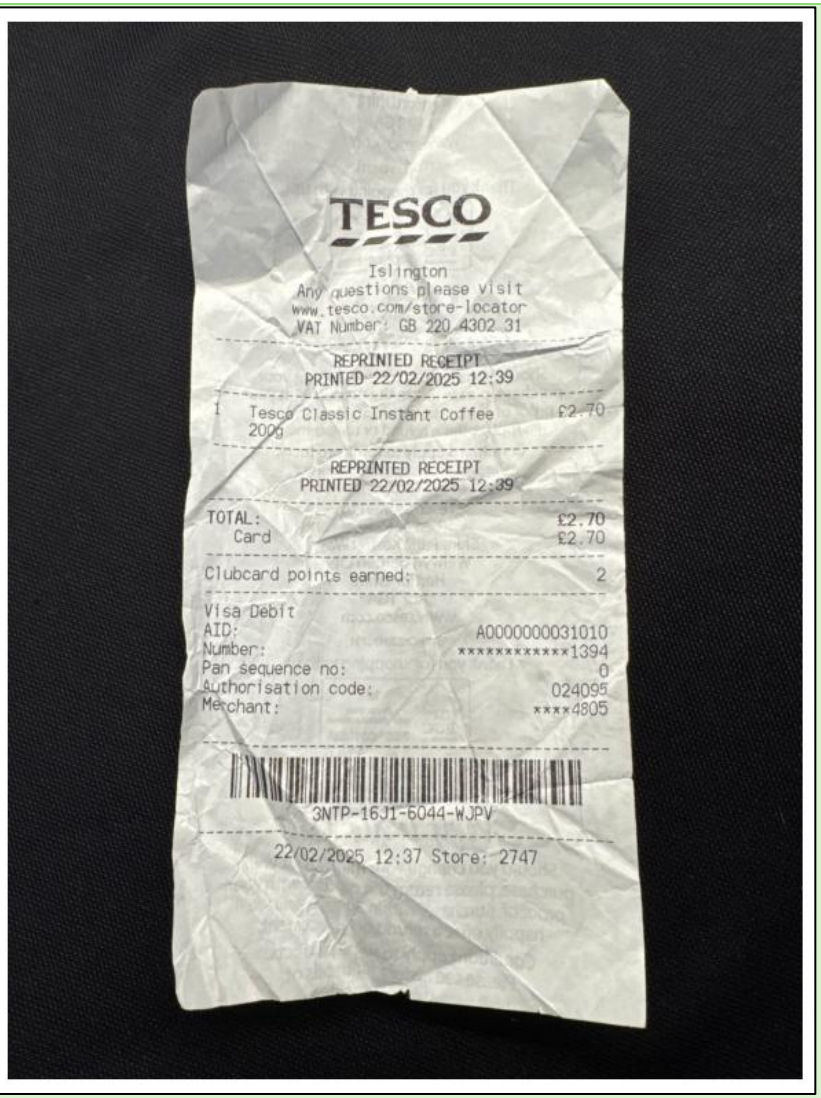
TOTAL: £1.40
Card £1.40

Clubcard points earned: 1
Clubcard points balance: 227

Visa Debit
AID: A0000000031010
Number: **********7018
Pan sequence no: 0
Authorisation code: 023806
Merchant: ****4805

6AM3-1JEZ-3044-Z68X

23/02/2025 14:48 Store: 6289 Checkout: 208



TESCO

Islington
Any questions please visit
www.tesco.com/store-locator
VAT Number: GB 220 4302 31

REPRINTED RECEIPT
PRINTED 22/02/2025 12:39

1 Tesco Classic Instant Coffee 200g £2.70

REPRINTED RECEIPT
PRINTED 22/02/2025 12:39

TOTAL: £2.70
Card £2.70

Clubcard points earned: 2

Visa Debit
AID: A0000000031010
Number: **********1394
Pan sequence no: 0
Authorisation code: 024095
Merchant: ****4805

3NTP-16J1-6044-WJPV

22/02/2025 12:37 Store: 2747

Appendix D – Ai pipeline Sprints folder and interface

To run interface folder

Installation

1. Set up a virtual environment in Windows PowerShell,
   run the following commands:

```
python -m venv env
.\env\Scripts\Activate.ps1
```

if policy error occurs:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

2. Install required packages:

```
pip install -r requirements.txt
```

Usage

- must be in the interface directory to run test.py and app.py
  Run the script with the following command:

```
cd interface
flask run --host=0.0.0.0
```

there will be 2 addresses: first will be the web version, second is the ip address you put in your mobile browser to use as a mobile app

To test receipt images only use the image light2.jpg found in receipt folder

To run Sprints folder

- sprint1.ipynb tests for basic ocr text extraction
- sprint2.ipynb preprocessing images
- sprint2regex.ipynb regex patterns
- sprint3.ipynb fuzzy matching food names
- receiptscan.py to test the edge detection code

to test receipt scan

cd Sprint – if project just opened

to run jupyter notebook code just press run on code sections

run command:

python receiptscan.py --image "img/receipt4.jpg" --debug 1

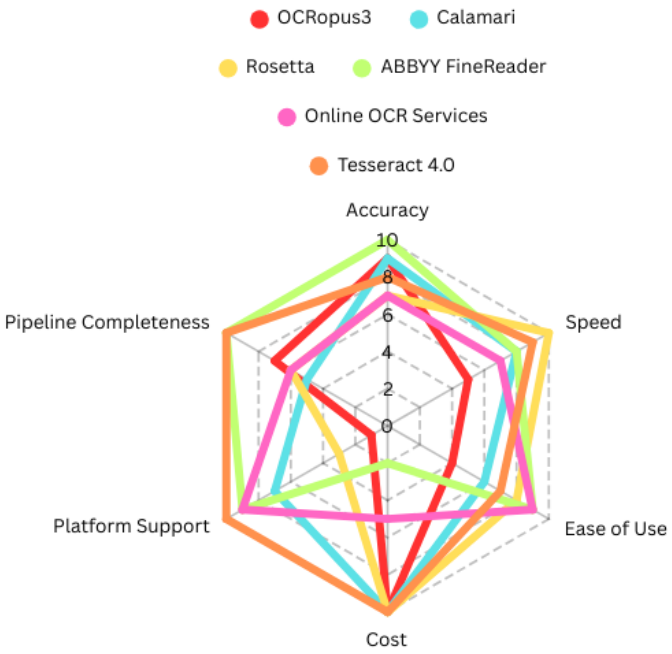
Appendix E – comparison of OCR Tools

OCR Tools Overview

With the growing demand for automating data entry numerous oct tools have been developed and evaluated over time which vary in terms of accuracy, processing speed, platform compatibility and cost . The table presents an overview of selected tools focusing on their core strengths, limitations and ideal cases where the evaluation was conducted by (Jain, Taneja and Taneja, 2021) on popular ocr software's.

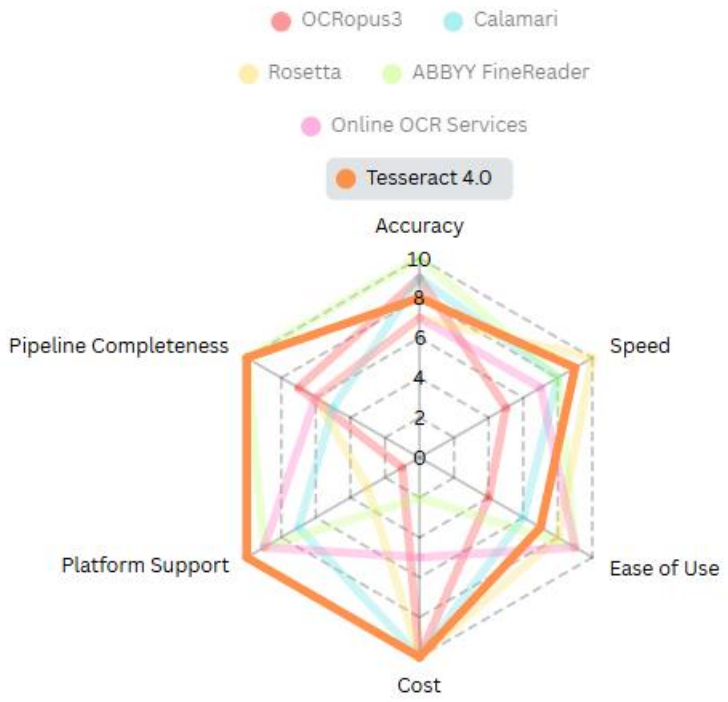
OCR Tool	Strengths	Limitations	Ideal case
ABBYY FineReader	Near 100% word-level accuracy on high-quality modern prints	Paid software Not suitable for early printed or distorted documents	High quality modern printed documents in business or enterprise settings

