Smart Dispensing System for Zero-Waste Shops

Student Name: Arthur Milner
Student Number: 21035478

UXCFXK-30-3 Digital Systems Project

Approximate Word Count: 10850

1 Abstract

Zero-waste shops provide a modern approach to sustainable shopping, however, the technologies in-place often fail to address key challenges. These include frequent spillages, inefficient stock monitoring, and unoptimized customer flows. This project presents a prototype solution to these challenges through the development of ESP32-powered dispensers, InfluxDB business-facing dashboards, employee notifications via Slack, and a Flask-based self-checkout system. Communication between these components is facilitated by Radio-Frequency Identification (RFID), Message Queuing Telemetry Transport (MQTT), and Node-RED. The final system being an Internet of Things (IoT) solution that enhances the efficiency and ease of both the staff and customer experience within zero-waste shops.

1.1 Git Repository

The following link provides access to a repository of artefacts and code used in the development and testing of my project:

`https://github.com/arthurmilner01/digital-systems-project`

2 Acknowledgements

I would like to express my gratitude towards Neil Phillips for supervising and providing continued support throughout the course of my project, Craig Duffy for the initial inspiration behind my project, and finally my friends and family for support during my time at university.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence

API Application Programming Interface

CRUD Create Read Update Delete

ERD Entity Relationship Diagram

GUI Graphical User Interface

IoT Internet of Things

JSON JavaScript Object Notation

MQTT Message Queuing Telemetry Transport

PCI Payment Card Industry

RFID Radio-Frequency Identification

SWOT Strengths Weaknesses Opportunities Threats

UI User Interface

UID Unique Identifier

3 Introduction

In response to public demands for more sustainable practices, the popularity of zero-waste shops continues to grow each year, but the processes within these shops fail to provide optimised customer flows, efficient dispensing, and competitive prices. The project aims to tackle this with an affordable modernisation of the shopping process, achieved through a lightweight IoT solution inspired by technologies used within supermarkets. The system leverages services such as MQTT, InfluxDB, and Node-RED, combined with sensors, microcontrollers, and a Python Flask web-based checkout. The project's primary focus is on the operational efficiency of zero-waste stores, with the hope that parts of the proposed solution could also translate to larger chains, promoting the wider adoption of zero-waste in retail.

3.1 Aims and Objectives

For my aims and objectives, I have taken a SMART approach, this is because projects without a complete framework for their goals are less likely to attain their desired outcomes **(Ogbeiwi, 2017)**[36]. The aims below are time-bound to 3 weeks before the project deadline.

3.1.1 Aim 1

To create a prototype of an affordable IoT-based smart dispensing system capable of handling products within a zero-waste shop. Designed to increase the efficiency of the customer flow by at least 15%.

3.1.1.1 Objective 1A Use an ESP32 with low-cost components to dispense products, calculating weight/cost with ±10% accuracy. This should be achieved 8 weeks before the project deadline.

3.1.1.2 Objective 1B Test 25 dispenses using the prototype system to estimate whether customer flow efficiency increases by at least 15% compared to systems in established zero-waste shops. This should be achieved 3 weeks before the project deadline.

3.1.1.3 Objective 1C Establish a Wi-Fi connection with 99% or greater uptime on the ESP32 to allow communication with other systems. This should be achieved 10 weeks before the project deadline.

3.1.1.4 Objective 1D

Configure Mosquitto MQTT broker and Node-RED to receive data from multiple ESP32s and correctly send the data depending on the context, with a 99% or greater uptime. This should be achieved 4 weeks before the project deadline.

3.1.2 Aim 2

To implement a fully functional Flask-based self-checkout which receives, manages, and stores data from dispensers into a database and processes it into payable transactions. The self-checkout should be familiar to customers and aim to process a customer checkout in under 1 minute.

3.1.2.1 Objective 2A

Integrate Stripe Application Programming Interface (API) to ensure secure and Payment Card Industry (PCI) compliant payment processing in under 20 seconds. This should be achieved 7 weeks before the project deadline.

3.1.2.2 Objective 2B

Store dispenses within an SQLite database, users scan their RFID tag to retrieve their dispenses from the database, saving an average of 10 seconds inputting their tag ID. This should be achieved 7 weeks before the project deadline.

3.1.2.3 Objective 2C

Add user account functionality to allow purchases to be paid with loyalty points and automatically email digital receipts, increasing customer retention by at least 10%, and reducing paper usage by at least 5%. This should be achieved 5 weeks before the project deadline.

3.1.2.4 Objective 2D

Employees will be able to scan their own RFID card to perform admin actions such as removing purchases and editing customer accounts. This should be achieved 6 weeks before the project deadline

3.1.3 Aim 3

To improve the business insight into the running of zero-waste shops by implementing an InfluxDB dashboard, which recieves information from the dispensing system. Displaying real-time stock levels, and shopping trends of customers within $\pm 10\%$ accuracy. The business insights should improve inventory management and the efficiency of employees by at least 10%.

3.1.3.1 Objective 3A

Data from the dispensing system will be converted into a real-time dashboard, updating after every dispense, displaying stock levels and business trends within ±10% accuracy using InfluxDB. This should be achieved 4 weeks before the project deadline.

3.1.3.2 Objective 3B

The dashboard will be easy to read and provide actionable insights, including notifications to employees of the lowest capacities, increasing employee efficiency by at least 10%. This should be achieved 4 weeks before the project deadline.

3.2 Chapter Outline

3.2.1 Chapter 1: Literature Review

The literature review details the identification of key issues to consider within my project, the benefits of automation, possible requirements of my system, and the technologies I can employ to capture these requirements.

3.2.2 Chapter 2: Requirements

The requirements chapter lists the functional and non-functional requirements of my project, alongside the literature which informed them, ranked using MoSCoW prioritisation.

3.2.3 Chapter 3: Methodology

The methodology chapter details the methodology I followed during development, the project schedule, and considerations which could delay the project.

3.2.4 Chapter 4: Design

The design chapter contains the design diagrams I created for my project, including a discussion and justification of the design choices.

3.2.5 Chapter 5: Implementation

The implementation chapter discusses the key features of my implementation, including challenges faced and details of testing.

3.2.6 Chapter 6: Project Evaluation

The project evaluation chapter includes my personal evaluation and reflection of the project.

3.2.7 Chapter 7: Further Work and Conclusions

The further work and conclusions chapter provides suggestions to improve the project in the future, alongside my final conclusion of the project.

4 Literature Review

4.1 Introduction

The following literature review explores current research regarding zero-waste shops, the overall retail sector, automated dispensing systems, and relevant hardware/software technologies in those areas. The aim of the review is to recognise the challenges and points of improvement for these areas and how they might apply to my own project work.

4.2 Review of Zero-Waste Shops and Retail

A good definition for a zero-waste shop is from **Re-Gen Waste (2023)**[40], "a zero-waste shop enables customers to live a more zero-waste lifestyle through eliminating packaging and encouraging the use of containers from home to fill and refill with bulk wholefoods, natural beauty and cleaning products plus much more". This outlines the type of products my system will look to dispense, suggesting the system will have to handle both solid and liquid products. **Kumar (2023)**[27] outlines the values of zero-waste shops and highlights my main inspirations for the choosing of my topic such as reducing plastic waste, supporting local producers and promoting sustainable practices.

Mack (2024)[28] discusses the flow of the average customer in a zero-waste

shop, you weigh each of your containers before filling them up, this is done either at the shop's till or a central scale in the store. Immediately, this appears to be somewhat convoluted, for example, if you were to fill a container halfway you would need to weigh it once again before filling it up with a different product. The weighing being done at a central location will also create congestion among customers as they wait for the till/scales. I hope to address the tedium of this in my own project by automating the weighing at the point of dispensing and calculating the difference, reducing the need for a central location that a customer might require multiple interactions with and streamlining the shopping experience. **Mack (2024)**[28] also discusses filling your containers, which are currently filled manually. **Somers (2024)**[45] considers spillages to be a considerable challenge across zero-waste stores. In my system I hope to minimise spillages, to do this I will rely on automated dispensing to reduce human input and consequently the likelihood of human error.

Somers (2024)[45] highlights a limitation of my project. Containers used by zero-waste customers often require a lot more resources than their single-use plastic counterpart, meaning more uses are required to see benefits. Solving this is out of scope for my system, however, a quality system would hopefully maintain a customer base and eventually result in a net-positive over single-use plastic. Price of goods is often mentioned when discussing zero-waste shops **(Somers, 2024)**[45], with some of the shops themselves admitting to products being priced higher **(Jungle Culture, 2019)**[20], this is again something I hope to address in my own system. A successful system should result in lower operating costs due to reduced labour and increased efficiency of customer checkout, hopefully achieved though streamlining the dispensing process and handling payments quickly.

I believe research shows bigger companies are not against adopting zero-waste ideas if the market is there and running costs are sustainable, a huge example of bigger corporations adopting zero-waste can be seen in the success of the Too Good to Go app, with Aldi, Greggs and more having a presence on the app **(Too Good to Go, 2024)**[48]. I trust the ease of using the Too Good to Go service and the benefits of promoting sustainable practice are the key drivers of its success, if a zero-waste dispensing system can be easily managed and maintained it could convince bigger corporations to research and adopt zero-waste at sections of their store as current technologies require too much admin, for example the frequent spillages **Somers (2024)**[45]. This can also benefit businesses as being sustainable gives them an advantage against competitors and improves public perception **(Townley, 2023)**[49].

"91% of retailers believe self-checkout contributes to higher sales as it is preferred by 71% of customers" **(Holden, 2024)**[16] and 96% of grocery stores offer a self-checkout **(Capital One Shopping, 2024)**[6]. These statistics suggest self-checkout not only promotes a business's profits, but it is almost an expectation for stores to include a self-checkout option due to their ubiquity. Providing a

consistent customer experience that is as expected makes it easier for customers to checkout, and I would hypothesise the familiarity is also comforting.

A major drawback of self-checkout is job displacement due to the reduced costs and management compared to traditional checkouts. To put this into perspective, in 2019 75,000 retail jobs were lost to automation **(Khan, 2023)**[24], and head of logistics at Primark estimates that 60-65% of retail will be automated by 2025 **(Zhadan, 2023)**[52]. **Nesbo (2023)**[34] disputes the effect of self-checkout on jobs, suggesting those whose jobs are at risk are simply offered a different position, in the context of my project this would allow zero-waste staff to focus more on stock replenishment rather than customer checkout. Nevertheless, it is still an important ethical consideration if people's livelihoods could be at risk and a key consideration for the negatives of my system. Installed technologies between zero-waste shops and supermarkets highlight the division in automation, perhaps this is understandable due to the difference in scale, but it leaves a gap for a cheap and effective alternative which I hope my project can provide.

Whilst researching retail technologies, I discovered that most supermarkets have little intention of keeping the option for cash payments at self-checkouts, with well-known brands such as Waitrose only accepting card on their self-service checkouts **(Hougton, 2024)**[17]. Following current trends, accepting cash payments seems low in priority for my system as society increasingly shifts towards being cashless. Research suggests going cashless is beneficial to the environment **(Rochemont, 2018)**[41], which aligns with the core benefits of the zero-waste business model. Unfortunately, it is also suggested an increase in cashless transactions correlates with an increase in cyber-attacks **(Mauladi, Laut Mertha Jaya and Esquivias, M.A., 2022)**[29], perhaps this could be due to COVID rather than being cashless specifically as more people work from home, increasing their vulnerability to internet scams such as phishing **(Deloitte, 2020)**[10].

4.3 Review of Automated Dispensing Systems

The core of the project is dispensing. **Wang et al. (2022)**[51] and **Alanazi et al. (2022)**[1] outline their views on an automated dispensing system in pharmacies. Research suggests 5-6% of hospital admissions are linked with medication error and the rate of medication error sits at 0.15%. **Wang et al. (2022)**[51] states automated dispensing greatly improved efficiency of drug distribution and reduced the error rates, beyond that, automated dispensing also gives pharmacists time to review the medication before dispensing and increases patient satisfaction **(Alanazi et al., 2022)**[1].

Fung et al. (2009)[14] argues a lack of evidence to claim a strong benefit of automated systems and highlights issues in studies, such as inconsistent definitions of outcomes across studies, or evidence suggesting more errors are

seen in the prescribing stage. Also mentioned is the major drawback of cost preventing more widespread adoption. Whilst this is specific to the medical field, the advantages and disadvantages seen are often shared across automated dispensing systems, for instance **Medrano-Galarza et al. (2017)**[30] studies automatic milk feeding in cows and concludes benefits of reduced labour and better outcomes whilst also having the barrier of cost.

These studies help identify overall aims for my dispensing system, the advantages almost directly translate to my own project and the disadvantages suggest what I should avoid. With high cost commonly scrutinized, I will focus on sourcing the best value components. I agree with much of the literature, despite **Fung et al. (2009)**'s[14] warnings regarding evidence, arguments I've observed appear drawn from exhaustive research. Admittedly, I found parts of the research of **Alanazi et al. (2022)**[1] supported the thesis of **Fung et al. (2009)**[14]. For example, an observational study involving just two hospitals is very similar to the studies Fung et al. (2009) critiques. I believe these concerns are outweighed by multiple other sources having different studies but similar conclusions **(Wang et al, 2022)**[51] **(Medrano-Galarza et al., 2017)**[30], the recency of these studies relative to **Fung et al. (2009)**'s [14] thesis suggests newer studies have acknowledged the previous gaps in research. Perhaps an accepted way to measure the success of these machines would further improve my confidence in the research and make results feel more tangible.

4.4 Review of Relevant Hardware Projects and Technologies

Existing automation projects, such as an automated water dispenser **(Kavya et al., 2021)**[22], automated weight and height measurement **(Ulyanida et al., 2022)**[50], an IoT automated pet food dispenser **(Kondapalli et al., 2019)**[25], an IoT automated agriculture robot **(Kulothungan, Kamalakannan and Thirugnanam, 2018)**[26], and an IoT medicine dispenser **(Philip et al., 2020)**[37], demonstrate a variety of automated systems relevant to my own goals. Key considerations across these projects are efficient automation, accessibility, cost effectiveness and reliability, all of which are principles I aim to incorporate into the design of my own project.

The water dispenser is particularly relevant and demonstrates the advantages of automation and using contactless technology to reduce water wastage, qualities which would be priceless to a zero-waste shop. Unlike my own goals, this project does not automatically dispense the correct amount based on the container, even suggesting the addition of more sensors in future work. Nonetheless, it gives me a good basis on what hardware I might use to begin my project as it demonstrates a low cost and reliable solution. My project will hope to fill the gaps in this project, not just with dispensing amounts but also with the

addition of IoT technology.

To bridge these gaps, I might consider the IoT approach of **Philip et al. (2020)**[37]. The use of sensors in both **Philip et al. (2020)**[37] and **Ulyanida et al. (2022)**'s[50] projects inspire ideas for my own, a similar IoT and sensor implementation could be adapted to remotely track stock levels, prices and peak hours for valuable business insights at a zero-waste shop. **Ulyanida et al. (2022)**'s [50] simple but effective Arduino based weight and height measurement system provides a potential base to build my own implementation. I may encounter a limitation in developing a User Interface (UI) that is both aesthetically pleasing and accessible, the UI seen in **Philip et al. (2020)**[37] and **Kondapalli et al., (2019)**[25], whilst effective, is very basic and lacks proficient styling. My project requires a diverse range of customers, a basic UI might defer customers from interacting with the system, especially if it will be interacting with sensitive data such as payment details. I would need to consider more universally friendly design in my own work.

Kulothungan, Kamalakannan and Thirugnanam (2018)[26] suggest an IoT approach could be more efficient than a human effort in agriculture. This implies a reduction of labour costs and if I can extend this principle to the context of zero-waste shops, they should see similar benefits for adopting the dispensing system. Research by **Elijah et al. (2023)**[13] contradicts the advantages delivered by **Kulothungan, Kamalakannan and Thirugnanam (2018)**[26], believing there are big issues with cost and security. Following work completed by **Kavya et al. (2021)**[22], I have confidence I can remain cost effective. Security could prove difficult regarding the taking of payment, by separating this from the hardware of my project I hope to mitigate risk.

