.

Architecture Diagram

Graphical user interface, diagram

Description automatically generated

Diagram

Description automatically generatedTimeline

Software used

|  |  |  |
| --- | --- | --- |
| Python | React Native | MongoDB |

Hardware

|  |  |  |  |
| --- | --- | --- | --- |
| Raspberry PI 4 | PI Camera | Barcode Scanner | Mobile Phone |

A picture containing company name

Description automatically generated

The Smart Kitchen

Project Engineering

Year 4

Niall McLoughlin

Bachelor of Engineering (Honours) in Software and Electronic Engineering

Atlantic Technological University

2022/2023

**Declaration**

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) in Software and Electronic Engineering at Galway-Mayo Institute of Technology.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

Niall McLoughlin

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Acknowledgements**

Use this section to acknowledge anyone, if you wish to, who might have helped during your project.

**Table of Contents**

[1 Summary 6](file:///C:\Users\cex\Downloads\ProjectEngReportExampleTemplate.docx#_Toc64975700)

[2 Poster 7](file:///C:\Users\cex\Downloads\ProjectEngReportExampleTemplate.docx#_Toc64975701)

[3 Introduction 8](file:///C:\Users\cex\Downloads\ProjectEngReportExampleTemplate.docx#_Toc64975702)

[4 Background 9](file:///C:\Users\cex\Downloads\ProjectEngReportExampleTemplate.docx#_Toc64975703)

[5 Project Architecture 10](file:///C:\Users\cex\Downloads\ProjectEngReportExampleTemplate.docx#_Toc64975704)

[6 Project Plan 11](file:///C:\Users\cex\Downloads\ProjectEngReportExampleTemplate.docx#_Toc64975705)

[7 Heading 12](file:///C:\Users\cex\Downloads\ProjectEngReportExampleTemplate.docx#_Toc64975706)

[7.1 Referencing 12](file:///C:\Users\cex\Downloads\ProjectEngReportExampleTemplate.docx#_Toc64975707)

[7.2 Notes on Content 13](file:///C:\Users\cex\Downloads\ProjectEngReportExampleTemplate.docx#_Toc64975708)

[8 Ethics 14](file:///C:\Users\cex\Downloads\ProjectEngReportExampleTemplate.docx#_Toc64975709)

[9 Conclusion 15](file:///C:\Users\cex\Downloads\ProjectEngReportExampleTemplate.docx#_Toc64975710)

[10 Appendix 16](file:///C:\Users\cex\Downloads\ProjectEngReportExampleTemplate.docx#_Toc64975711)

[11 References 17](file:///C:\Users\cex\Downloads\ProjectEngReportExampleTemplate.docx#_Toc64975712)

# Summary

The Smart Kitchen is a food inventory system designed to help people manage the food in their kitchens, recommend recipe ideas, save money and ultimately help reduce personal food waste. A Report by the Irish Government published in 2021 states that over 700,000 TONNES of food is wasted in the country every year [1], with a third of this coming from households. Of this household waste 60% is classified as avoidable food waste, coming from “Plate scrapings, leftovers, gone off fruit and veg and passed its date perishables”. This adds up to a cost of between 400 and 1000 euro per household every year. With climate change bringing more extreme weather events, the certainty of our food supply chain could become a major issue in the near future. This is why I believe reducing personal food waste is a matter of great importance.

The project uses a barcode scanner to track and store which food items are in your kitchen, and a camera using Optical Character Recognition (OCR) to track best before dates on perishable items. Once all food items are stored in a ‘digital pantry’, this can be viewed on a mobile application, with recipe ideas being generated based on the user’s food inventory.

The project uses a Raspberry PI 4 with a camera module to read food best before dates, with a one-dimensional barcode scanner being used to enter barcode data. This information is then uploaded to a database which can the accessed via the mobile application. The application is written in JavaScript using React native as a framework, with MongoDB being used to handle the database. Optical Character recognition is achieved using Tesseract and OpenCV.

# Poster

A picture containing timeline

Description automatically generated

# Introduction

The Smart Kitchen was created with the aim of providing an easy-to use and intuitive way for an individual to manage their food inventory. The project was designed to streamline and simplify both food purchasing and consuming decisions, making it easier for users to plan out their meals and helping them to make smarter choices relating to food. By helping users keep track of their current food inventory the project hopes to reduce persona food waste which can in turn have an effect of the environment both locally and globally.

The motivation for this project stemmed from a personal desire to reduce food waste, remove the stress related to food management and to lower food expenditure.

the Irish environmental Protection agency released a food waste [2] hierarchy earlier this year which detailed steps which can be taken in order to the prevent food waste. Although there are food redistribution and donation services available, the most import and step detailed in the hierarchy was “Prevention at source”, this was the smart Kitchen aims to achieve.