	Supports 150+ languages		
OCRopus3	High accuracy on early printed books Free	Not available on windows Only linux needs a set of commands in sequence for complete OCR rather than a single command so is a slower tool	Digitizing early printed books or historical documents
Tesseract 4.0	Full OCR pipeline LSTM-based recognition engine Supports multiple languages Free	Slightly less accurate than calamari for line-level text No GPU support	General- purpose OCR for documents across all major operating systems
Calamari	Better line-level recognition (lowest CER at 0.114%) Supports GPU training free	Not a full OCR pipeline No dictionary	High-speed recognition for modern text-line documents in OCR pipelines
Rosetta	Scalable for web-scale image recognition Character-level recognition for multilingual + symbols	Slight drop in accuracy vs LSTM-based models No dictionary support	Real-time OCR on social media images, multilingual image content/
Online OCR Services (e.g, Google Docs OCR, OnlineOCR)	Easy to use Convenient for casual users Often free	Privacy risks File size and usage limitations May fail on complex documents	Lightweight Occasional OCR tasks for personal



The radar chart includes the following dimensions (scored from 1 to 10, where 10 is very good, from the evaluation table):

- **Accuracy**: How well the OCR tool recognizes text.
- **Speed**: Processing speed of the tool.
- **Ease of Use**: User-friendliness and setup.
- **Cost**: 10 = free, lower values = more expensive.
- **Platform Support**: Compatibility across operating systems, especially for a phone app.
- **Pipeline Completeness**: Whether the tool supports the full OCR process from image to structured text.



Based on the radar chart analysis, Tesseract has been chosen as the most suitable OCR tool for the proof-of-concept project as it offers a well-balanced combination of features including high accuracy, cross-platform support making it compatible with major operating systems including mobile, open source and free to use ensuring cost-effectiveness. Whilst some tools demonstrate a higher accuracy or speed, tesseract has a more comprehensive and scalable solution ideal for general-purpose ocr tasks such as receipt scanning, its strong performance across multiple evaluation metrics and zero cost make it the most practical and efficient choice for this project.

Appendix F – Updated Objectives and Research questions

1. **Compare Multiple OCR Tools for Receipt Text Extraction Test:**
   Evaluate at least two OCR tools (e.g., Tesseract, Google Vision API) for receipt text extraction. Measurement: Compare OCR performance based on accuracy (correctly extracted text) and error rate using a sample of receipts.

   Changes: due to time constraints the implementation focused on **Tesseract** which was determined by the literature review of alternative tools refer to appendix

2. **Evaluate Different NLP Techniques for Processing Extracted Receipt Text Test:**

Compare at least two NLP techniques for cleaning and structuring OCR-extracted text. Measurement: Assess effectiveness based on accuracy in identifying food items and retailer names.

Changes: Although advanced NLP techniques were initially considered, the project used a rule-based approach (regular expressions and fuzzy matching) as it provided good results for the proof-of-concept

3. **Implement an OCR Tool to Extract Food Item and Retailer Data from Receipts Test:** Implement an OCR tool and evaluate its ability to extract food names.

4. **Preprocess Extracted Receipt Data into a Structured Format Test**: Convert raw OCR text into a structured dataset containing food item names, retailer, and other relevant attributes.

   Changes: Although the initial plan included extracting additional attributes such as the retailer name, this was omitted for the proof-of-concept, as the focus was solely on reliably obtaining the food item names

5. **Develop a Method for Matching Extracted Food Items to a Food Database with Predicted Expiry Date Test:**
   Implement text matching technique to link extracted food names to a structured database containing predicted expiry dates**.**

6. **Sub-objective 6: Develop a User-Friendly Web Application to Demonstrate the System**
   Originally not included in the PDD, this sub-objective was later added to provide a practical demonstration of the AI receipt scanning system. The web application was built using Flask to allow users to upload receipt images, automatically update a pantry database with matched food items and their expiry dates, and display the data on a dashboard.

Research Questions

The following research questions guided this project and some changes were made to the original set of questions (refer to appendix A for old set) due to changing the sub objectives

1. **How accurately can Tesseract extract food item names from shopping receipts?**

   Which investigated the reliability of OCR in converting receipt images to text which formed the foundation of the system

2. **How effective is a rule-based text cleaning approach using regular expressions in extracting food item names from the OCR output**

   Which determined whether simple, heuristic based techniques are enough to filter and structure OCR data, eliminating the need for complex NLP methods

3. **What is the best method for matching extracted food items to a food database?**
4. **What are the key technical challenges in using receipt scanning for food inventory automation?**

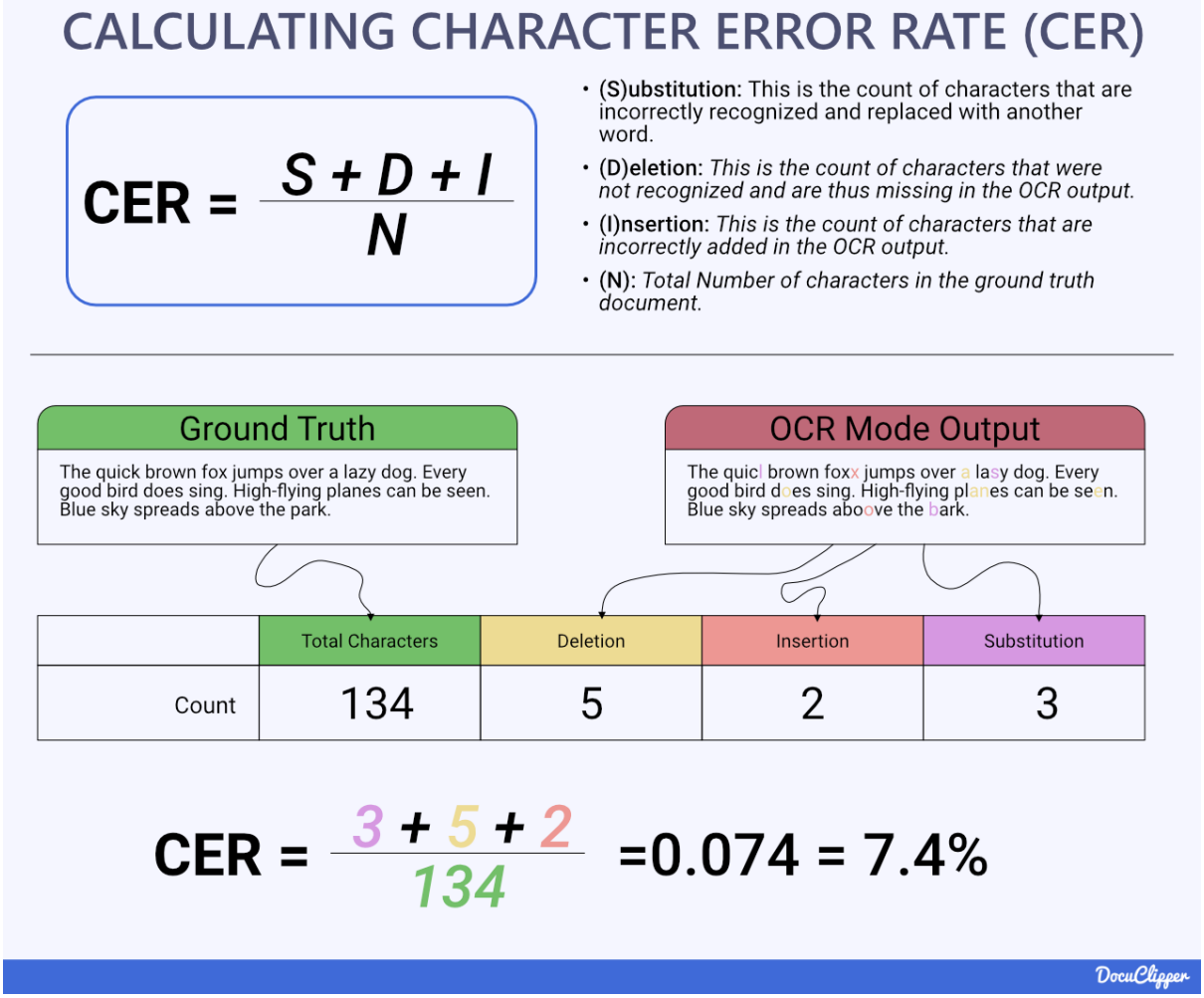
Which evaluated issues related to image preprocessing, inconsistent receipt formats.