Improvements to my project could follow **Jia et al. (2012)**'s[19] research on use cases of RFID in IoT, particularly its capacity to reduce costs and improve the efficiency of existing systems. Extensive evaluations of RFID **(Costa et al., 2021)**[8] **(Munoz-Ausecha, Ruiz-Rosero and Ramirez-Gonzalez, 2021)**[32] are overall positive and highlight the adaptability of the technology. RFID's usage spans multiple sectors, from industrial **(Zhai et al., 2016)**[53] to healthcare **(Jara, Zamora and Skarmeta, 2011)**[18], implying RFID can be adapted for use in my own work. For example, RFID tags could provide tracking for products being dispensed by customers, adding another dimension to enhance the previously discussed systems within zero-waste shops. If each customer had their own tag, you could implement a rewards system to encourage customer loyalty and improve customer retention, which also sees success in targeted marketing campaigns, as reward systems build up a reliable mailing list **(Smith, 2010)**[44]. RFID's aptness in abstracting complex processes from the customer is highly desirable, requiring just a single, simple action to perform a wide range of tasks, as demonstrated by examples listed in **Fig. 1** and projects mentioned above.

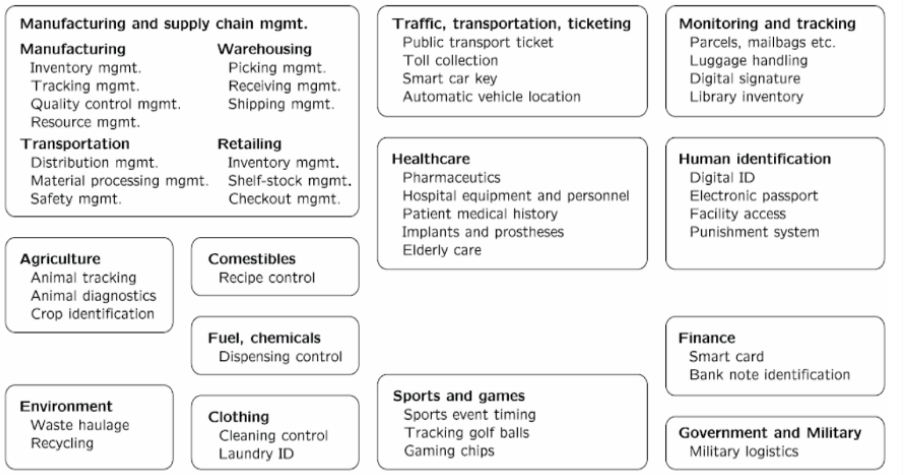


Figure 1: Applications of RFID **(Kaur et al., 2011)**[21]

The general concern of RFID is security, and the benefits are its versatility, ease of use and low cost, especially over alternatives such as sensor-equipped Wi-Fi nodes **(Costa et al., 2021)**[8]. To mitigate security concerns, I will separate my RFID implementation from both payment processing and sensitive data storage. Despite this precaution, RFID has other limitations, for example, the issue of signal collision. This is especially concerning in a confined space with many tags, such as a zero-waste shop. Anti-collision algorithms can fix the issue but incur extra cost **(Kaur et al. 2011)**[21], consequently, their benefit would have to be evaluated for my system. **Munoz-Ausecha, Ruiz-Rosero and Ramirez-Gonzalez (2021)**[32] believe RFID could bring IoT to the point of "every existing object" being able to connect to the internet. The promising future of RFID future-proofs my design and highlights its value as a suitable component in my smart dispensing system.

Baldini, Chessa and Brogi (2023)[4] believe IoT has "huge innovation potential, but at the same time, its spread can induce one of the highest environmental impacts", the consequence of this is a rise in interest for green IoT deployments. **Shapsough et al. (2021)**[42] uses IoT to optimise a renewable energy source, **Fig. 2** showcases the technology stack used to achieve this. Their results were highly successful and presented energy efficiency, the project can take heavy inspiration from this flow of data to leverage its efficiency. The project also highlights efforts to achieve green IoT solutions and demonstrates how large-scale IoT can be sustainable.

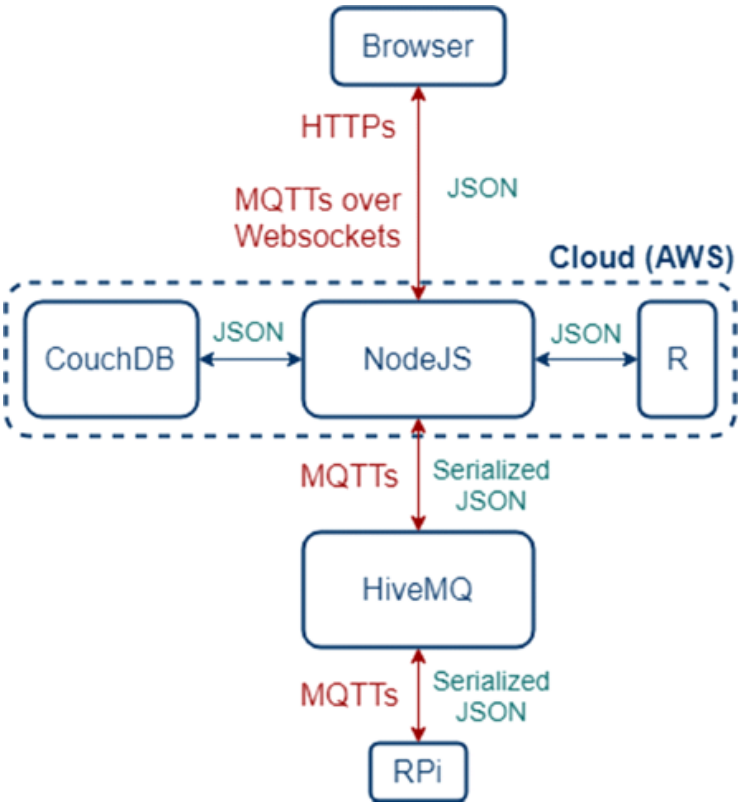


Figure 2: Flow of data from **Shapsough et al. (2021)**[42]

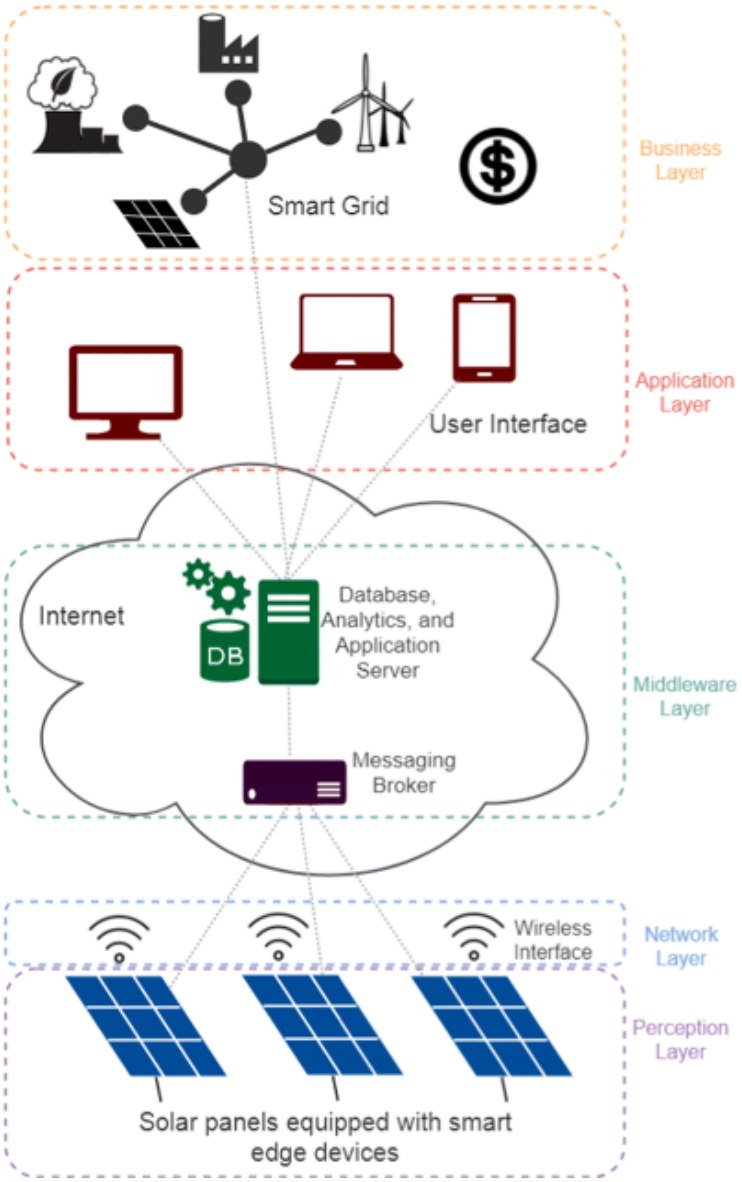


Figure 3: IoT project system architecture from **Shapsough et al. (2021)**[42]

The information from these sources inspires my hardware project's design and implementation. The circuit diagrams and technical specifications will help with my own wiring, diagrams, and hardware decisions. The literature also provides insight into possible limitations of IoT systems, for example, in the pet feeder project, time appears to have been a major constraint as the future work left

"a lot more things to improve" **(Kondapalli et al., 2019)**[25]. It also appears an accessible and aesthetically pleasing UI for these IoT projects is difficult to achieve. The UI in both the pet feeder and the medicine dispenser, whilst functional, was very basic. An unattractive UI will dissuade both customers and potential adopters of the system, appropriate software technologies will have to be considered. Time is a major constraint on my project, therefore I will adopt an Agile methodology to combat this and help maintain consistent progress.

4.5 Review of Relevant Software Technologies

For the checkout application of my system, I intend to use a Python based web application, Python has many options for frameworks that will be more than suitable for my needs. **Ghimire (2020)**[15] researches the two most popular python frameworks and their validity for an e-commerce app. The study found Flask provided simplicity and flexibility whilst Django best fit large-scale applications, the comparison utilised an SQLite database, which I believe also fits the needs of my web application. I consider Flask to be the most suitable for my project as it is a middle ground between the simplicity of Bottle **(Bottle, 2024)**[5] and the complexity of Django. **Morchid et al. (2024)**[31] demonstrates a successful embedded systems project utilising a RaspberryPi3 with Flask to detect fires, its structure can help with my own project architecture regarding the technologies I might need **(Fig. 4, 5 & 6)**. This project also proves Flask is lightweight enough to run on minimal hardware, exhibiting a responsive web application which could potentially outperform software found on supermarket self-checkouts, as in my experience it is often sluggish.

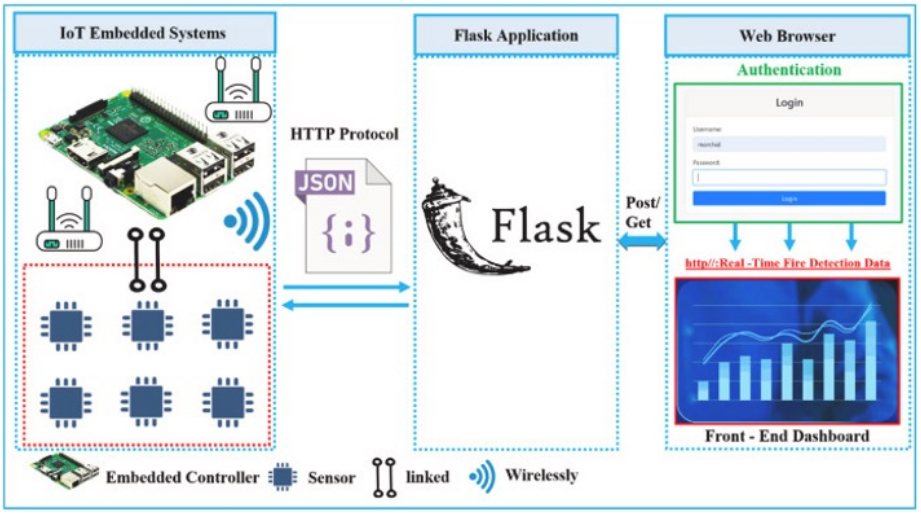


Figure 4: Framework of **Morchid et al. (2024)**[31] IoT project

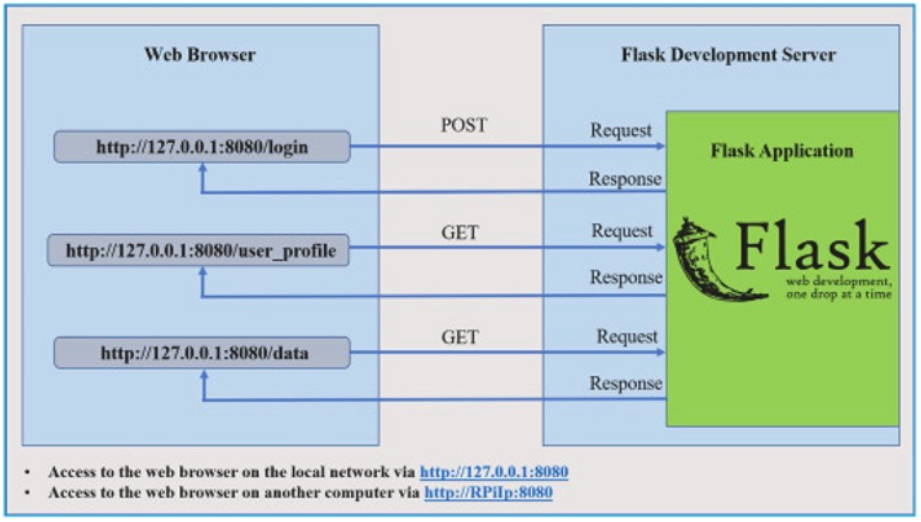


Figure 5: Process between Flask and web browser from **Morchid et al. (2024)**[31]

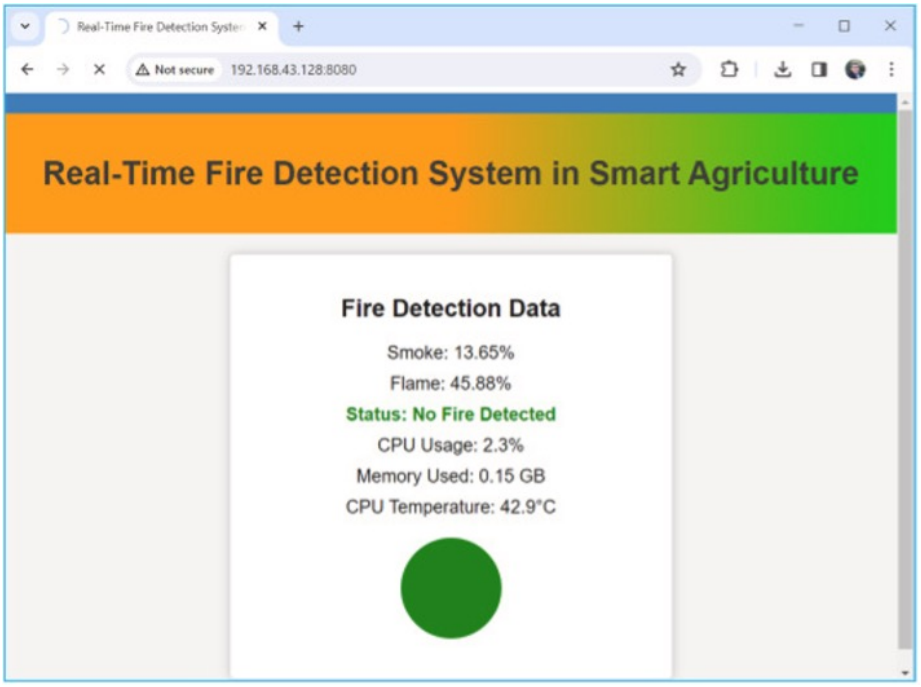


Figure 6: Flask displaying fire detection data in real time in the web browser **(Morchid et al., 2024)**[31]

The system will be required to process payments, studies by **Do et al. (2022)**[11] leads me to consider external payment processing as a viable solution due to the huge number of security concerns, developing my own would be both time-consuming and a serious security risk. **Kemenes (2024)**[23] suggests Stripe as a secure option, albeit whilst working for a company offering services that integrate with Wise, it does still provide a good comparison between Wise and PayPal and their key differences **(Fig. 7)**.