The intended scope of project is for use by individual in the home. In an age where smart phones are used for an ever-increasing range of tasks, why not make an application that makes it easy for people to be both savvy and conscientious when purchasing and consuming food.

# Background

This project has two main aspects. The Hardware elements of the project work towards data acquisition and entry, and the mobile application element which works towards using and displaying the Data. In this section I will go into details about how both of these were achieved.

4.1

Raspberry PI and associated hardware

### – Raspberry PI

Raspberry Pi is the name given to range of compact and powerful single board computers created by the Raspberry Pi foundation who are based in the UK. All Raspberry pi’s are powered by ARM processors, who are a company who specialize in creating efficient and lower power consumption processors making them extremely popular in the embedded systems market.

The Raspberry Pi can operate using a range of operating systems, however the standard and officially supported operating system is Raspian. Raspian is a Debian-based Linux Operating System which is specifically designed to be optimized for the Pi. It focusses on a simplistic and user-friendly approach which makes it popular with hobbyists of all skill levels.

Most Raspberry Pi’s come with several GPIO pins and serial connections, expandable storage capabilities, and have built-in networking capabilities, this makes raspberry Pi’s the number one choice for IOT style projects,

The Original Pi, named the model B was created in February of 2012, it boasted a 700MHz ARM1176jZf-S processor along with 512MB of RAM. The model that was used for this project (the Raspberry Pi 4 model B) uses the 1.5Ghz ARM Cortex-A72 and comes with 2GB of RAM [3], which is vastly more powerful than the original model.

In summary the Raspberry Pi model 4 is a lightweight and compact board, while still being more than powerful enough to handle all requirement for the project.

### - pi camera

The Pi Camera Module was first released in May 2013, with its predecessor the Pi Camera module 2 being released in April 2016. It is a compact camera module designed specifically for use with the raspberry pi and interfaces with the board using a custom CSI (Camera Serial Interface). The Module 2 has a high-quality Sony IMX219 8-megapixel sensor with a pixel resolution of 3280x2664. This is important within the scope of the project, as the higher the picture quality, the easier it is to carry out Optical Character Recognition

### -Barcode Scanner

The Project uses a one-dimensional (1D) barcode scanner to read barcode labels. An example of a 1D barcode would the standard labels placed on retail Items

## Mobile application

The Mobile application element of the project was using a modified MERN Stack. MERN stands for MongoDB, Express, React/react Native and Node.js, each of these terms will be explained in depth throughout this section.

MERN is a popular and powerful development stack with many benefits, due to the entire stack being built in JavaScript, a consistent programming style and syntax through out the entirety of the application. Due to the popularity of the MERN stack there is a large community of developers providing resources and tutorials online which was a major benefit throughout the project.

### – react native

React is an open-source UI framework created by Facebook in March 2015, used for creating web applications. React Native is a UI framework used for creating cross-platform (Android and IOS) mobile applications. React Native uses a component-based architecture, this means it comes with several pre-built components which are used to create UI elements, and programmers can combine and alter native components to create custom components.

React is currently the second most popular mobile application development framework in 2023, holding a 38% market share {ref}.

### – node.js

Node.js is an open-source JavaScript runtime environment released in 2009, which used to develop server-side applications. The main benefits of node.js is that it is written in JavaScript, which, as said above, allows for consistent coding practices thought the entire application.

Node.js uses an event driven architecture based of an “event loop”. This allows node.js to process multiple operations in a non-blocking manner, by triggering a callback function when a request comes in and adding it to a queue.

### – express

Express is a web application framework for node.js which provides features to help build web application and interface with APIs

It allows users to create routes for handling incoming requests and allows users to pass parameters inside these requests.

### – MongoDB and atlas

Tesseract is an open-source Optical Character Recognition (OCR) engine developed by Google. OCR is a process of extracting texts from images, such as scanned documents or photographs. Tesseract can recognize text in over 100 languages, and it can process both printed and handwritten text (depending on the handwriting).

## – OCR

OCR stands for Optical Character Recognition, is the technology used to convert printed or handwritten text from images/ scanned documents into readable digital text.

OCR technology uses various image processing and machine learning algorithms which are trained using large datasets to analyze images, detect text, and recognize characters. The image is first pre-processed to remove any noise or distortion and improve clarity. The text is then broken up into individual characters or words. Finally, the characters are recognized and translated into machine-readable text.