Appendix G – Calculating OCR accuracy

Based on the scope of this project , the author chose to focus the calculation of the accuracy on the food items and the price (for regex patterns)in the receipt instead of the entire receipt due to being the core functionality of the system despite it not giving the system's overall performance the food items are more relevant to food tracking to avoid non-relevant details.

CER: Character Error Rate

Character error rate is an accuracy measurement which measures the ratio of total character level mistakes in the OCR text compared to the total characters In the ground truth text across the whole set

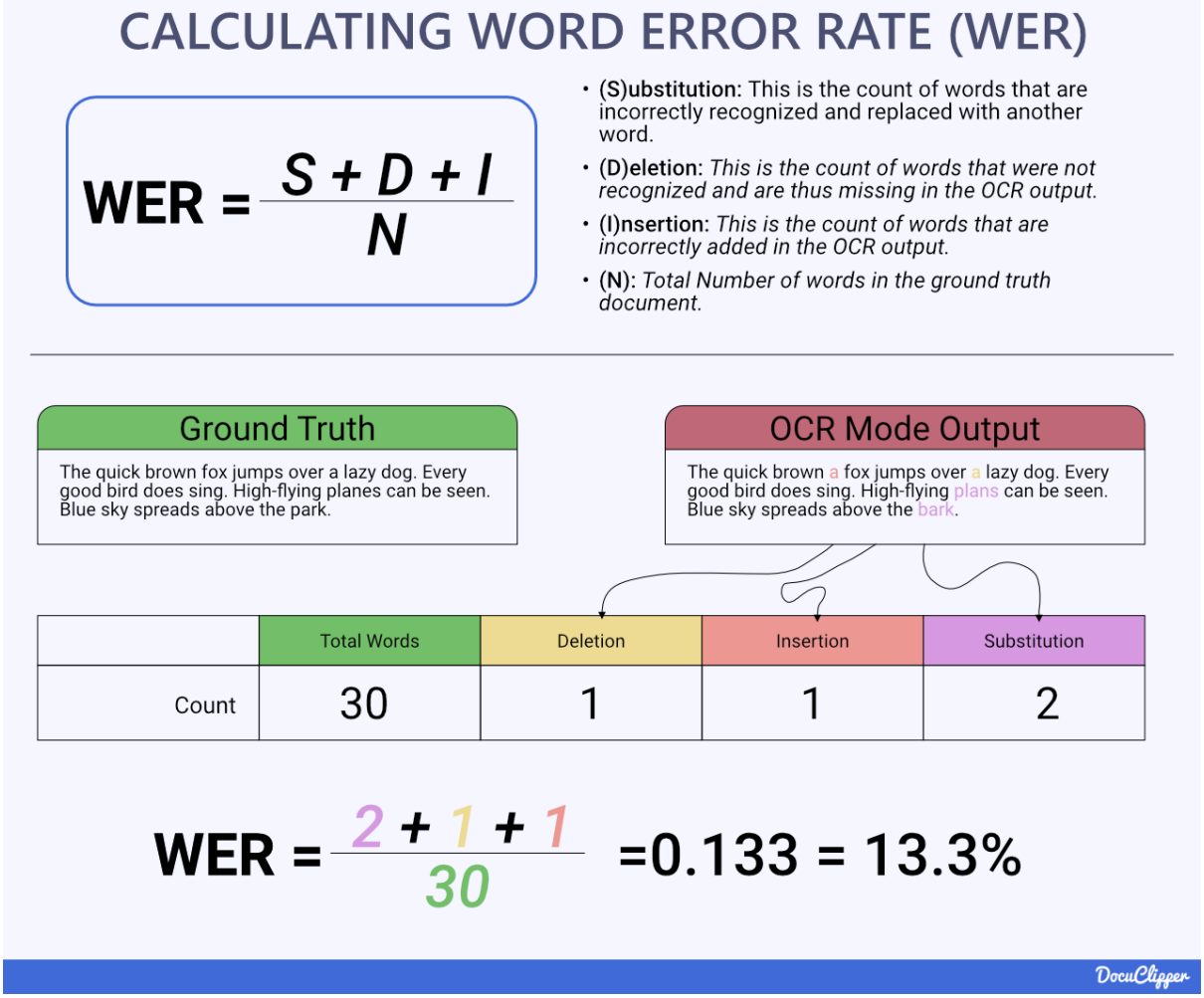


(www.docuclipper.com, 2024)

The lower the CER, the better : 0% being perfect OCR conversion accuracy

WER: Word Error Rate

WER measures the percentage of whole words that contain one or more inaccurate characters when compared to the ground truth



(www.docuclipper.com, 2024)

Lower WER indicates higher accuracy with 0% being perfect

Appendix H – Key Deviations summary

	Reason for Deviation
Receipt Testing Approach	Initially plan was to test OCR using multiple receipts on different conditions however due to time constraints and the need to establishing a working proof of concept it was

	decided to use one controlled receipt for initial testing to first validate the core functionality of the system and then carrying out the other receipts for technical testing in sprint 5 to assess robustness.
NLP Techniques	The Initial plan included implementing advanced NLP techniques however the decision to focus on simpler methods such as regex-based filtering and fuzzy matching was made to simplify development and provided good results for proof of concept. Advanced NLP techniques would require more time and resources to implement and test so to ensure the system can be validated ,regex filtering was used to validate the system before moving on to complex techniques.
Lack of Initial Interface plan	The initial project plan did not include the development of a user interface however to demonstrate the system's core functionality and to facilitate testing it because clear that a basic interface was necessary. Due to time constraints the author chose to build a prototype interface using HTML,CSS and javascript instead of developing a mobile app because of limited time and expertise in mobile app development, this approach allowed rapid prototyping. **Future consideration:** although it was web based, it can be developed to a mobile application if more time was available however the web app served as a prototype to demonstrate the system's capabilities and usability where flask also supports mobile browser but not all laptops were able to use the mobile ip for example my laptop did not allow me to run it on my phone however another laptop could run the web app on mobile browser as well.
Receipt Upload instead of scanning	Initially the plan was to implement a feature that allowed users to scan receipts for the ai driven process however given the constraints of developing a web app and the time limitations . it was not feasible to integrate scanning capabilities directly within the web app as scanning technology would typically require access to device cameras and specialized software which is better suited for mobile applications so to simplify the development process and ensure the core functionality of the receipt scanning pipeline

could be tested receipts were uploaded instead as pictures.

Appendix I – Web development With Flask

.

Flask is a simple web framework for python where it works with another library known as jinja 2 that allows dynamic rendering in the application making It easy to work with variables (GeeksforGeeks, 2023). The routing system helps mapping URL to python functions making it easy to set up, and control the application as well as being easy to test since the server is lightweight making this ideal to use for this feasible project during development. It is also the most popular tool to handle relational databases such as SQLite making it an ideal framework for this project.

Flask was used in this following way:

Web app with flask

Flask feature	**Application in project**
@app.route()	Declared for all endpoints including / , /pantry, /edit/<id> and ,/delete/<id>
File handling	To upload receipts
Request.files	Used to process the uploaded receipt images
Render_template()	Used to render dynamic HTML pages like index.html and pantry.html
Redirect() url_for()	Used for redirecting users after editing or deleting pantry items

SQLite integration	Handled using sqlite3 module and helper functions like get_db_connection()
Templating(Jinja2)	Displayed pantry entries, expiry dates, notifications dynamically

Flask HTTP methods can define how the client and servers communicate which allows browsers or apps to send requests and receive requests (GeeksforGeeks, 2022)

HTTP methods

method	**Used for**
GET	To request data from server
POST	To submit data to be processed to the server

Each route interacts with the backend logic using HTTP methods where in the project, that has preprocessing, fuzzy matching and SQLite functions returning dynamically rendered templates.