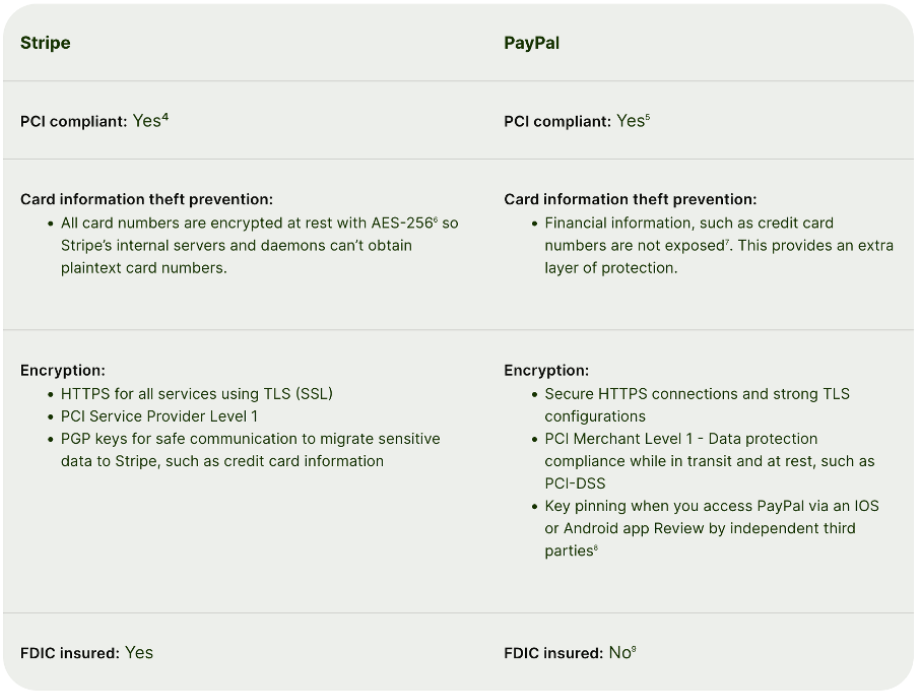


Figure 7: Comparison between Stripe and Paypal by **Kemenes (2024)**[23]

Niranjanamurthy, M. (2024)[35] believes Stripe to be a "good payment solution for web developers" but does incur higher fees than those of PayPal. Stripe does not require a user sign-up to accept payments, its API is also compatible with many libraries, including Flask. Stripe is widely implemented and proven in industry, with its clients including DoorDash and Shopify **(Stripe, 2024)**[46]. There is a limitation of my project in that I do not have access to the pre-existing payment system a zero-waste shop might employ, I believe Stripe to be the best option as a placeholder as it will work universally as a standalone software so no hardware should be required, Stripe is also PCI compliant **(Kemenes, 2024)**[23], this is extremely important for the ethical handling of customer payment details and failure to comply can have disastrous consequences for a business.

4.6 Overview

My research has helped me to identify requirements and gaps my project can fill, notably, reducing spillages through automated dispensing, beautifying UI

to be customer facing with the Flask framework, enhancing current microcontroller based liquid dispensers by implementing IoT, and optimising the flow of shopping within a zero-waste shop through automatic weighing and RFID powered self-checkout.

5 Requirements

5.1 Introduction

For my requirements I will be using MoSCow prioritization, defined in **Fig. 8**. This approach provides more structure to my requirements while ensuring effective weighting of importance, consequently reducing the risk of an inoperative project. I have also included the key sources from my background research that helped shape each specific requirement.

Figure 8: A breakdown of the MoSCoW acronym and description of each category **(Techtarget, 2023)**[47]

5.2 Functional Requirements

5.2.1 Functional Requirements: Must

Table 1: Must Functional Requirements

ID	**Requirement Description**	**Relevant Literature**
MF1	The dispensing system will improve the process of dispensing products to help reduce checkout times and spillages in zero-waste shops.	• Somers (2024) [45] • Ulyanida et al. (2022) [50] • Mack (2024) [28] • Kavya et al. (2021) [22]
MF2	The dispensing system will communicate with the checkout system and display details of a customer's order based on their RFID tag's Unique Identifier (UID).	• Jia et al. (2012) [19] • Munoz-Ausecha, Ruiz-Rosero and Ramirez-Gonzalez (2021) [32]
MF3	Checkout system will be able to process payments/refunds and provide customers with proof of purchase.	• Stripe (2024) [46] • Do et al. (2022) [11]
MF4	Dispensing system will accurately monitor the stock levels of products and notify employees when stock levels are low.	• Morchid et al. (2024) [31] • Shapsough et al. (2021) [42]

ID	Requirement Description	Relevant Literature
MF5	Dispensing system will send real-time dispense data to be displayed via a business-facing dashboard.	• Morchid et al. (2024) [31] • Shapsough et al. (2021) [42] • Somers (2024) [45]
MF6	Dispensing system will be able to handle a reasonable variety of containers.	• Mack (2024) [28]
MF7	At checkout customers can add predefined items and custom dispenses.	• Holden (2024) [16] • Nesbo (2023) [34]
MF8	Customers will be able to register using their name and email to attach it to their RFID tag. This will allow receipts to be sent to registered emails, alongisde a reward points system. The account email is stored in the database for future communications.	• Smith (2010) [44]
MF9	Employees will be able to cancel dispenses, remove individual items, authorise refunds, and modify customer accounts.	• Holden (2024) [16] • Nesbo (2023) [34]

5.2.2 Functional Requirements: Should

Table 2: Should Functional Requirements

ID	Requirement Description	Relevant Literature
SF1	The cost and product name attached to a dispenser should be changeable by employees using the checkout system.	• Wang et al. (2022) [51] • Alanazi et al. (2022) [1]
SF2	The dispensing system should communicate issues to staff and customers via a display.	• Kavya et al. (2021) [22]
SF3	Checkout system should incorporate a rewards system where each £1 spent earns 1 loyalty point, which is equal to 1 penny, helping improve customer retention.	• Smith (2010) [44]

5.2.3 Functional Requirements: Could

Table 3: Could Functional Requirements

ID	Requirement Description	Relevant Literature
CF1	The system could include voice guidance to improve accessibility.	• Holden (2024) [16]
CF2	The checkout system could display an estimate of waste reduced to the customer at checkout based on their purchases.	• Somers (2024) [45] • Jungle Culture (2019) [20]

ID	Requirement Description	Relevant Literature
CF3	Both systems could apply anti-collision algorithms to prevent RFID tag collisions.	• Kaur et al. (2011) [21]

5.2.4 Functional Requirements: Wont

Table 4: Wont Functional Requirements

ID	Requirement Description	Relevant Literature
WF1	Checkout system will not accept cash payments.	• Houghton (2024) [17] • Rochemont (2018) [41]
WF2	Dispensing system will not validate customer containers, the expectation being customers will bring acceptable containers.	• Somers (2024) [45] • Kumar (2023) [27]

5.3 Non-Functional Requirements

5.3.1 Non-Functional Requirements: Must

Table 5: Must Non-Functional Requirements

ID	Requirement Description	Relevant Literature
MNF1	The system must be scalable enough to handle multiple dispensers connecting to one checkout system.	• Jia et al. (2012) [19] • Holden (2024) [16] • Shapsough et al. (2021) [42]
MNF2	Customer payments must be handled securely and be PCI compliant.	• Do et al. (2022) [11] • Kemenes (2024) [23]
MNF3	The system must have an uptime of approximately 99% during business hours.	• Morchid et al. (2024) [31]
MNF4	The dispensing system must have minimal error in its measurements, with around a 10% error margin.	• Kavya et al. (2021) [22] • Ulyanida et al. (2022) [50]

5.3.2 Non-Functional Requirements: Should

Table 6: Should Non-Functional Requirements

ID	Requirement Description	Relevant Literature
SNF1	The dispensing system should be cost effective by using low-cost components.	• Kavya et al. (2021) [22] • Medrano-Galarza et al. (2017) [30]
SNF2	The dispensing system's individual components should be replaceable for easy maintenance.	• Wang et al. (2022) [51]

5.3.3 Non-Functional Requirements: Could

Table 7: Could Non-Functional Requirements

ID	Requirement Description	Relevant Literature
CNF1	The system could use a green IoT approach to reduce its energy usage and footprint.	• Baldini, Chessa and Brogi (2023) [4] • Shapsough et al. (2021) [42]
CNF2	The checkout system could have personalisation options to allow it to reflect the branding of different stores, such as colour schemes.	• Philip et al. (2020) [37] • Morchid et al. (2024) [31]

ID	Requirement Description	Relevant Literature
CNF3	The system could use minimal power by operating in sleep mode outside of business hours.	• Baldini, Chessa and Brogi (2023) [4] • Shapsough et al. (2021) [42]

5.3.4 Non-Functional Requirements: Wont

Table 8: Wont Non-Functional Requirements

ID	Requirement Description	Relevant Literature
WNF1	The system will not use overly expensive components to maintain affordability for zero-waste stores.	• Medrano-Galarza et al. (2017) [30] • Kavya et al. (2021) [22]
WNF2	The system will not be able to perform offline due to the use of IoT and Wi-Fi communications.	• Shapsough et al. (2021) [42] • Morchid et al. (2024) [31]

6 Methodology

6.1 Agile Methodology

Aligning with the conclusion in my literature review, I will use SCRUM Agile methodology. The flexibility of SCRUM eases the worries of integrating new

technologies in my project as it allows for flexibility and adaptation to limitations, which is particularly important considering my very limited experience in hardware/IoT technology.

While Agile is designed for collaboration, I am developing the project alone. I believe even in solo development, Agile realises its key benefits, for example the efficient breaking down of implementation tasks, and effective feedback from stakeholders, notably the project supervisor. I held weekly stand-ups with myself, in which I reviewed the project's progress and key issues which are halting progress, the stand-up results guided the week's development.

6.2 Project Management

The project will use GitHub for version control. This is essential in large projects, providing consistent back-ups, easy access over multiple devices, and a historical record of development.

Fig. 9 and **Appendix A, Figure 1** illustrate the comprehensive project schedule. This is intended to indicate whether the project is progressing as planned for completion, and provide an overall structure to development. Additionally, I developed a risk register **(Appendix A, Figure 2)** to list contingencies that may impede my compliance with the Gantt chart schedule and mitigate the probability of project delays.

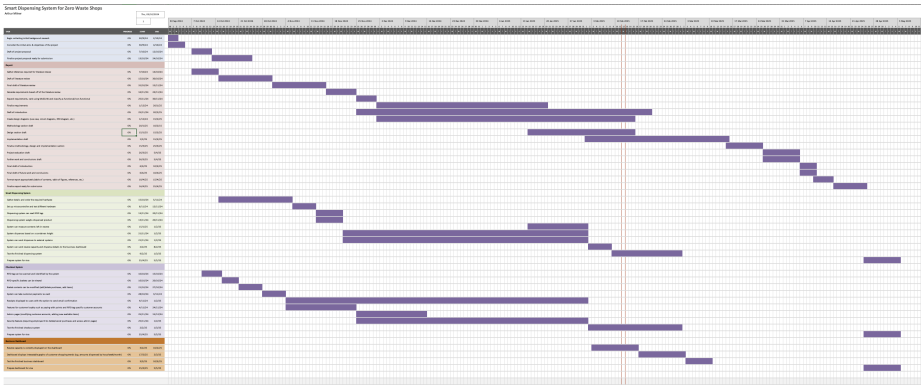


Figure 9: Gantt chart with the project at 0% completion

7 Design

7.1 Initial Planning

Initially, prior to submitting the project proposal, I conceptualised potential solutions for my topic and its operation framework **(Appendix B, Figure 1)**, adopting a high level approach to evaluate the prospective final solution and its feasibility. This phase involved broadening the project to function more as a shop management system rather than solely concentrating on dispensing, incorporating features such as a self-checkout and a business dashboard.

In designing my dispensing and self-checkout system, it was essential to consider the target users to create informed designs. Although the customer base tends towards the younger generations **(Profitable Venture, 2024)**[38], the system must be inclusive of all ages and demographics, as excluding certain groups may alienate potential customers. Consequently, my designs embody accessible practices, including minimal input requirements, straightforward dispensing gestures, contrasting colours at checkout, and prominent text and buttons. For the business dashboard, I deemed it essential to incorporate the viewpoint of zero-waste employees, focusing on the information they would find most beneficial, such as product capacities and daily sales revenue in relation to their targets.

I aimed for the dispensing system to be compatible with the existing containers typically used for product storage in retail environments to reduce the waste in replacing them, as evidenced in my designs. The liquid dispensing system necessitates only the incorporation of an ultrasonic sensor and pump tubing into the storage container for integration. The self-checkout system should, in practice, seamlessly integrate with any pre-existing payment system utilised by a zero-waste shop; however, for demonstration purposes, my designs employ the Stripe API for payment processing.

I considered the maintenance and updating of the system, advocating for a modular approach that allows for the replacement of malfunctioning sensors instead of the entire dispensing unit. In a practical scenario, the complete shop management package may be offered to stores through a subscription model, encompassing maintenance and updates.

Comparisons between hardware options were also drawn at this stage to decide on the most appropriate options for implementation **(Appendix B, Figures 2-5)**. Following these preliminary steps, I progressed to the development of high-level design diagrams.

7.2 High Level Diagrams

7.2.1 Self-Checkout Wireframes

For my self-checkout, design interactions are intended for a touchscreen, therefore, designs include large buttons and aim for minimal text input. A touch screen has the benefit of familiarity and ease-of-use, as it is the standard technology in self-checkouts, ensuring a minimal learning curve for new customers.

Figure 10: Self-checkout wireframe for the 'Home' page

Fig. 10 shows accessibility considerations by using large text and buttons, this is the page in which users will input or scan their RFID tag.

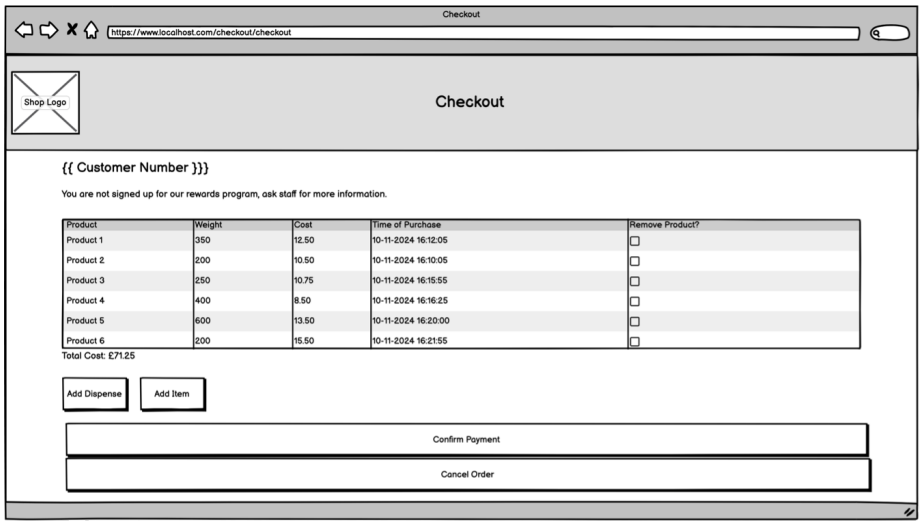


Figure 11: Self-checkout wireframe for the guest view of the 'Checkout' page

Fig. 11 shows the guest view of the checkout. Separate views were necessary as the account checkout presents the supplementary information and options. Customers can add or remove items to their basket, and confirm or cancel their order, subject to employee approval.