4.3.1 – open CV

OpenCV (Open-Source Computer Vision) is a free and open-source computer vision and machine learning software library that is used to analyze, manipulate, and understand visual data from various sources, including images and videos. It was first developed in 1999 by Intel and is now maintained by the OpenCV community. OpenCV is designed to be cross-platform, so it can run on a wide range of devices, including desktops, mobile phones, and embedded systems.

OpenCV is written in C++ and provides interfaces for several other programming languages, including Python, Java, and MATLAB. The library includes a wide range of image processing and computer vision algorithms, including feature detection, object recognition, segmentation, and tracking. These algorithms can be used to perform tasks such as face detection, gesture recognition, autonomous driving, and medical imaging.

One of the main features of OpenCV is its support for real-time computer vision applications. OpenCV includes high-performance algorithms that can process images and videos in real-time, which is essential for applications such as robotics, surveillance, and augmented reality.

4.3.2 – Tesseract

Tesseract is an open-source Optical Character Recognition (OCR) engine developed by Google. OCR is a process of extracting text from images, such as scanned documents or photographs. Tesseract can recognize text in over 100 languages and can process both printed and handwritten text.

Tesseract was initially developed by Hewlett-Packard in the 1980s, but it was later released as an open-source project in 2005. Google later acquired the project and has been maintaining it since then. Tesseract is written in C++ and it supports interfaces for several programming languages, including Python, Java, and C#.

One of the key features of Tesseract is its accuracy. It has been shown to achieve high levels of accuracy in recognizing text from various sources, including low-quality scans and images with skewed text. Tesseract also includes several preprocessing technique, such as binarization and noise reduction, to improve its accuracy.

Tesseract is widely used in various applications, including document digitization, text recognition in images and videos, and optical character recognition in mobile applications. It is also integrated with several other open-source libraries, such as OpenCV (which provides image processing functions) to provide a complete solution for text recognition.

# Project Architecture

Graphical user interface, diagram

Description automatically generated

Figure 4‑1 Architecture Diagram

# Project Plan

I used Jira as my project management tool, updating it every second week after stand-up meeting, and assigning new tasks through 2-week sprints. In the beginning of the project, it was quite difficult to gauge how many tasks could be completed within each 2 period, however throughout the course of the project it became easier to accurately assign a manageable workload.

# Project Code

## Front-end code

The Front-end code for the project consists of 7 different screens utilizing 14 different custom-made components and numerous pre-made components. The Project uses Bottom tab navigation to navigate between the 5 main screens, it also contains a stack navigator for navigating around the option screen. The 5 main screens of the project are as follows:

Pantry Screen:

This screen contains the list of all available food items used in the user’s inventory, items can be added for the raspberry PI or manually from with in the app. There are two main ways to display a list of items in react native, one is a scroll view, the other is a flat list. The main difference between a scroll view and a flat list is how items are rendered, in a scroll view every item in the list in rendered when the list is initially loaded, a flat list on the other hand uses “lazy rendering” meaning that list items are only rendered when the user is nearing them. Lazy rendering is beneficial when dealing with dynamic lists, as if a list were to contain thousands of items, rendering every item at once could cause significant performance issues. The Fetch Pantry uses a flat list to display the items, it fetches these list items from the database using a “fetch pantry” function. Each list item displays the name, amount, and use-by date of the pantry item.

Each list item has two buttons. The first button is update, which allows the user to alter the quantity of an item. It does this by invoking the update pantry component, which calls a function to retrieve and update an item from the database.

The Second button is a delete button. This button uses the delete pantry component to call a function which removes the item from the database. Both buttons the product ID which is passed to it to identify which list items to act on.

Shopping screen:

The Shopping screen is intended to act as a quick on the go shopping list to reminder users what items to buy. uses a flat list to display a list of items similarity to the pantry screen, however the shopping list does not have an update function, only a delete.

Items are adding to the shopping list by using a text input box on the top of the screen.

Recipe Screen:

The recipe screen features a flat list of recipes. A picker (drop down menu) at the top of the screen allows the user to toggle between all recipes and available recipes. If the user selects available recipes, the fetch recipe component fetches all food items in the pantry and cross references then with the ingredients need to make each recipe. If the pantry contains all the needed ingredients, then the recipe will display in the available recipes section, otherwise the recipe will disappear.

The project comes with several pre-loaded recipes. However, users can enter their own custom recipes by navigating over to the options screen which contains a nested add recipe screen.

Due to the length of each recipe (Name, description and up to five ingredients), the flat list items contain only the names of each recipe. Each Items on the list is a pressable component that when pressed, prompts a modal to pop up containing all details of the pressed recipe, this allows easy navigation on what could otherwise be a very cluttered page.