Flask Routing		
Route	**HTTP method**	**Purpose**
/	GET, POST	For index page to upload, display total items and expiry alerts
/pantry	GET	Lists all the items in the pantry
/edit/<id>	POST	Editing the item to update food name or expiry date
/delete/<id>	POST	Deleting item from the inventory

Jinja2 allowed embedding logics such as variables, loops and conditions within the HTML files and was used through flasks built in render_template() function to dynamically insert python data into the HTML templates. Below is the default syntax (Palletsprojects.com, 2024).

- `{% ... %}` for Statements
- `{{ ... }}` for Expressions to print to the template output
- `{# ... #}` for Comments not included in the template output

This syntax was used to dynamically render the dynamic HTML pages where users could view pantry items and expiry dates, get visual notifications for food items expiring soon, edit or delete entries.

Appendix J – Regex Filtering

RegEx is a sequence of characters that forms a search pattern which can be used to see if a string contains a specified pattern (W3Schools, 2019). Using the re module in python, has a set of functions that can be used. It can be used to search, replace and parse text with complex patterns of characters (Bogotobogo.com, 2021).

Function	Description
findall	Returns a list containing all matches
search	Returns a Match object if there is a match anywhere in the string
split	Returns a list where the string has been split at each match
sub	Replaces one or many matches with a string

Re.compile() returns a pattern object.

1. **^** matches the beginning of a string.
2. **$** matches the end of a string.
3. **\b** matches a word boundary.
4. **\d** matches any numeric digit.
5. **\D** matches any non-numeric character.
6. **(x|y|z)** matches exactly one of **x**, **y** or **z**.
7. **(x)** in general is a remembered group. We can get the value of what matched by using the **groups()** method of the object returned by **re.search**.
8. **x?** matches an optional **x** character (in other words, it matches an **x** zero or one times).
9. **x*** matches **x** zero or more times.
10. **x+** matches **x** one or more times.
11. **x{m,n}** matches an **x** character at least **m** times, but not more than **n** times.
12. **?:** matches an expression but do not capture it. Non capturing group.
13. **?=** matches a suffix but exclude it from capture. Positive look ahead.
    a(?=b) will match the "a" in "ab", but not the "a" in "ac"
    In other words, a(?=b) matches the "a" which is followed by the string 'b', without consuming what follows the a.

Following some common patterns (Bogotobogo.com, 2021)., this helped the author create a regex pattern to successfully capture a pattern.


```
pattern = re.compile(r'^[\*]?(.*?)\s*£(\d+[.,]\d{2})$',re.IGNORECASE\
```

matches beginning of string

optionally match a * as some have a * with food item name and some don't

Captures any characters stopping at the price to get it one at a time

. matches any character (OpMt, 2012)

Optional space between food item name and price

Captures prices as this format 1.99 , 1,99 as ocr can have mistakes between reading a , and ,

\d+ is for one or more digits

[.,] considers dot or comma

\d{2} means exactly 2 decimal digits

$ end of line

Case insensitive to ignore the case when matching for flexibility

Here re.sub is used to replace i to 1 as the OCR misread it

```
# Fix "i" to "1" for measurements like "i60G"
item_name = re.sub(r'\bi(\d+G)\b', r'1\1', item_name)
```

Regex pattern	
\b	is used to match to the word boundary
i	Lower case i was chosen as it was misread during preprocessing instead of 1 as the pattern
(\d+G)	This is the capturing group where \d+ means one or more digits followed immediately by a G as this was misread as i60G instead of 160G

\b	Is to say end of match for the word boundary
r'1\1'	Replacing i with 1 and then putting back whatever was captured inside the parenthesis in group 1 (there is only one group in this pattern)(GeeksforGeeks, 2024)

This would be useful if measurements were considered for the expiry dates however since they were manually assigned , the measurements were removed as the focus was on extracting just the food items.

Appendix K – String matching additional research

Fuzzy string matching is a technique to find strings that partially match instead of exactly matching (Dutta, 2021)

Today Fuzzy matching is used in many industries for various reasons such as in financial institutions to identify potential fraud by matching slightly different versions of names or account details to detect fraudulent activities, educational institutions to merge student records with different name or address variations with slight differences, government agencies to link public records and for fraud prevention and detection by identifying individuals attempting to manipulate records with minor variations in their personal information, in retail to identify and merge duplicate product listing with slight name or description variations (lbarrera, 2021). This technique was therefore chosen since OCR readings may still have spelling errors and variations when matching to a food database, therefore a exact match may not be possible.

The key components of fuzzy matching include (CodeWithHarry, 2025)

	Fuzzy matching components
Phonetic Matching:	Techniques such as Soundex transform words to phonetic representations based on how they are pronounced rather than how they are spelled to match similar sounding terms even when their spellings differ
Tokenization:	Tokenization breaks down the text into smaller unit tokens such as words or fragments of words to allow better handle variations in words order, spacing and punctuations when comparing the strings
Similarity Measures	Fuzzy matching algorithms use similarity measures to calculate how similar the data is between 2 strings based on character similarity, token overlap, or phonetic similarity. Common measures include Levenshtein distance, Jaro-Wrinkler distance, cosine similarity, and n-gram similarity

Thresholds and similarity scores	Fuzzy matching involves setting thresholds or similarity scores to determine when 2 strings are considered a match as the threshold is the minimum acceptable percentage to be matched. Strings below this are considered non-matches so adjusting the values can influence trade of between strict matching and broadly matching.

Focusing on similarity, an algorithm that measures the similarity one way is using Levenshtein edit distance (Pykes, 2019)

The edit distance sees how close two strings are by finding the minimum number of "edits" needed to the other strings, there are 3 main types (Pykes, 2019), using this calculation

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j) + 1 \\ \mathrm{lev}_{a,b}(i,j-1) + 1 \\ \mathrm{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Which corresponds to a delete, insert and replace.

Edit Operation	Description
Insert	Add a letter
Delete	Remove a letter
Replace	Change one letter to another

Example
"Appl" -> "Apple"
"Applle" -> "Apple"
Aplle" -> "Apple"

Combining this shows the possible of strings that are E edits away where E is the number of edit operations. Following this example, the distance between "Aplle" and "Apple" is one since "p" is replaced with an "l" leading to an exact match.

Another Example is Damerau- Levenshtein Distance which is an extension to of Levenshtein Distance that also considers swaps of 2 characters as a single edit operation for identify errors such as distance between el and le

Switch	Swap two adjacent letters	"Appel" -> "Apple"

Fuzzy matching can be implemented in python following the libraries (Kishore Nallan, 2023).