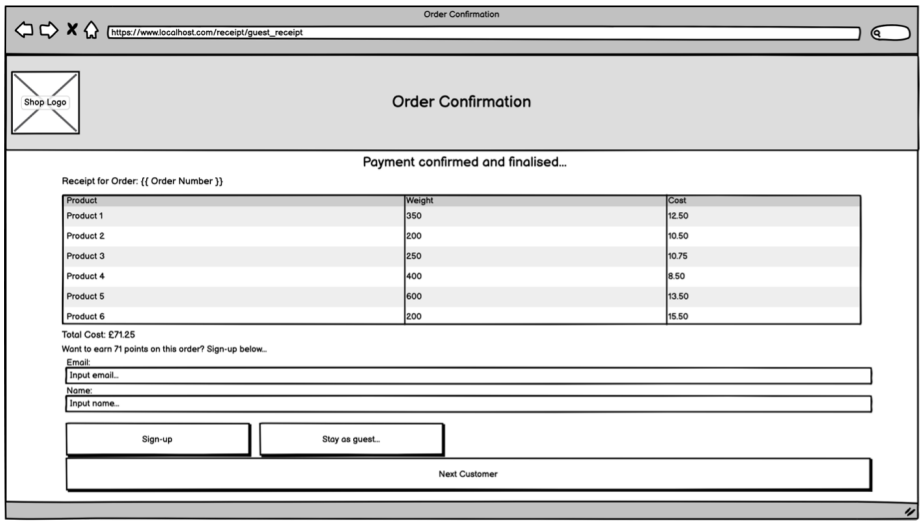


Figure 12: Self-checkout wireframe for guest view of the 'Receipt' page

Fig. 12 shows the receipt presented to guest customers, it shows the loyalty points they could earn on the order to entice them to sign-up, and provides the input fields to complete a sign-up.

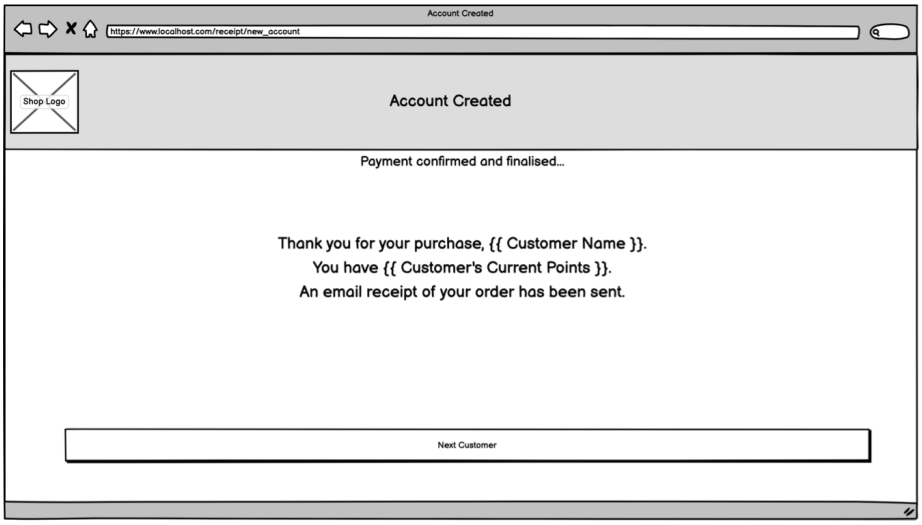


Figure 13: Self-checkout wireframe for the 'Account Created' page

Fig. 13 is presented to a newly registered customer, simply confirming the creation of their account and subsequent email receipt.

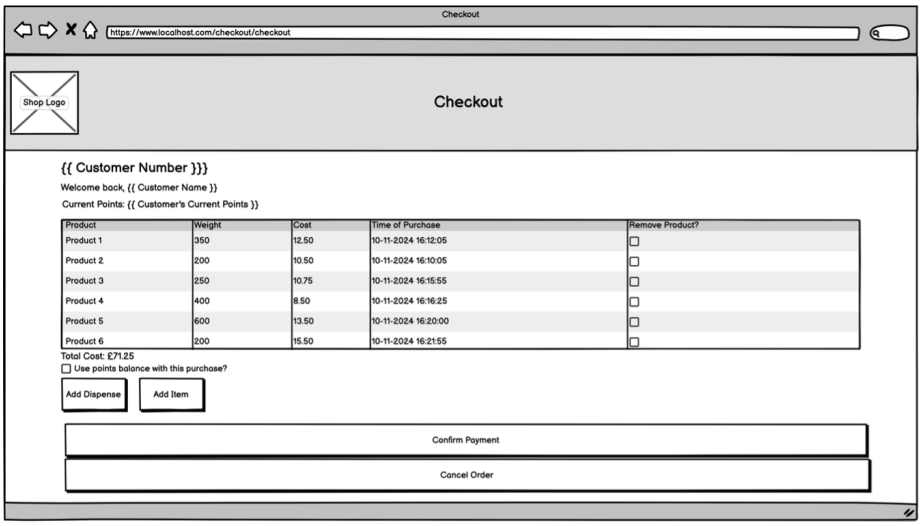


Figure 14: Self-checkout wireframe for the account view of the 'Checkout' page

Fig. 14 is the account checkout page, here the customer is greeted, and they can see the points attached to their account. There is also a check box to confirm whether the customer wishes to use their rewards points towards the order.

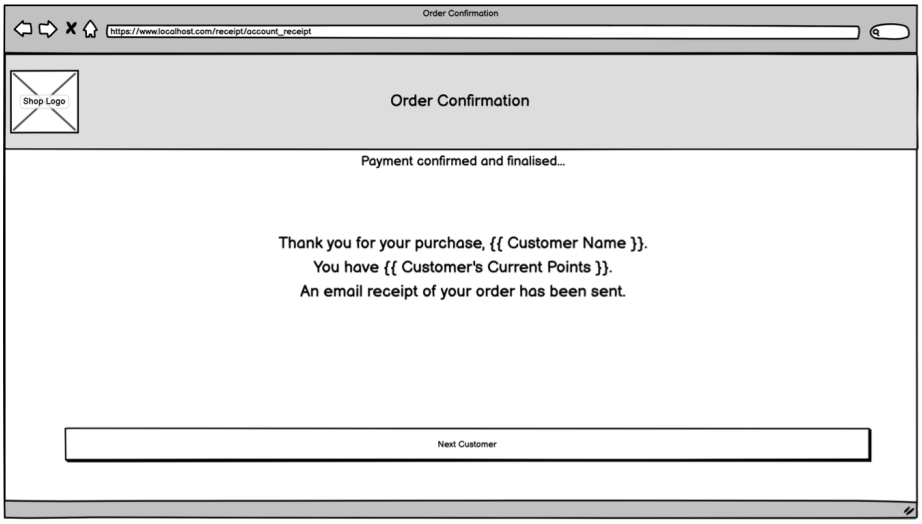


Figure 15: Self-checkout wireframe for the account view of the 'Receipt' page

Fig. 15 is the account receipt, no details have to be displayed as an email receipt will be automatically sent.

Figure 16: Self-checkout wireframe for the 'Refund Request' page

Fig. 16 is the wireframe for requesting a refund, customers will input the order ID on their receipt and the amount they wish to refund, keeping the UI as simple as possible to minimise errors and improve efficiency.

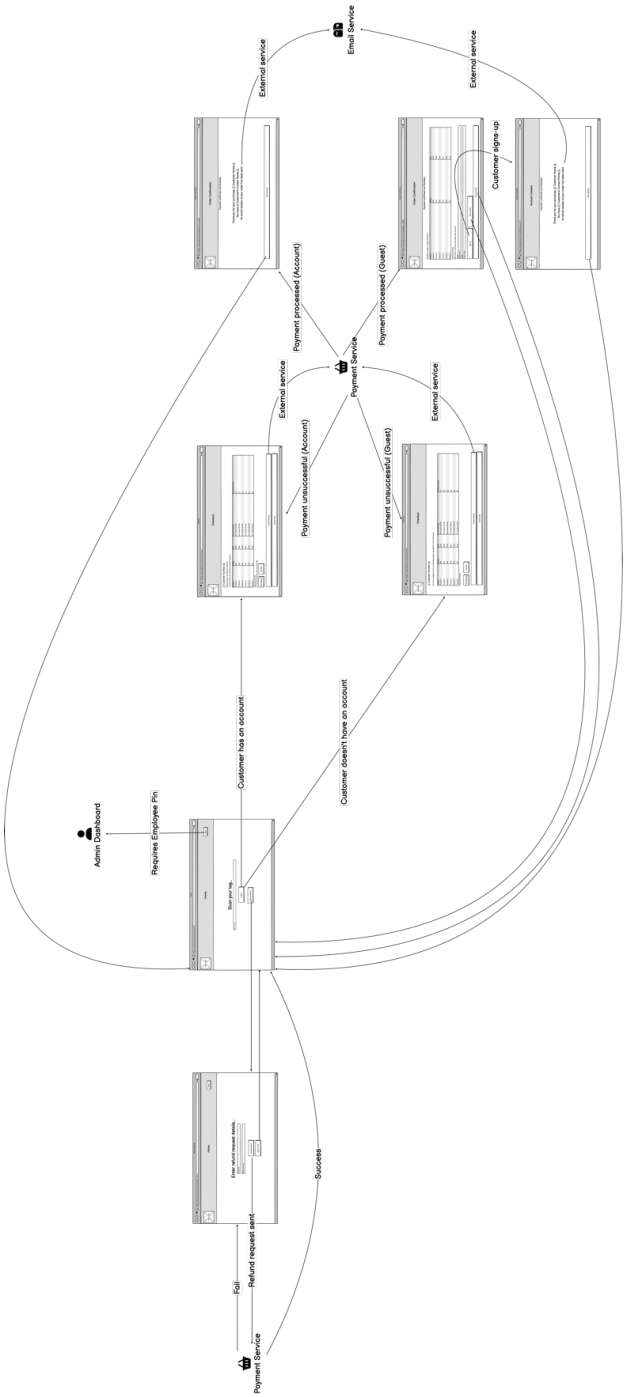


Figure 17: Wireflow diagram showing how the wireframes interact with each other

Fig. 17 suggests possible customer flows, including how the checkout could be integrated with external services, and security features for access control.

7.2.2 Self-Checkout Mock-Ups

Appendix B, Figure 6 contains mock-ups for the self-checkout system. These mock-ups serve as a detailed design reference, facilitating better evaluation of the checkout's accessibility regarding colours, font-size, and human-computer interaction prior to implementation. In a practical context, they would also be beneficial for soliciting feedback from interested zero-waste retailers.

7.2.3 Site Map

Fig. 18 shows the proposed site map for the final checkout system, encompassing both GET and POST routes as necessary. The administrative routes represent Flask-Admin, indicating the modifications employees can make from the admin panel. Flask-Admin encompasses all requisite functionalities for an administrative panel, while concurrently minimising the intricacies of implementation and testing.

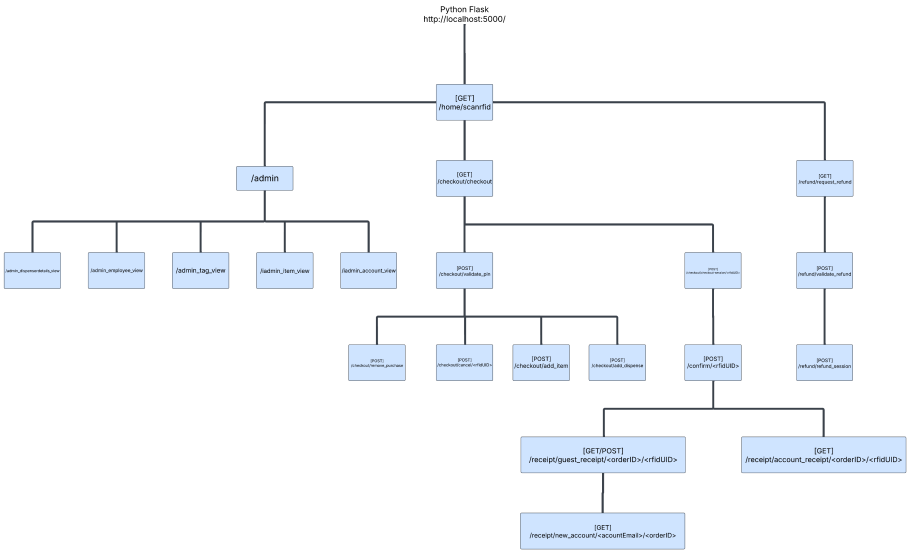


Figure 18: Site map design for the self-checkout

7.2.4 Dashboard Wireframes

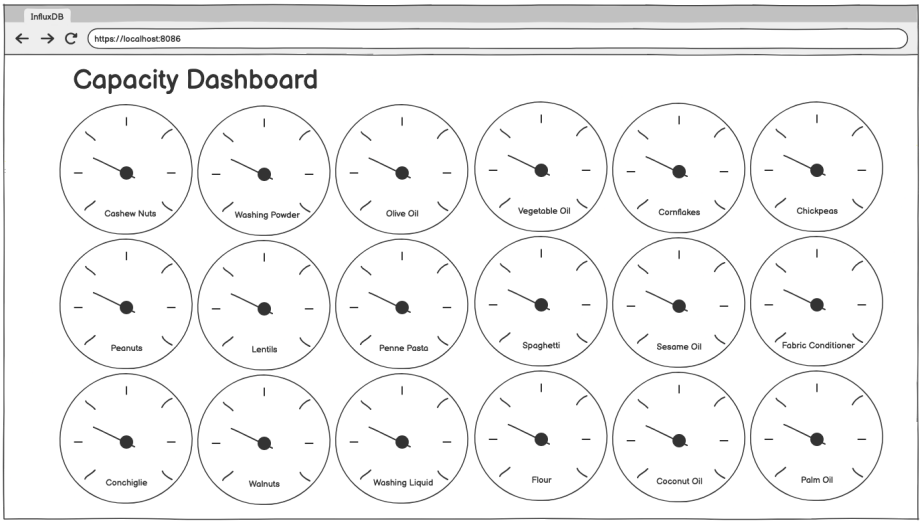


Figure 19: Wireframe for the 'Capacity' dashboard

For the capacity dashboard **(Fig. 19)**, the design was made to provide an at-a-glance look for as many products as possible, achieved using multiple gauges. In my final design the gauges will be colour coded dependent on the capacity.

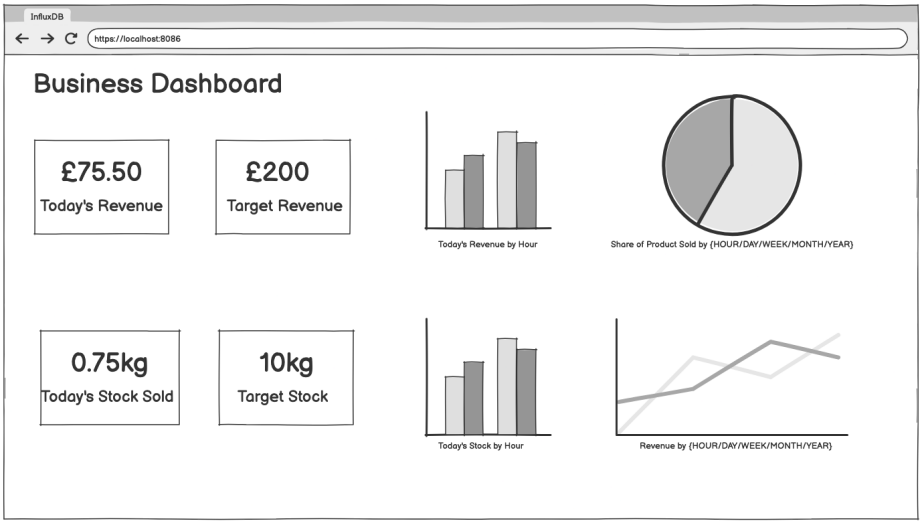


Figure 20: Wireframe for the 'Business Trends' dashboard

Fig. 20 showcases the main criteria to capture in my final dashboard, displaying current and target revenue/stock sold can provide motivation to improve sales and show current performance. The visualisations help identify peak times and provide a comparison of product performance, promoting effective staff allocation and informed product catalogue decisions.