Home Screen:

This is the introductory screen of the project. It welcomes the user to the application and informs them of any items in the pantry that do not have use-by dates associated with them.

It displays any items that have been scanned in by the barcode scanner but are not recognised i.e. are not present in the barcodes database and therefore do not have any corresponding information. The user then has two options, either dismiss the item or save it, and give it a name and amount. If the user chooses to save the item, it is entered into the pantry list and the barcode and corresponding name are saved to the barcode database so that the item will be recognised in the future.

## Back-End code

The back-end code an express.js application which is used to handle interactions with the mongo atlas database. An express.js routes handle the different routes involved in the application. Each Collection in the database has a separate schema associated with it. A schema is used to define the structure of each database model i.e., the various information fields for each item. Full Crud functionality is achieved by having four different routes associated with each collection, these are: add, update, delete and fetch. Except for barcode which only has an add route.

Password checking and encryption is also handled within sever code by using bcrypt. Bcrypt is a password hashing function used to encrypt passwords. The password is salted 12 times, salting is act of add a random piece of information data (in this case a string) to the password. This prevents hackers from using precomputed tables of hashed passwords.

7.3 Python Code

The Raspberry pi continuously runs a python script in order to operate. The script first connects to the mongo atlas database using Pymongo. The script waits for a barcode to be scanned, it then checks if this barcode is present in the main barcode database which is stored in a local file, if the barcode is not present it checks in the user created barcode database. If the barcode is present in either of these databases, it adds the pulls the item name and quantity for these databases and add the item along with all corresponding information to the panty database. If the barcode is not present in either of the barcode databases, it still adds the item to the pantry database but assigns the word “unknown” to the name and quantity field, which will flag the item for the uses attention when viewed on the mobile application.

The script also implements OCR, it does this triggering the Pi camera module to take five pictures once a barcode has been scanned. Once these five pictures are taken, they are saved locally. Thresholding is then applied to the pictures using OpenCV. Thresholding is an image processing technique which involves converting an grayscale image into a binary image, meaning that each pixel in the image is either completely black or white (255 or 0). This helps to define text on the image and makes it easier for Tesseract to complete its optical character recognition.

Tesseract first applies image processing techniques to improve the text visibility of the image and prepare it for OCR, such as scaling to make the image a standard size and noise reduction to improve the clarity, it also applies binarization (similar to thresholding). It then breaks the image up into blocks and processing each block to look for individual characters. It looks for characters by analysing each block using characters detection algorithms from statistical and neural network models. These models are trained using large datasets of images which have already known text. Based on this training the OCR engine decides which characters are present in the image with a high degree of accuracy. Although this accuracy does depend on many factors such as the image resolution, object distance from camera, original text quality and angle of text relative to the camera.

Once the optical character recognition is completed, the text in stored in a buffer, string parsing is then used to look for a group of characters than conform to the standard date format e.g., xx/xx/xxxx. If this criterion is met, the date information is then passed to the barcode item as the date field and saved to the database.

# Ethics

The Smart Kitchen’s goal of reducing food waste leads this to be an inherently ethical project. With the effects of climate change becoming more apparent every year, the need for individuals to be responsible with their food choices has never more important. If even a small amount of food is saved due to the creation of this project, it will have a positive impact overall.

# Conclusion

The time spent developing the smart Kitchen has been challenging but rewarding. The project has a capacity to make the lives of its users easier and hopefully has a positive impact. The project achieved its originally stated goals of using to OCR to capture use by dates, and its goal of have a recipe selection which displayed dynamically based on the user’s pantry items.

In conclusion, this project has given me a far greater understanding and appreciation of the intricacies and work required in order to build a full stack application.

There are some features which could still be added to the project, such as ingredient substitution and pre-loaded shopping lists. However, the project in its current form still provides an excellent platform for users to base their food decisions on and takes away much of the stress related to food purchasing and consumption.

# References

|  |  |
| --- | --- |
| [1] | Irish Environmental Protction Agency. [online] Available: https://www.epa.ie/our-services/monitoring--assessment/waste/national-waste-statistics/food/ . [Accessed 01 05 2023] |
| [2] | Irish Environmental Protction Agency. [online] Available: https://www.epa.ie/our-services/monitoring--assessment/waste/national-waste-statistics/food/ . [Accessed 01 05 2023] |
| [3] | Raspberry Pi Foundation. [online] Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/ . [Accessed 01 05 2023] |
| [4] | Syndell Tech. [Online]. Available: https://syndelltech.com/flutter-vs-react-native-which-is-best-in-2023/. [Accessed 01 05 2023]. |
|  |  |
|  |  |