Fuzzy matching libraries in Python

library	description
difflib	Uses the ratcliff/Obershelp string matching algorithm that calculates similarity metric between 2 strings as $D_{ro} = \frac{2K_m}{\|S_1\| + \|S_2\|}$ Twice the number of matching (overlapping) characters between the two strings divided by the total number of characters in the two strings. This is calculated by sequenceMatcher
Python-Levenshtein	This is the Levenshtein distance between 2 strings in the number of insertions, replacements and deletions to calculate the edit distance and similarity ratio between 2 strings
Fuzzywuzzy	A rich library for string similarity and like python Levenshtein but also has a ratio function and more advanced functions for handling other complex string-matching scenarios such as token functions to treat individual words as

tokens and calculating pair – wise string similarity with scorers

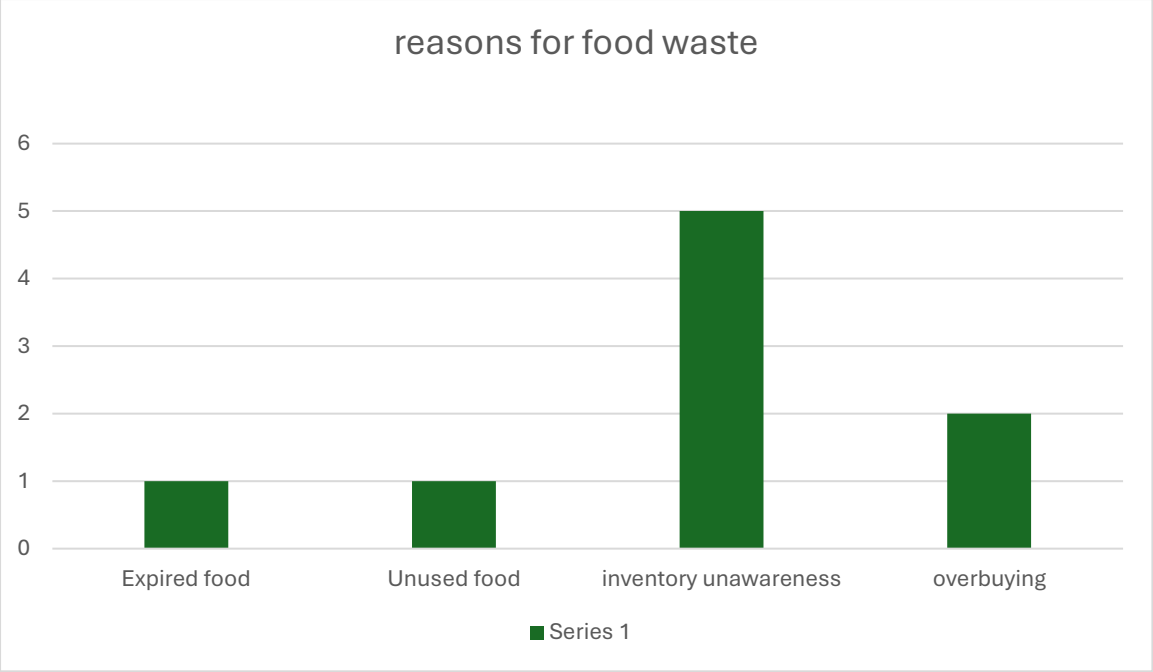
Appendix L– Fuzzy matching method

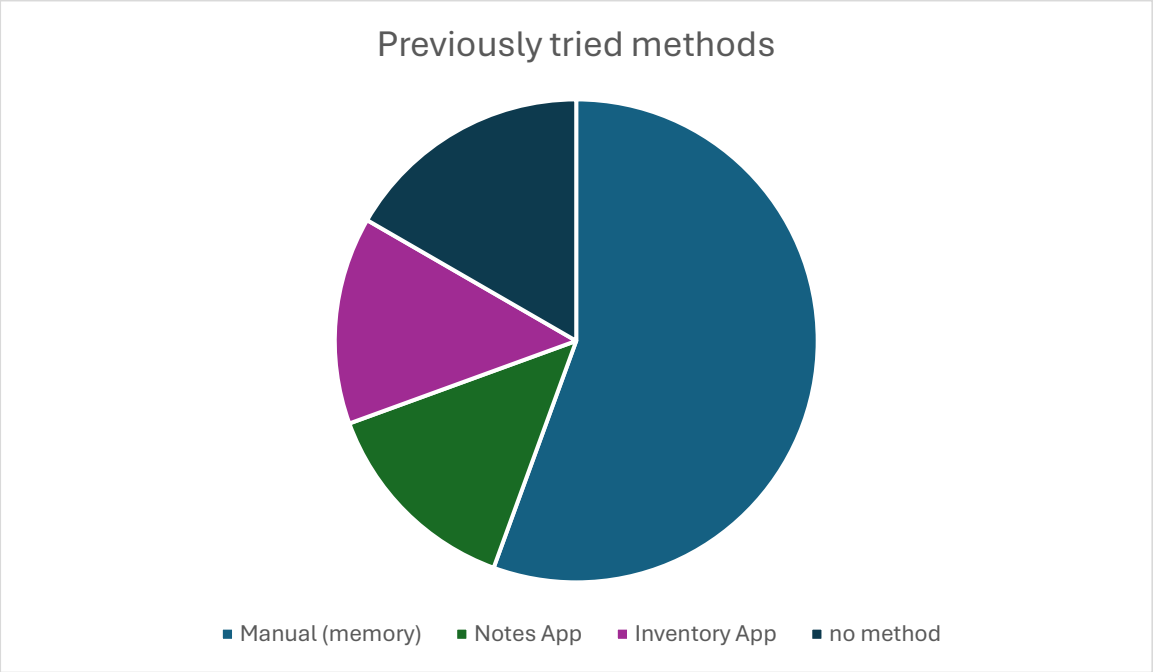
With considering the fuzzy matching approach described in section 3.2.4 and discussed further in appendix K with the libraries supported in python , Fuzzy wuzzy was chosen as the library to use for the methodology since it provides higher level tokenization methods where minor variations in word order or inconsistencies can be common in food items despite python Levenshtein being a easy library to calculate edit distances it does not handle tokenization which is important in this use case of matching the food item to the food database because there could be word order variations . Difflib also does not consider word orders so was not chosen as the method

The fuzzy library was explored and some functions was tested to see which function provided the best results (Pykes, 2019).

	Fuzzywuzzy library
Simple ratio fuzz.ratio()	Calculates the similarity considering the order of input string
Partial ratio fuzz,partial_Ratio	Finds the partial similarity by comparing the shortest string with sub-strings
Token sort ratio fuzz.token_sort_ratio()	Ignores the order of words in strings. It splits each string into individual words and sorts it into alphabetically and calculates the similarity between the 2 strings using Levenshtein distance
Token set ratio fuzz.token_set_ratio()	Removes common tokens before calculating similarity
Process	Enables to extract text from a collection

Appendix M – User Testing Results





	UI feedback
feature	**note**
Add button labels	Although this seems necessary, having too much text in a simple app design can become overwhelming for users, and apps like this are being designed so users can adopt to their everyday life meaning, since the app will be used on a daily basis , users will not be

	confused with labels as they would quickly learn icons.
Manual entry option	Although the participants wanted flexibility in cases where receipt scanning fails, currently this is out of scope since the project is evaluating the feasibility of using ai for receipt scanning to update pantry to eliminate the manual input however to consider real world problems where sometimes receipt scanning may fail future improvements would need to consider this beyond the simple fall back the author implemented
Filter by expiry	This is currently out of scope as the prototype focuses on seeing how the process would use ai to update pantry with expiry dates however this would be useful considering the entire pantry and filtering near by expiry to know what to use when as the notifications may only show a specific threshold to what the user may set such as 3 day currently. This would help with meal prepping and overbuying,
Show quantity	This would help users see their inventory and as user testers expressed, not knowing what is in their inventory, currently this feature was out of scope, so multiple cases where handles as separate duplicates however future implementation should consider this so users can properly visualise their inventory however due to this currently being out of scope in validating using ai to update user pantry with receipts this was not prioritised.

Example of user testing answers

Participant interview Questions

Current food inventory and waste management habits

- Can you describe your current approach to managing your food inventory at home? currently keep track of what food you have, what's expiring soon, and what you need to buy.

  Manage by seeing fridge or cupboard what they have and may need to buy and expiry dates when using it.

- How do you feel about the amount of food waste in your household?

  Quite a lot since can forget what is expired or expiring soon and not realise as well until being used

- Do you have any estimates on how much money or food is wasted each week or month?

  15 – 20 percent of what we spend each week around £70 – 100

  On estimate around roughly 10 percent of what they spend each week is wasted, so around £20 worth of food

- In your opinion, what are the main reasons for food waste in your home?

  Forgetting to keep track of the expiry dates and poor management of food still available within my house

- How much do you think food waste affects your household expenses?

  The user usually spend more than what is usually needed within my house, and because of this more money is wasted which could have been saved for other things.

- Have you tried any tools or methods to reduce food waste? If so, what worked and what didn't?

  No tools have been used or considered before, used just memory from what I have purchased before, so it is hard to keep track of everything

- How important is reducing food waste to you on a personal level?