7.2.5 Use Case Diagrams

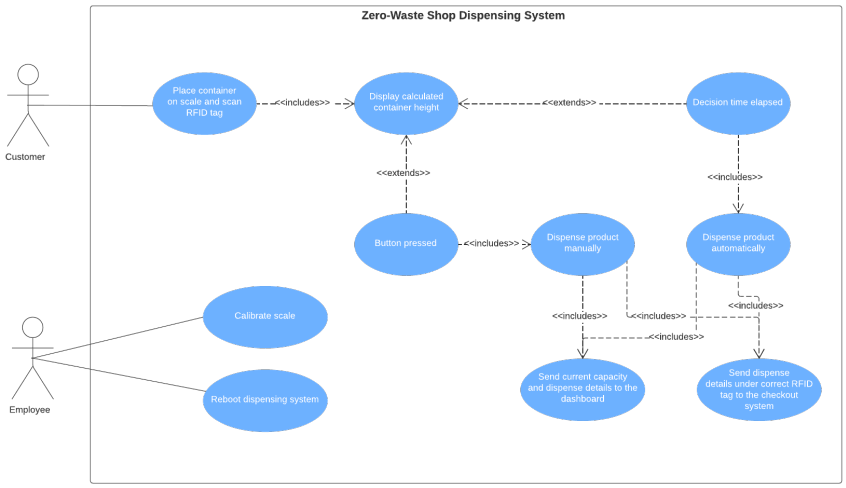


Figure 21: Use Case diagram for the dispensing system

Fig. 21 shows a simplified view of the dispensing flow and how customers will have the option for either manual or automatic dispensing.

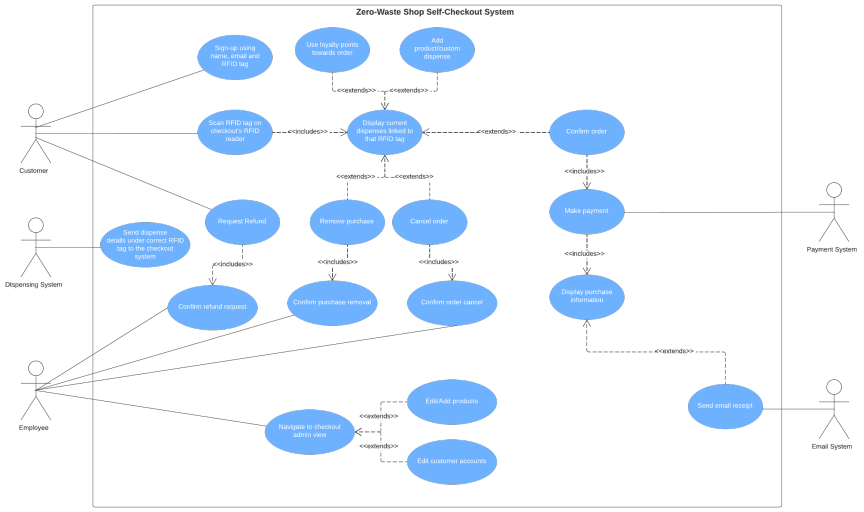


Figure 22: Use Case diagram for the self-checkout system

Fig. 22 shows how the self-checkout should receive data from the dispensing system, the many options for customers during checkout, and what options require what actors.

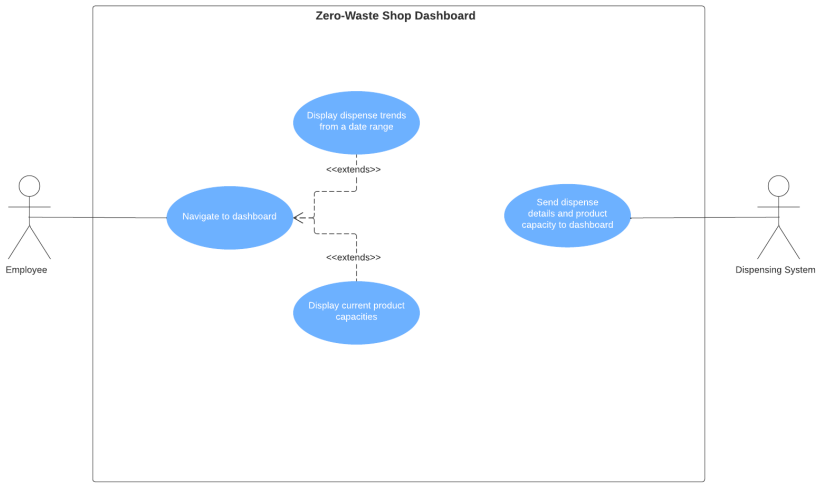


Figure 23: Use Case diagram for the dashboard system

7.2.6 Flow Diagram

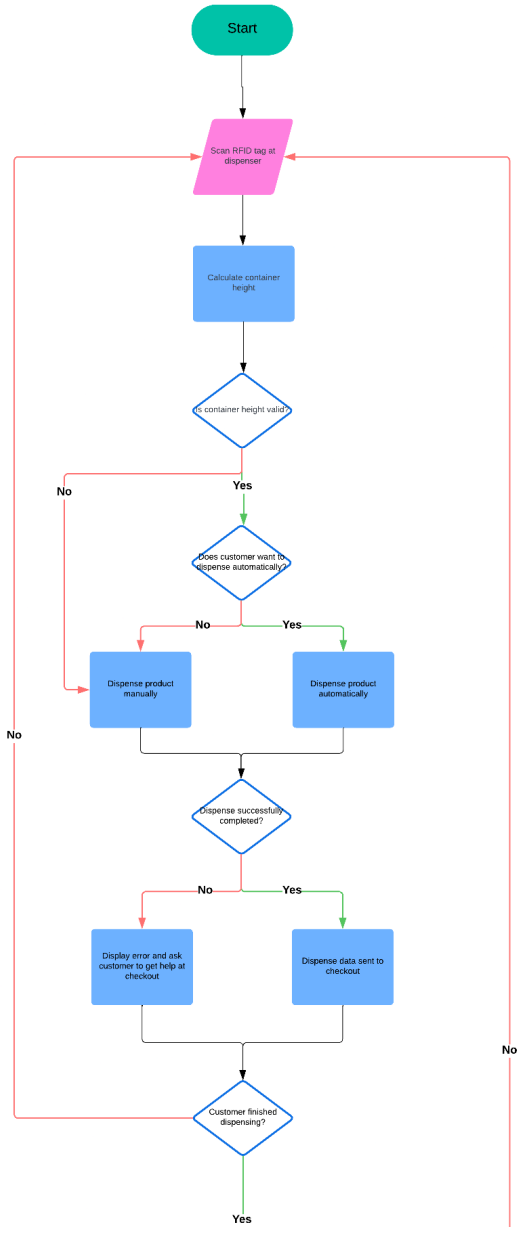


Figure 24: Customer flow diagram (1)

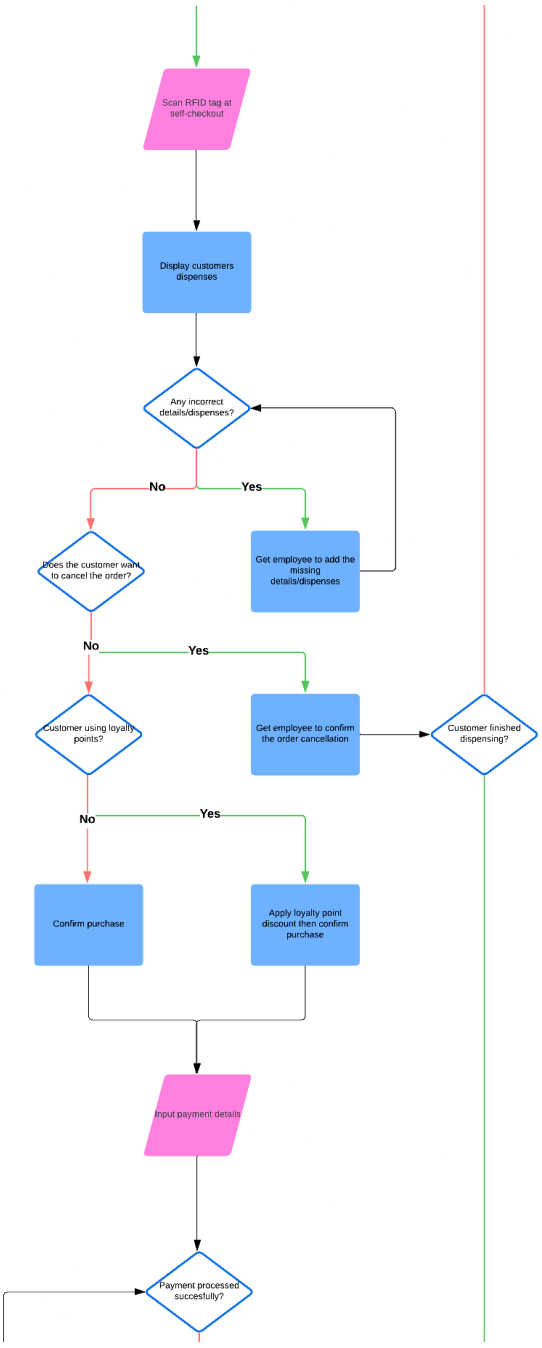


Figure 25: Customer flow diagram (2)

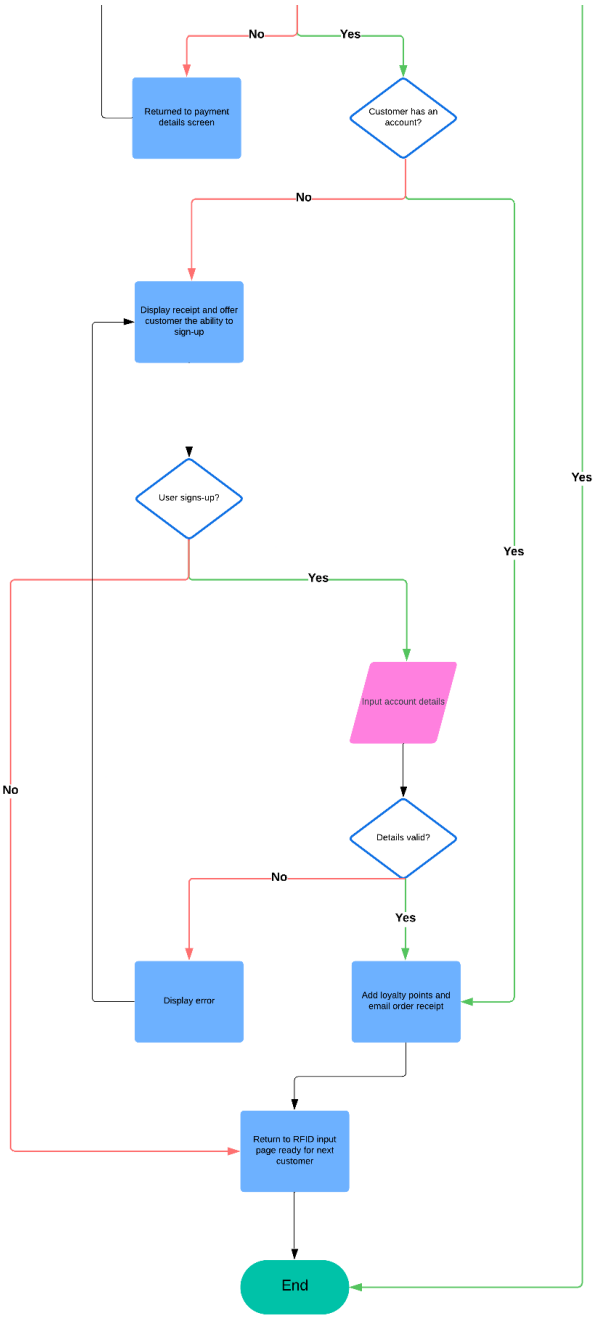


Figure 26: Customer flow diagram (3)

Figs. 24-26 cover the entire flow of a new customer from dispense to checkout, showing how certain flows are repeatable until the customer achieves the desired outcome, and the numerous decisions a customer can make to ensure the system handles many types of users.

7.2.7 System Architecture Diagram

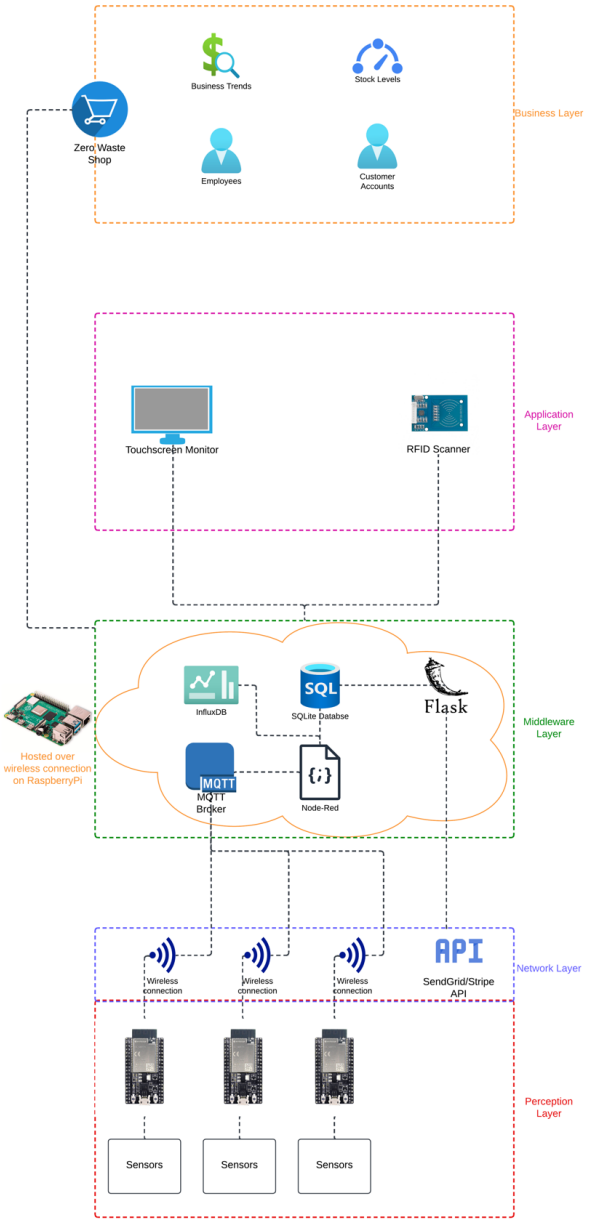


Figure 27: System architecture diagram for the entire project

Fig. 27 shows the interaction between the system and external systems, as well as existing technologies. MQTT offers a streamlined communication solution between the ESP32 and Raspberry Pi, while Node-RED significantly facilitates message management via MQTT, enables JavaScript Object Notation (JSON) parsing, and permits dynamic message routing based on topics. InfluxDB offers a time-series database and a Graphical User Interface (GUI) which can facilitate dashboard development. Ultimately, SQLite offers an adequate solution for prototyping the self-checkout system; however, deploying a MySQL server, for instance, would enhance scalability in a production environment. The diagram illustrates the system's scalability, depicting multiple ESP32 microcontrollers interfacing with a singular MQTT server.