  Pretty important , wasting food not good for environment, being wasteful not a good habit to have

  Somewhat important since it affects the environment, but due to being busy often not much thought is put into it

- Would an automated system that scans your receipts to track your food inventory and notify you of expiry dates be useful to you? Why or why not?

  Yes it would be useful since it would help reduce waste, wont have to keep track of stuff annually and it wont waste my time since it would be automatically done

  Yes would be very useful, will help me keep track of food that is still available within my house saving time and money

Test Interface

- Did the interface provide you with information that helped you understand your current food inventory better?

  Yes , it was very clear

  Yes It was clear

- How did the experience of scanning your receipt compare with your expectations based on our discussion about food waste?

  It matched since it showed everything I need to manage to prevent food waste

- Do you think the automated notifications and pantry updates would help you plan your meals more effectively?

  yes

Ending interview

- Any final thoughts about system or their food management habits?
  Easy to use, clear about what it does, it will be useful to a average person on a daily basis
  It is simple and easy to understand how to use, and using it per day it will help me focus more on keeping a good habit with my food usage and spending per week

All the other answers have been retained by the author (but were uploaded on Moodle for submission as extra material) and the names were removed to keep data anonymous.

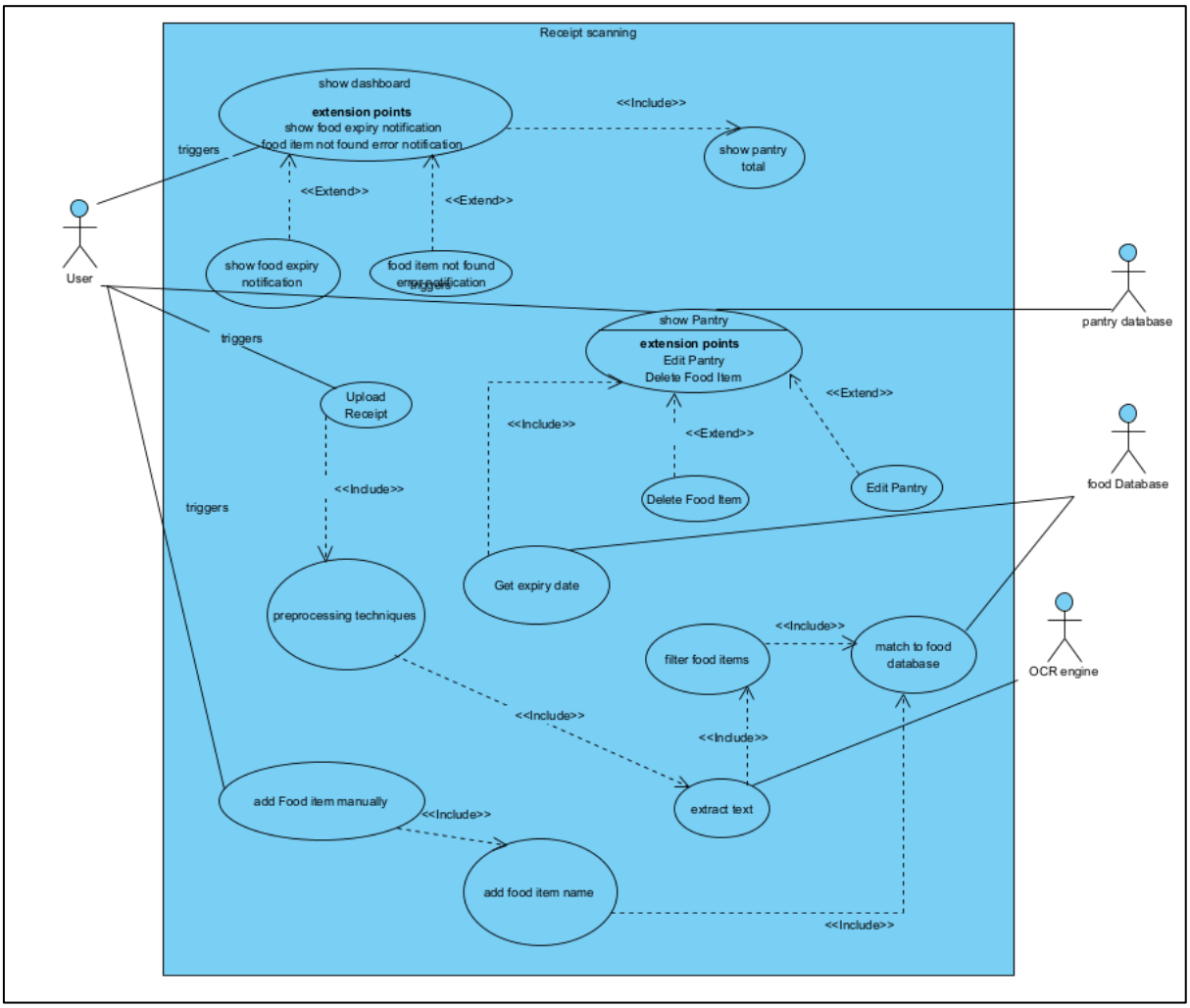
Appendix N - Requirements

Functional and non-Functional Requirements

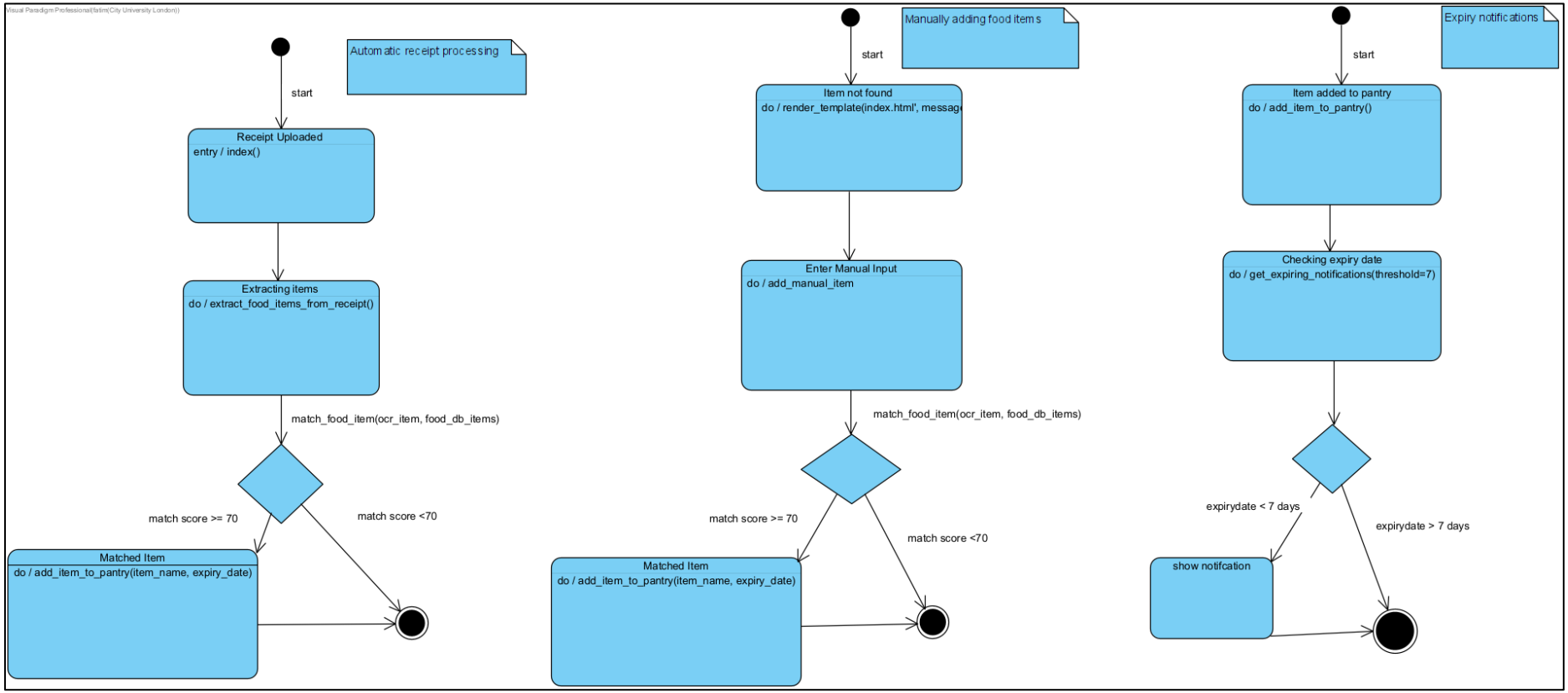
iteration	Requirement ID	Requirement Description
Sprint 1 – Baseline OCR extraction		
	1.1	The system must extract raw text from a receipt using Tesseract OCR for the output to be analysed
	1.2	The system is tested on receipts with varying backgrounds to see which condition gives good text extraction
	1.3	Evaluate the accuracy of the OCR in extracting the food item names from the receipts using CER and WER
Sprint 2 – image Preprocessing Techniques and Regex Filtering		
	2.1	Explores whether grayscale, thresholding and blurring techniques improve OCR accuracy on receipts.
	2.2	System must re-run OCR after each preprocessing technique to assess improvements on text extraction
	2.3	Parameter tuning (kernal sizes, threshold values) should be explored to assess impact on OCR output quality
	2.4	The system is tested to see if edge detection preprocessing improves OCR results
	3.1	The system should be able to process OCR output using regular expressions to filter out non-food data (e.g totals, prices, measurements)
Sprint 3-Database Matching		
	4.1	The system should be able to match the extracted food items against a local food database using fuzzy string matching
	4.2	The system is tested to evaluate how different fuzzy thresholds affect accuracy and false positives when matching food item to database to fetch expiry date