7.3 Low Level Diagrams

7.3.1 Data Flow Diagram

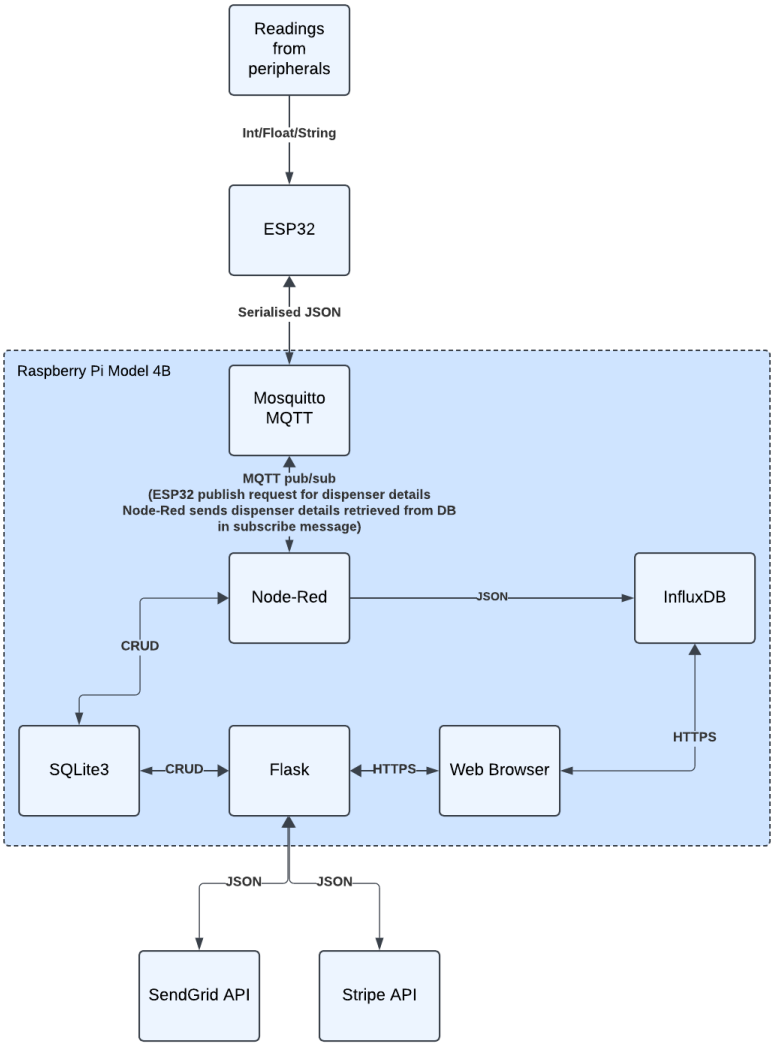


Figure 28: Data flow diagram for the entire project

Fig. 28 gives a complete flow of data for the entire system, including what format data is expected to be sent/received in to allow effective communication

between many pieces of software and hardware. The diagram also shows the intention for the ESP32 to receive product name and details from the self-checkout database to allow for product name and prices to be edited from the self-checkout admin.

7.3.2 State Diagram

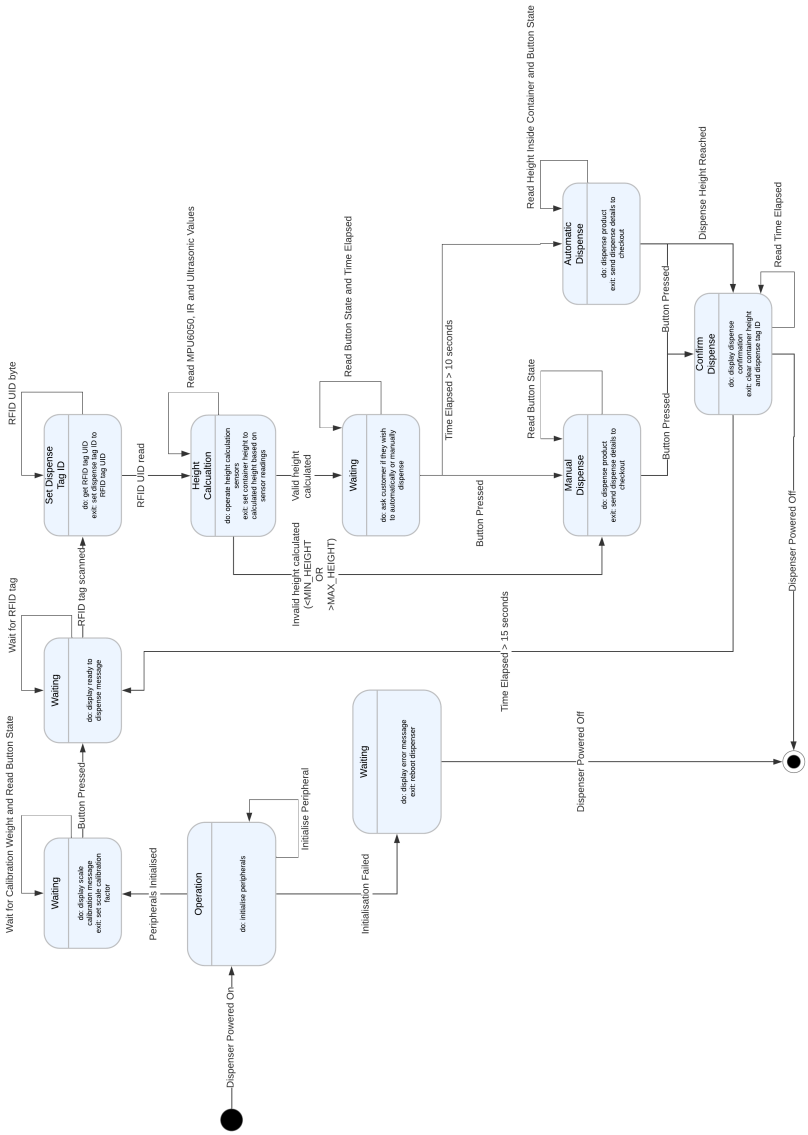


Figure 29: State diagram for the dispensing system

Fig. 29 defines the dispensing system's error handling, how it will interact with customers, and when it will store/dispose information about the current

dispense at a lower level than **Figs. 24-26**. A timeout feature helps to keep customer input to a minimum while not interfering with the system's effectiveness, and using a single button promotes simplicity and does not intimidate customers.

7.3.3 Class Diagram

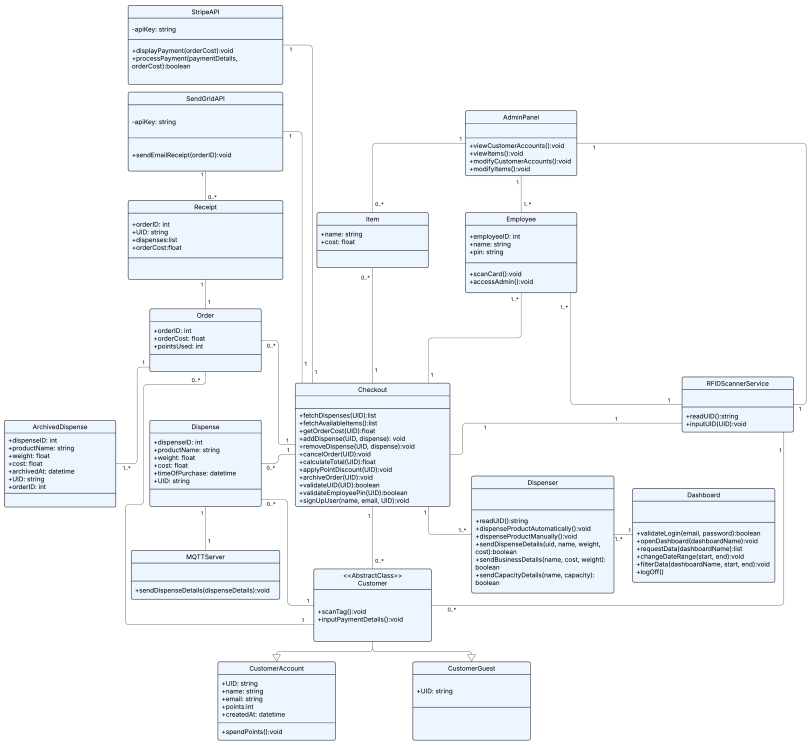


Figure 30: Class diagram for the entire system

Fig. 30 separates the project into objects, visualising the potential separation of concerns for final implementation, and how external services might communicate through the Checkout. The classes Checkout, Dispenser, Dashboard, RFDIScannerService, and AdminPanel are controller classes.

7.3.4 Sequence Diagram

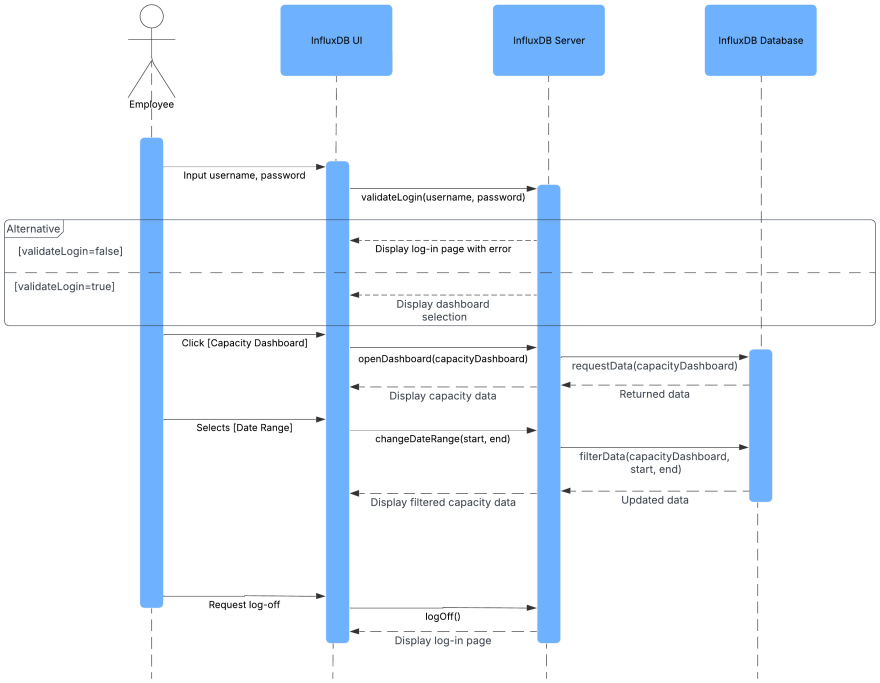


Figure 31: Sequence diagram for opening and filtering the dashboard

Using methods from **Fig. 30**, **Fig. 31** shows the steps employees would take to access a particular dashboard and filter the results as desired.

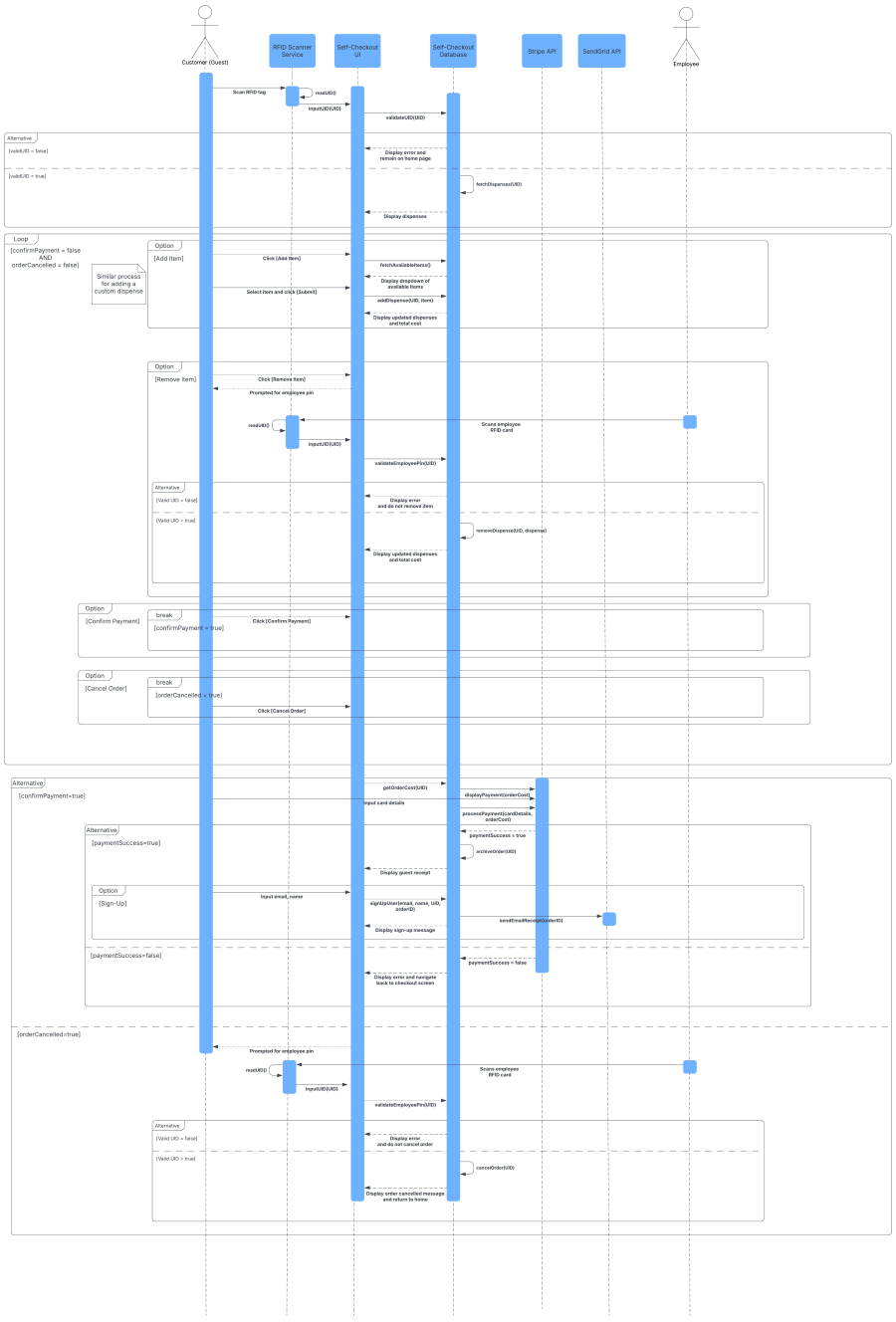


Figure 32: Sequence diagram for a guest customer using the self-checkout

Fig. 32 uses methods from **Fig. 30** to detail the multiple options and error handling a customer can experience when using the self-checkout as a guest. This includes the actions requiring employee input, such as when a customer decides to remove a dispense.

7.3.5 Entity Relationship Diagram

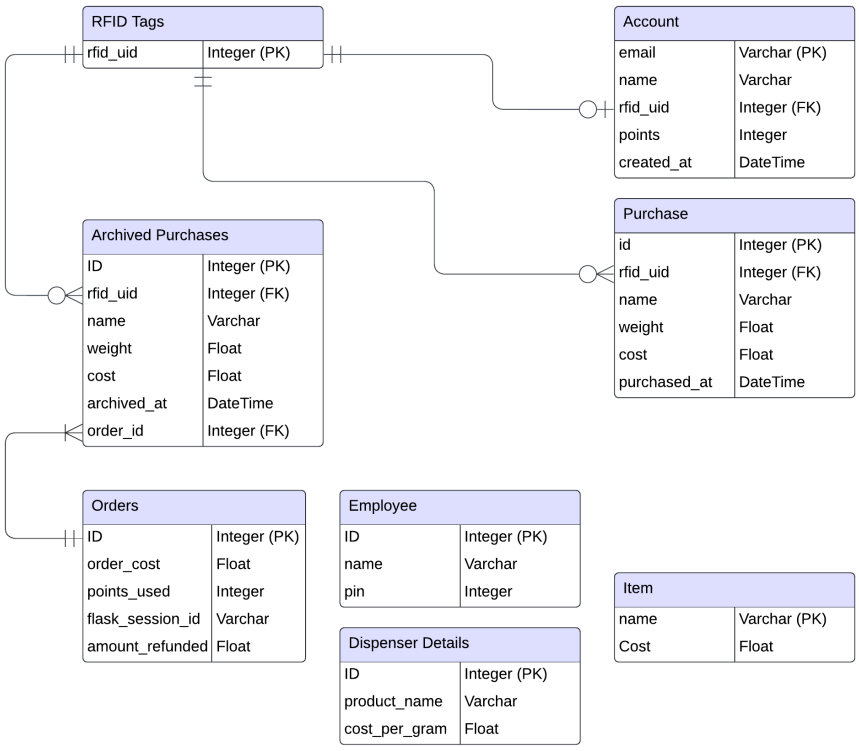


Figure 33: Design stage Entity Relationship Diagram (ERD) for the self-checkout database

Fig. 33 is my final database design before implementation, the Employee table stores employee pins, the Item table will store pre-defined stock, such as glass containers, RFID Tags stores valid customer tag numbers, and dispenser details stores product names and costs for dispensers to retrieve using MQTT/Node-RED. When a Purchase is paid for it is moved to Archived Purchases, which will be attached to a row in Orders. Purchase and Archived Purchases will be

attached to a customer tag/RFID UID.