Sprint 4- Web App Integration		
	5.1	The prototype should integrate the tested OCR preprocessing, Regex filtering and matching database layers into a web application
	5.2	The web app should allow a user to simulate uploading a receipt and viewing extracted food items with its expiry dates automatically updated
	5.3	Interface should allow users to view and edit entries of their pantry
Sprint 5- Technical Testing		
	6.1	The system is evaluated using more receipt images with different qualities to assess robustness

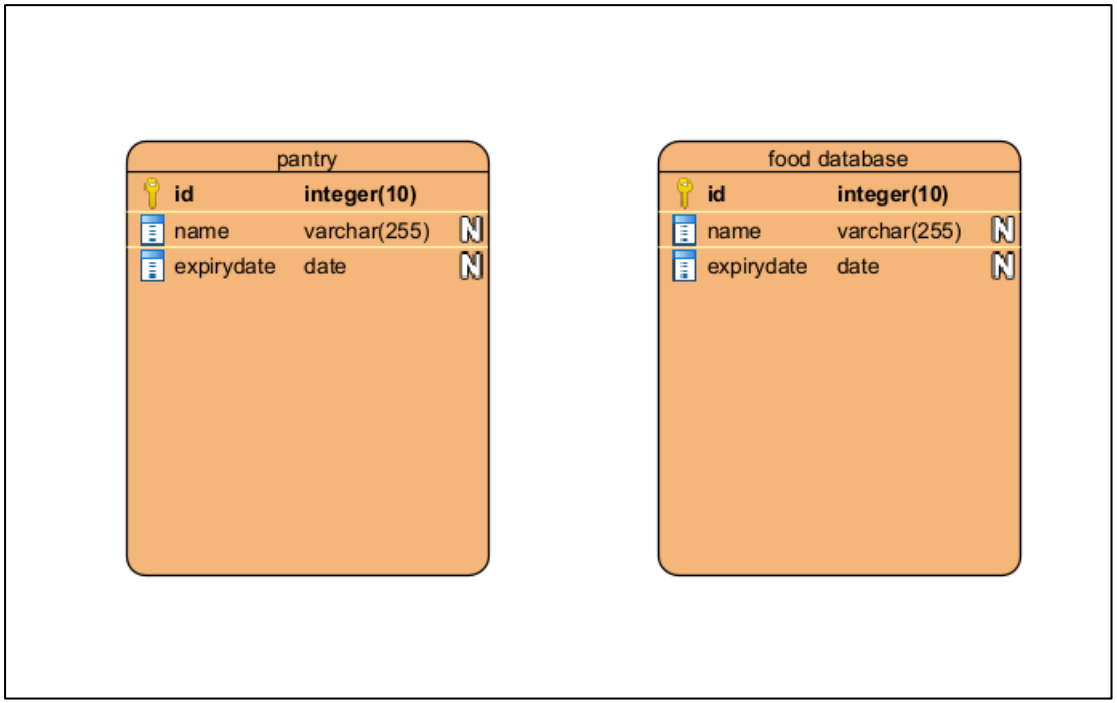
Use Case


Receipt scanning
show dashboard
extension points
show food expiry notification
food item not found error notification
<<Include>>
show pantry total
triggers
<<Extend>>
<<Extend>>
User
show food expiry notification
food item not found error notification
triggers
pantry database
triggers
show Pantry
extension points
Edit Pantry
Delete Food Item
<<Extend>>
Upload Receipt
<<Include>>
<<Extend>>
food Database
<<Include>>
triggers
Delete Food Item
Edit Pantry
Get expiry date
preprocessing techniques
<<Include>>
filter food items
match to food database
OCR engine
<<Include>>
<<Include>>
add Food item manually
<<Include>>
extract text
add food item name
<<Include>>

State Diagram



ERD diagram


pantry
id integer(10)
name varchar(255) N
expirydate date N
food database
id integer(10)
name varchar(255) N
expirydate date N

Appendix O – Research Questions and Finding

	Highly effective
	Moderately effective
	ineffective

	Research questions			
	How accurately can Tesseract extract food item names from shopping receipts?			
Investigated through:	Evaluation	Findings	Future improvements	
Sprint 1 + 2 Sprint 5 Chapter 5	OCR alone With pre-processing	Through analysing the results from the results chapter, receipt 2 showed good initial baseline results so was subsequently selected for the next sprint to further improve. The other receipts showed that receipts that were noisy (backgrounds) struggled to read the receipt which included extra words and characters due to the noise affecting the reading which influenced a preprocessing step of removing backgrounds as a potential stage to help with receipts with noisy backgrounds, the blurry texts often led to text being misinterpreted and characters as shown by low CER rates and WER rates influencing a need for blurring techniques to help the ocr recognise the text as well as contrast to make the text more bold and clear. After the preprocessing techniques were tested	More techniques could be explored which were not explored here to improve the ocr text extraction even more, limiting the amount of post processing required after using OCR to extract the food items such as skew for any receipts that could be taken not straight, and edge detection to detect the edges and crop the background. An OCR model could also be fine tuned or trained with a dataset to better understand receipt	

		and explored, tesseract was able to extract the food items more well which aligns with the research in chapter 3,where preprocessing steps are essential to help the OCR text be more recognised . The blur affects helped reduce any text blurs, grayscale removing any colour that could affect the OCR as well as the thresholding to contrast the text to make it more clear.	fonts, layout styles and text alignments.	
	How effective is a rule-based text cleaning approach using regular expressions in extracting food item names from OCR output?			
Sprint 2 Sprint 5	Regex Regex + post-processing	The use of regular expressions was effective as a baseline to extract the food name and price however sometimes also captured other details such as total, and points and such since they also contained numbers and such so post processing was required to filter out irrelevant lines such as “total” to not capture those as well as a abbreviation dictionary to capture words that are abbreviated as it’s actual food item name as discussed in chapter 3 as a post processing method as well as adding patterns to correct ocr errors such as converting ‘I’ to ‘1’ to remove measurements which were not needed. This combined strategy resulted in accurate extraction for the chosen receipt however during technical testing in sprint 5 it was cear that it struggled with inconsistent receipts therefore lacked general robustness across varying layouts.	This could include using a nlp model such as bert to classify items or identify them to learn patterns from annotated receipt data and adapt to unseen formats. It can be used to pre train on unlabelled data text where it can learn the context and representations of words (pawangfg, 2020). This would allow system to learn generalisable patterns from annotated receipts data and then adapt to unseen formats	
	What is the best method for matching extracted food items to a food database?			