7.3.6 Circuit Diagrams

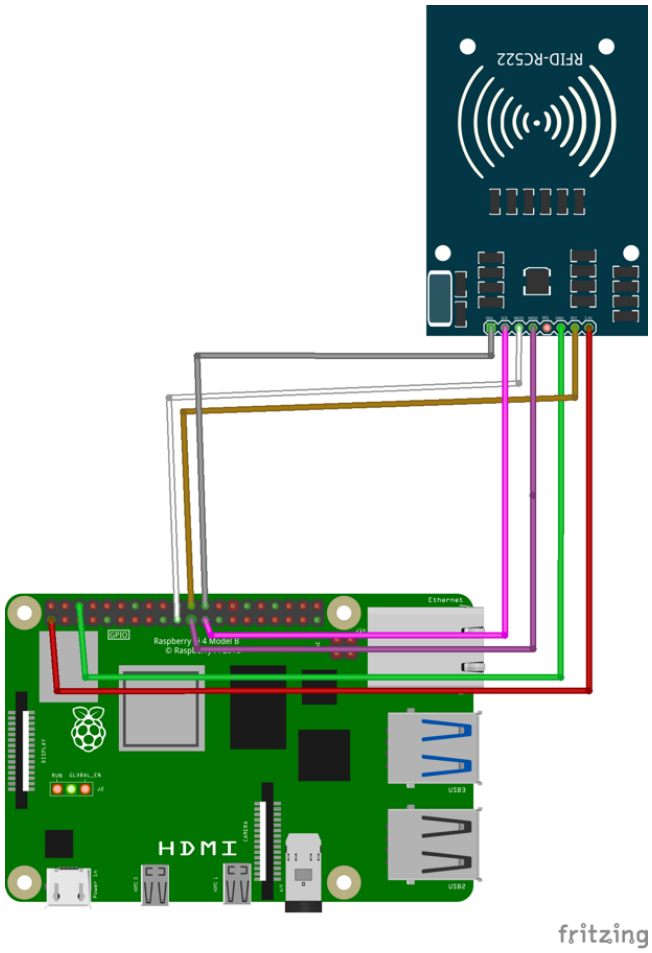


Figure 34: Design stage RaspberryPi 4 Model B circuit diagram

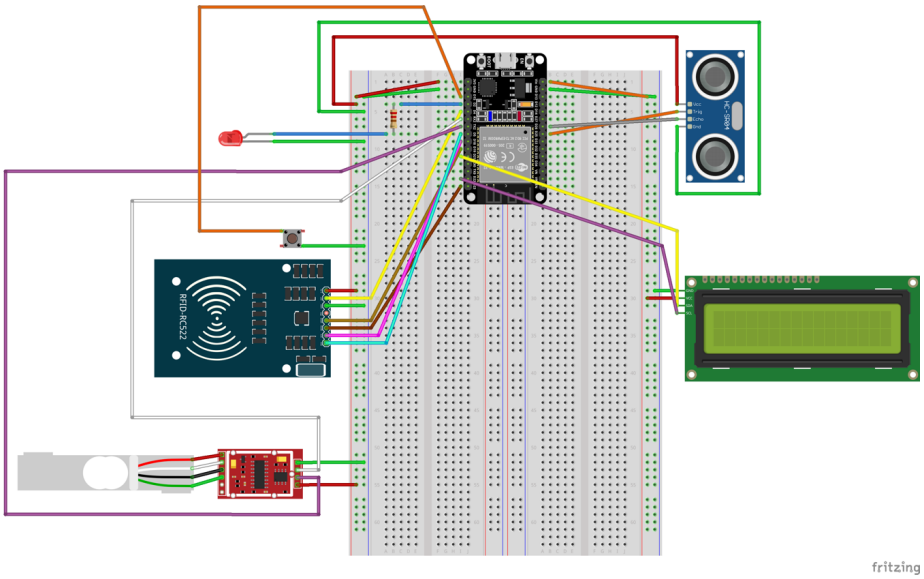


Figure 35: Design stage solids dispenser circuit diagram

Figs. 34 & 35 are circuit diagrams created before beginning any electronics wiring; they help to see how required peripherals can be connected to the controller while also providing useful implementation diagrams when applicable modifications are applied. The solids dispenser **(Fig. 35)** will be manually operated, allowing for the presentation of both manual and automatic dispenser models, with solid spillages being less of an issue than liquid spillages. My approach to the liquid dispenser will be trial and error with various sensors to achieve the best results, so it does not yet have a circuit diagram.

7.3.7 Test Design and Test Stages

During the design phase I have formulated a preliminary test plan **(Appendix B, Figure 7)** derived from the project's requirements, detailing essential features to evaluate within my system. The majority of implementation will involve iterative testing, primarily on edge cases, with thorough testing of the entire project occurring near the end of development.

Thorough testing will include the development of test cases, development of unit tests, trial-and-error testing, applying necessary fixes, and repeating until all errors are resolved. Unit testing will be conducted using PyTest with the self-checkout. Testing of the dispensers will involve completing many dispenses, verifying everything works as expected, and attempting to capture the results of

unexpected customer behaviour. Testing of the dashboards will involve verifying a mixture of both real and dummy data presents as expected.

7.4 Design Conclusion

Following this, design is extensive enough to confidently begin implementation. The majority, if not all, of the intended systems features have been captured, and the design is in line with high priority requirements as per **Appendix B, Figure 8**.

8 Implementation

8.1 Hardware

8.1.1 Solids Dispenser

Figs. 36-38 present the final solids dispenser; the wiring is unchanged from my design in **Fig. 35**. Solid product can be dispensed, weight and cost are calculated and displayed digitally, RFID tags can be read, and remaining capacity is measured. The dispenser also has the addtional benefit of displaying a digital price tag, simplifying product price changes within the store.

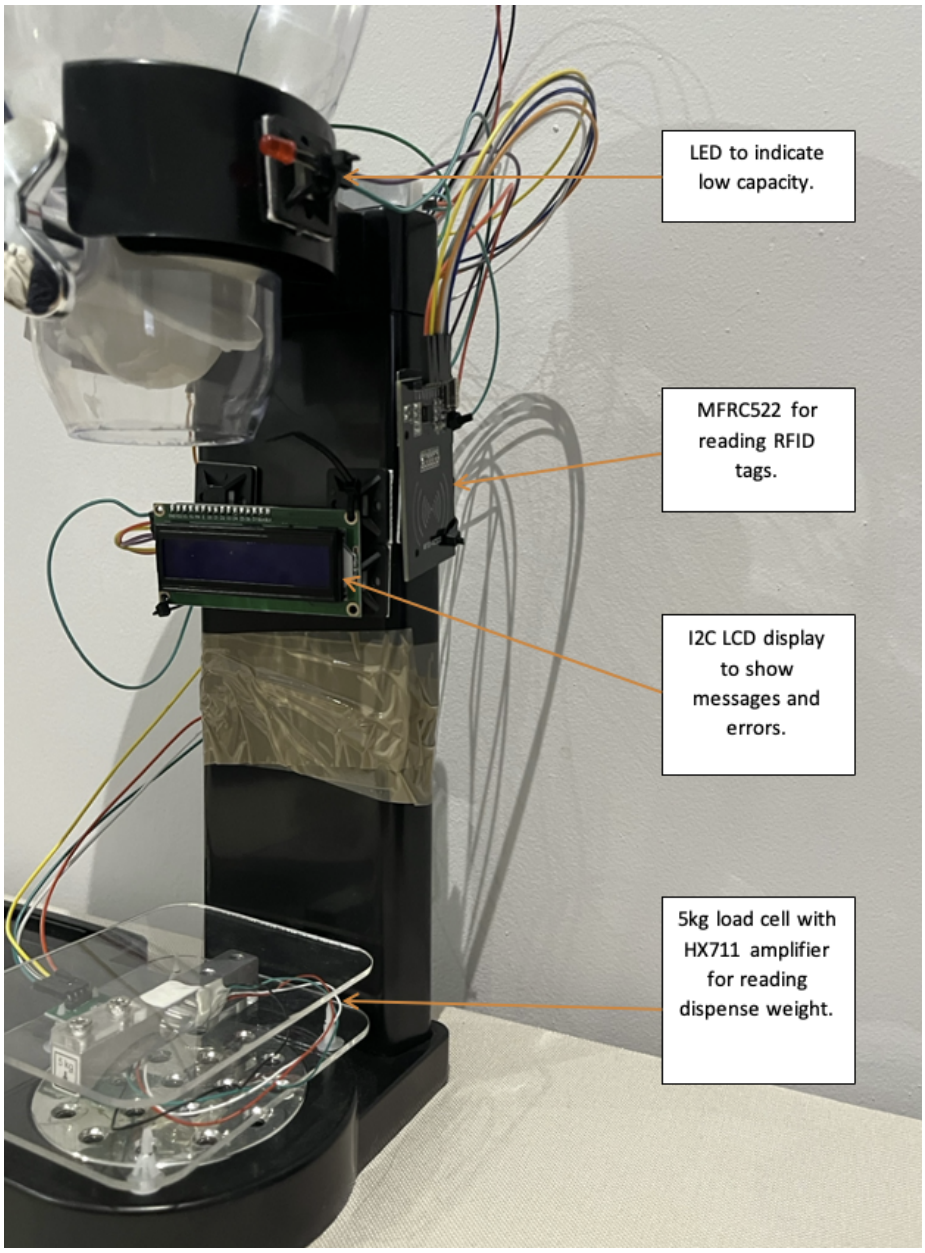


Figure 36: Right side of solids dispenser with annotations

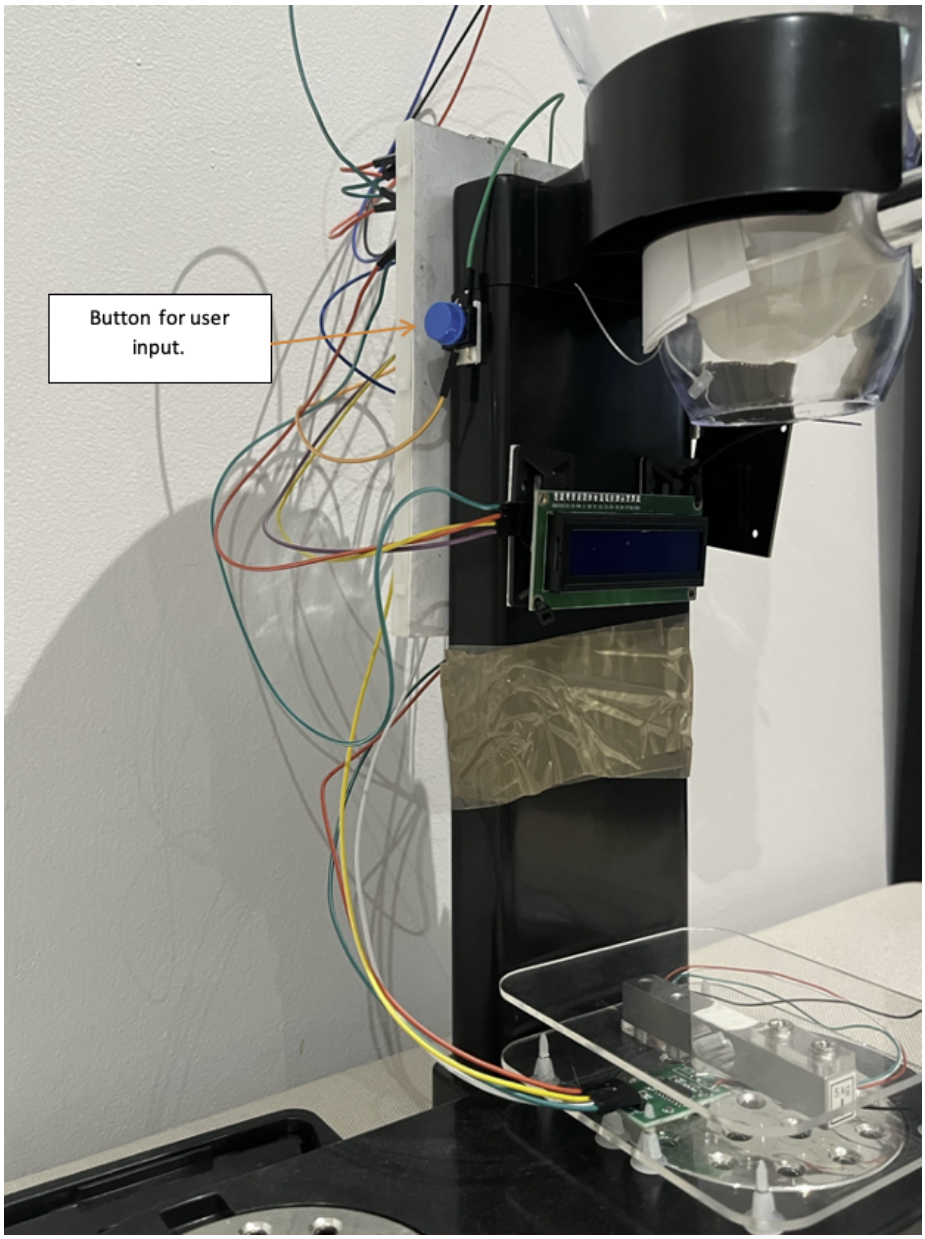


Figure 37: Left side of solids dispenser with annotations

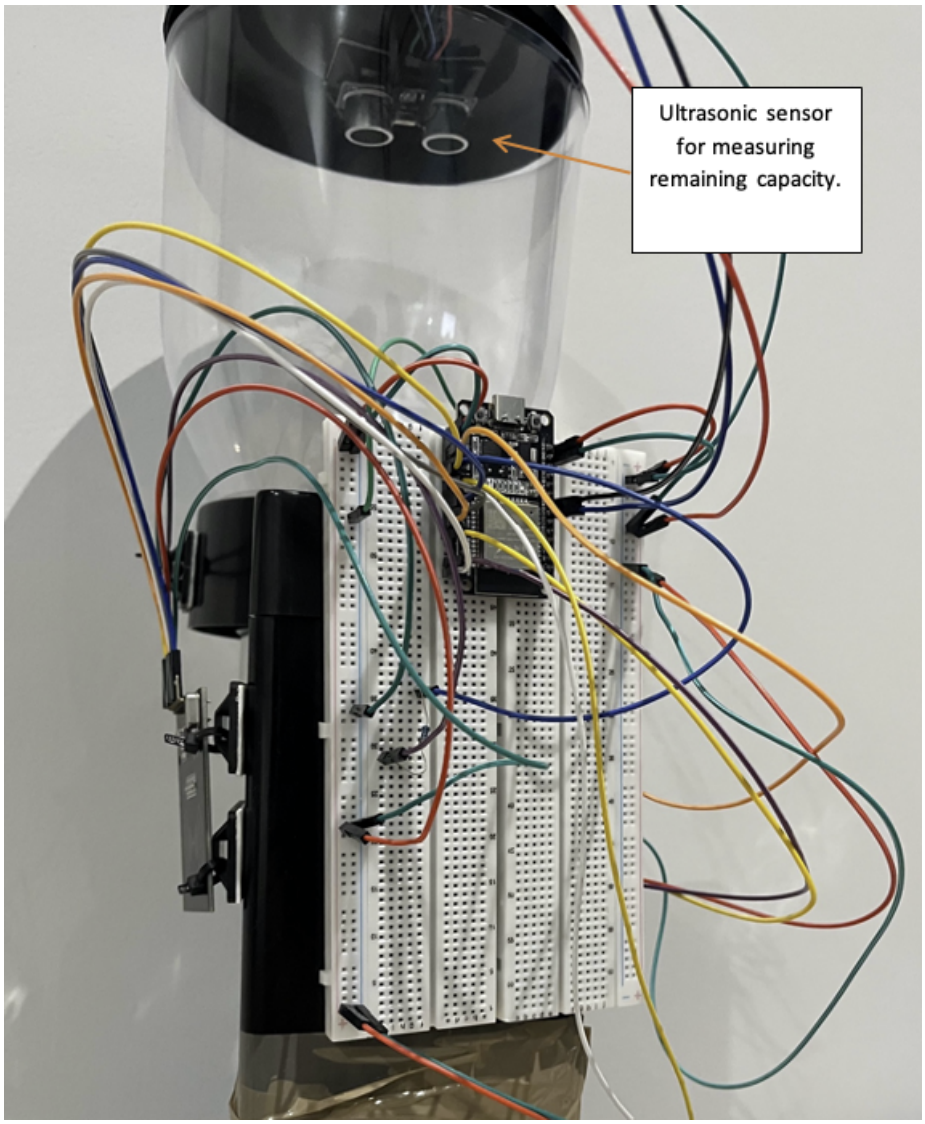


Figure 38: Back of solids dispenser with annotations

8.1.1.1 Challenges Encountered

HX11 Load Cell/Scale

For the scale I found there was a trade-off between accuracy and weight limit,

through researching the weights of common zero-waste shop products and testing the accuracies of different scales, I found 5kg was the best combination of weight limit and accuracy.