Literature review chapter 3 Appendix L Sprint 3	Fuzz.ratio() Fuzz.partial_ratio() Fuzz.token_sort_ratio() Fuzz.token_set_ratio()	After the literature review in section 3.3.4, and Appendix L, Fuzzy matching was chosen as the method due it its flexibility and token- based methods to handle word order variations which aligned with the findings from wang,Li and fe(2011) who showed its robustness in string similarity so using fuzzy wuzzy some methods showed high a recall where the items were correctly matched and some incorrectly matched with some methods showing high precision where the food items what were matched were actually correct. Fuzz ratio performed well because there were no word order changes , partial_ratio() did not always perform well as it compares the longer strong with the shorter one giving incorrect matches such as “apple” for “apple pie” this would not be ideal for a food database matching since apple does not equate to apple pie which means it would retrieve incorrect expiry dates, making the system flawed. Token set ratio focuses on common words and ignores extras therefore is not ideal as its best used for use cases that need to match common tokens so in this case “apple” matched to “pineapple” which would again give an incorrect expiry date making the system flawed. Token sort ratio tokenized the strings and sorted them alphabetically, so this is good for inconsistent ordering . Token sort ratio was found to offer more reliable performance however to further assess reordered words may need to be tested against partial.ratio() as they both provided high precision, however	Although Fuzzy wuzzy was chosen, the methods may need further testing with more food item data, as only a small amount of foods were tested therefore may not be able to generalise well if more similar food items were added. The other distance techniques were not tested as mentioned in the appendix which could be better at string matching then fuzzy matching that uses Levenshtein distance in this proof of concept with a small food database. So to improve more food items should be tested to assess its speed in matching as well as accuracy since, users may not want a slow system. Therefore, this must be tested before being developed to see how it handles more receipts with more similar item names.	

		token_sort was chosen as the method due to its ability to handle reordered words.		
	What are the key technical challenges in using receipt scanning for food inventory automation?			
Sprint 1-5	OCR reliability Generalisability of Regex Fuzzy-matching performance Automation reliability	Tesseract performed very well on the controlled receipt however failed on different receipts during sprint 5 as the output relied on image quality and receipt format. Common Abbreviations were handled with dictionary mapping to help with database matching. The regex worked well on the control receipt however failed with different receipts since they have different format and layouts which shows its sensitive to changes outside the controlled environment meaning fuzzy matching heavily relies on the OCR consistency and the regex pattern to know the food items therefore failed when tested with a different receipt	Future improvements may require a custom tesseract model trained on receipts to improve generalisation, A robust abbreviation dictionary could be made to handle more abbreviations to be able to match more food items however this may not be suitable for a long-term solution as there are many food items , nlp could be more beneficial to understand the food items.	

References

Dray, S. (2021). Food Waste in the UK. *House Of Lords Library*. [online] Available at: https://lordslibrary.parliament.uk/food-waste-in-the-uk/.

List, I. (2025). *Read 22 NoWaste Reviews (2025)*. [online] JustUseApp. Available at: https://justuseapp.com/en/app/926211004/nowaste-food-inventory-list/reviews [Accessed 2 Feb. 2025].

Ridvy Avyodri, Lukas, S. and Hendra Tjahyadi (2022). Optical Character Recognition (OCR) for Text Recognition and its Post-Processing Method: A Literature Review. doi:https://doi.org/10.1109/ictiia54654.2022.9935961.

Ullah, H., Altamimi, A.B., Uzair, M. and Ullah, M. (2018). Anomalous entities detection and localization in pedestrian flows. *Neurocomputing*, 290, pp.74–86. doi:https://doi.org/10.1016/j.neucom.2018.02.045.

Kaderabek, A. (2023), Exploring Optical Character Recognition (OCR) as a Method of Capturing Data from Food-Purchase Receipts. *Survey Methods: Insights from the Field, Special issue: 'Food Acquisition Research and Methods'*. Retrieved from https://surveyinsights.org/?p=17190

Patel, C., Patel, A. and Patel, D., 2012. *Optical character recognition by open source OCR tool Tesseract: A case study*. International Journal of Computer Applications, 55(10). Available at: Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study

Karandish, F. (2019). *A Comprehensive Guide to Optical Character Recognition (OCR)*. [online] Moov AI. Available at: https://moov.ai/en/blog/optical-character-recognition-ocr.

Quested, T.E., Parry, A.D., Easteal, S. and Swannell, R. (2011). Food and Drink Waste from Households in the UK. *Nutrition Bulletin*, [online] 36(4), pp.460–467. doi:https://doi.org/10.1111/j.1467-3010.2011.01924.x.

Vina, A. (2024). *The Role of Computer Vision in OCR: Enhancing Text Recognition*. [online] Ultralytics.com. Available at: https://www.ultralytics.com/blog/the-role-of-computer-vision-in-ocr-enhancing-text-recognition.

Jain, P., Taneja, Dr.K. and Taneja, Dr.H. (2021). Which OCR toolset is good and why? A comparative study. *Kuwait Journal of Science*, 48(2). doi:https://doi.org/10.48129/kjs.v48i2.9589.

Wisniowski, K. (2022). *SQLite vs SQL - What's the Difference ? (Pros and Cons)*. [online] Cloud Infrastructure Services. Available at: https://cloudinfrastructureservices.co.uk/sqlite-vs-sql-whats-the-difference/.

Martinez, J. (2025). *What is the OCR Accuracy and How it Can be Improved*. Available at: https://www.docuclipper.com/blog/ocr-accuracy/.

Wang, J., Li, G. and Fe, J. (2011). Fast-join: An efficient method for fuzzy token matching based string similarity join. *CiteSeer X (The Pennsylvania State University)*. doi:https://doi.org/10.1109/icde.2011.5767865.

Pythonbasics.org. (2021). *What is Flask Python - Python Tutorial*. [online] Available at: https://pythonbasics.org/what-is-flask-python/#jinja2 [Accessed 24 Apr. 2025].

GeeksforGeeks (2023). *Flask Tutorial*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/flask-tutorial/.

GeeksforGeeks. (2022). *Flask - HTTP Method*. [online] Available at: https://www.geeksforgeeks.org/flask-http-method/.

Palletsprojects.com. (2024). *Template Designer Documentation — Jinja Documentation (3.1.x)*. [online] Available at: https://jinja.palletsprojects.com/en/stable/templates/.

W3Schools (2019). *Python RegEx*. [online] W3schools.com. Available at: https://www.w3schools.com/python/python_regex.asp.

Bogotobogo.com. (2021). *Python Tutorial: Regular Expressions with Python - 2021*. [online] Available at: https://www.bogotobogo.com/python/python_regularExpressions.php

OpMt (2012). *How .* (dot star) works?* [online] Stack Overflow. Available at: https://stackoverflow.com/questions/12666768/how-dot-star-works.

GeeksforGeeks (2024). *Python Regex: Replace Captured Groups*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/python-regex-replace-captured-groups/

developer.mozilla.org. (n.d.). *Regular expression syntax cheatsheet - JavaScript | MDN*. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions/Cheatsheet.

pythonexamples.org. (n.d.). *Python re.sub() – Replace using Regular Expression - Python Examples*. [online] Available at: https://pythonexamples.org/python-re-sub/

GeeksforGeeks (2020). *Python Substituting patterns in text using regex*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/python-substituting-patterns-in-text-using-regex

Dutta, M. (2021). *Fuzzy String Matching – A Hands-on Guide*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/07/fuzzy-string-matching-a-hands-on-guide/.

Pykes, K. (2019). *Fuzzy String Matching in Python Tutorial*. [online] Datacamp.com. Available at: https://www.datacamp.com/tutorial/fuzzy-string-python?dc_referrer=https%3A%2F%2Fwww.bing.com%2F [Accessed 28 Apr. 2025].

lbarrera (2021). *Fuzzy Matching 101: Cleaning and Linking Messy Data*. [online] Data Ladder. Available at: https://dataladder.com/fuzzy-matching-101/.

CodeWithHarry. (2025). *CodeWithHarry - Learn Programming and Coding*. [online] Available at: https://www.codewithharry.com/blogpost/fuzzy-matching-and-how-it-works

Kishore Nallan (2023). *Fuzzy string matching in Python (with examples)*. [online] ExampleSite. Available at: https://typesense.org/learn/fuzzy-string-matching-python/.

pawangfg (2020). *Explanation of BERT Model - NLP*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/.