I also found the scale needed to be calibrated on boot to adapt to environmental factors such as temperature and humidity, I used the equation below from **Random Nerd Tutorials, 2025**[39] and code shown in **Fig. 39** with a 500g calibration weight to overcome this.

$$Calibration\ Factor = \frac{Weight\ Reading}{Known\ Weight}$$

```
//Calibrate the scale on system start up
display.clearDisplay();
display.setCursor(0,0);
display.println("Scale calibration needed!");
display.println("Place 500g of weight on scale");
display.println("and press button...");
display.display();

//Waiting for button press
while (digitalRead(BUTTON_PIN) == HIGH) {
  delay(50);
}

//Uses known weight to calculate appropriate calibration factor
if(scale.is_ready())
{
  float scaleValue = scale.get_value(50); //Gets scales average reading over 50 readings

  //Calculating calibration factor based on the error between scale reading and known weight
  float knownWeight = 500.0; //Weight of calibration weight
  float calibrationFactor = scaleValue / knownWeight;
  scale.set_scale(calibrationFactor);

  //Display calibration factor
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("Calibration Done!");
  display.print("Calibration Factor: ");
  display.println(String(calibrationFactor, 2));
  display.display();
  delay(1000);
}
else
{
  //Show error message if scale not ready
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("Error occured, scale is not ready.");
  display.println("Restart system and try again.");
  display.display();
  while (true); //Stop program if scale not ready
}

//Print scale ready if calibration successful
display.println("Scale Ready! Please remove the calibration weight so the scale can reset.");
display.display();
delay(8000); //Delay to give time for removal of calibration weight
scale.tare(); //Zero scale with new calibration factor
```

Figure 39: Scale calibration code

8.1.2 Liquid Dispenser

The liquid dispenser shares similarities to the solids dispenser, with a few key differences, namely a 12V peristaltic pump allowing quicker dispensing and increasing the efficiency of the customer flow, an OLED screen, and additional sensors to minimize spillages. The final circuit and layout of the liquid dispenser can be seen in **Figs. 40 & 41**. Its implementation was the most challenging aspect of the project due to challenges discussed below.

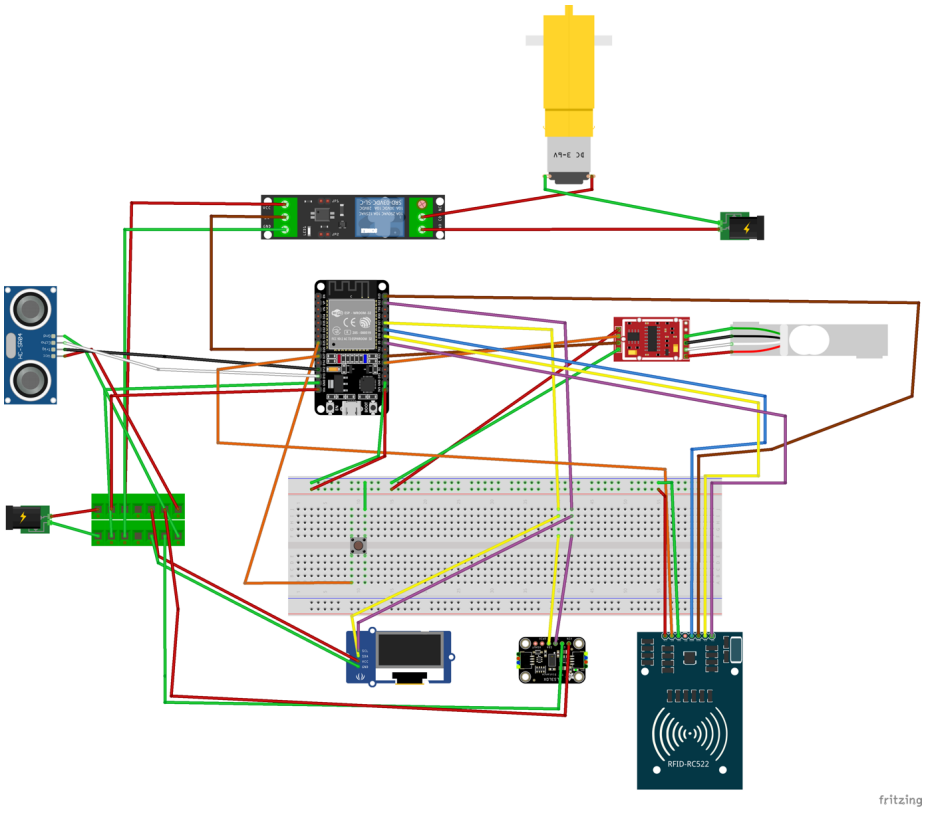


Figure 40: Final liquid dispenser circuit diagram

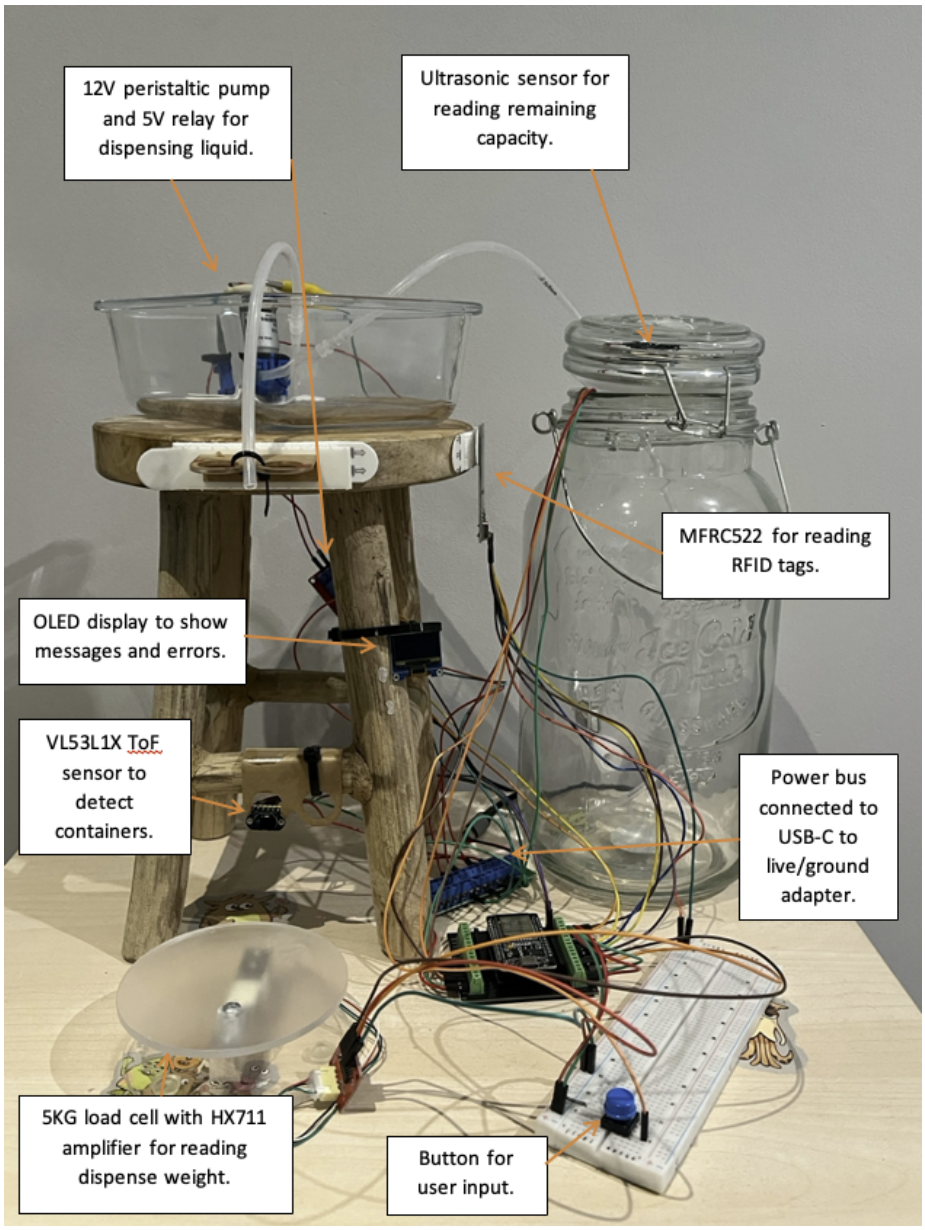


Figure 41: Liquid dispenser layout with annotations

8.1.2.1 Challenges Encountered

Delivery Issues

Long wait times for new hardware hindered the progress, delivery times would often be multiple weeks, new hardware was often required as product descriptions tended to overclaim real world performance. If a part didn't work as expected, or lacked accuracy, I would be waiting for its replacement.

Liquid Dispenser Power Issues

Attempting to power the liquid dispenser through the ESP32's VIN and 3.3V pins resulted in unexpected behaviour and inconsistent readings. The ESP32 can provide a maximum of 500mA of power, which was insufficient to power the additional hardware required for liquid dispensing.

I found it incredibly difficult to source a solution for this online, as much of the published content assumed a lot of prior knowledge I simply didn't have. It required me to further understand how power is provided and flows in a circuit. After estimating the milliamps the dispenser would require, with a buffer to handle spikes, I concluded 5V with 3000mA would be sufficient.

Deciding on the power source was challenging, USB-C 3.0 seemed most appropriate, as not only are the cables ubiquitous, but a single plug adapter can power multiple dispensers. The source of power can be adapted to meet a shop's needs, for example, if there were a lack of outlets, power banks could provide power and be charged outside of business hours. A power distribution board **(Fig. 42)** is used to power 5V peripherals and the ESP32, whilst the ESP32's 3.3V pin powers 3.3V peripherals.

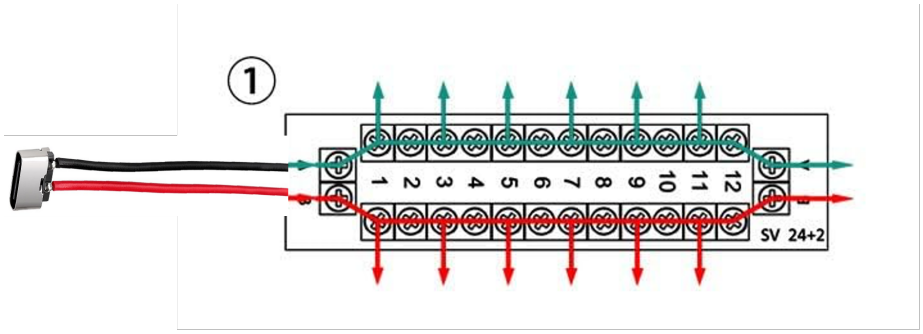


Figure 42: Power distribution board with USB-C adapter

12V Peristaltic Pump

Being 12 volts, the liquid pump required its own power supply, and consequently

a method to connect it to the ESP32.

Initially, I used a relay module without an optocoupler to allow the ESP32 to interface with the 12V pump, but the pump would remain powered-on. I realised the cause of the issue was the relay requiring 5Vs to trigger on/off, which the ESP32 couldn't provide operating on 3.3V logic. To address this, I implemented a JQC-3FF-S-Z relay module, which possesses an optocoupler. The optocoupler provided better isolation, helping protect the ESP32, but also required less current to trigger, meaning a 3.3V signal was sufficient to reliably trigger the relay, and consequently the pump.

Height Calculation

The biggest problem with not just the liquid dispenser, but the whole project, was attempting to implement a non-contact solution for automatic dispensing into unknown containers. An issue considered from the project's inception **(Appendix A, Fig. 1)**. There were few resources online, much of what I considered useful prior to beginning implementation had major differences in what they considered met the requirement of automated dispensing. For example, some dispensed liquid for a known container **(CircuitDigest, 2018)**[7], and much of the resources dispensed solely based on whether a container was detected by an ultrasonic sensor **(AutomatedSteven, 2023)**[3]. A useful resource utilised SOHCAHTOA to calculate the height of tall buildings, but even this required user input for adjusting the angle of measurement **(Electronics Champ, 2022)**[12].

My initial plan was to use an ultrasonic sensor on a servo motor, measuring the distance at each angle increment, then stopping when the ultrasonic no longer detected the container. Using the initial distance between the ultrasonic and the container, and the final recorded angle of the servo motor converted to radians, to get the adjacent/height of the container.

$$adjacent = \frac{opposite}{tan(\theta)}$$

This setup was not accurate enough to detect any height consistently, taking ultrasonic readings during movement resulted in inaccuracies, whilst the servo angle did not satisfy the high accuracy required. Following this I added an IR sensor to detect the container and signal the servo to stop, alongside an MPU-6050 gyroscope to record an accurate angle of measurement. This saw success with select containers, but provided inconsistent results, particularly with glass containers due to infrared detection. Following a Strengths Weaknesses Opportunities Threats (SWOT) analysis provided in **Appendix C, Figure 1**, I decided to remove the feature from the final implementation. The implementation artefacts for automatic dispensing are included in **Appendix C, Figures 2-5**.

The final solution of preventing dispensing when not detecting a container beneath the pump tubing worked as an alternative, achieving the desired outcome of reducing spillages.

8.1.3 RaspberryPi RFID Scanner

To read and submit an RFID tag's UID into a selected text field, I developed a background Linux service involving a python script to read the UID and simulate keyboard input. The wiring between the Pi and the RFID scanner follows wiring defined in **Fig. 34**.

8.1.3.1 Challenges Encountered

Keyboard Simulation

Initially for simulating keystrokes, I used the pynput library, but it failed to recognise the focused input. After considerable troubleshooting, I discovered the model RaspberryPi 4B uses the wayland display manager, which is incompatible with python keyboard input libraries. Wtype is a Linux package designed to be wayland-compatible for simulating keystrokes, which after configuration discussed below, was able to recognise the focused window and input field within my service, replacing pynput.

Incorrect UID Reading

My project utilizes 13.56Mhz MIFARE tags with a UID of 4 bytes. Using the mfrc522 library, I couldn't understand why the RFID number was reading completely different on my RaspberryPi compared to the dispensers, after reading the RFID in bytes on the RaspberryPi there were 5 bytes being read, with the final byte having a value of 0, changing the decimal interpretation entirely. A solution was offered by the pirc522 library, which operates similarly to mfrc522, but reads the UID into an array, making it easily split with python.

8.1.3.2 Python Script

Fig. 43 is the script used by my Linux service. The key components of this script are:

- **Signal**: Using signal to detect shutdowns (SIGTERM signal) and allow the service to exit gracefully alongside resetting GPIO pins.

- **Subprocess**: Using subprocess to run Wtype in the command line with the detected RFID UID.
- **Reading UID**: Reading the 4 UID bytes, then converting them into a hexadecimal string, then into a decimal number, and finally a string with an appended `\n` to ensure the input is submitted when passed to Wtype.

```python
import signal
import sys
import time
import RPi.GPIO as GPIO
from pirc522 import RFID
import subprocess

#Following script runs as a service on the raspberry pi

#Function for when a shutdown is detected``
def onShutdown(signal, frame):
    #Reset GPIO pins on the RPi
    GPIO.cleanup()
    #Quits the script with code 0 (Success)
    sys.exit(0)

def typeText(text):
    try:
        #Using wtype which allows keyboard simulation for wayland display managers
        #Couldn't use pynput because it didn't recognise the focused window
        subprocess.run(['/usr/bin/wtype', text])
    except Exception as e:
        print(f"Error sending text: {e}")

#Listening to shutdown, SIGTERM is for service shutdowns
signal.signal(signal.SIGTERM, onShutdown)
#Init RFID scanner with default GPIO pins, SDA = 8, SCK = 11, MOSI = 10, MISO = 9, RST= 25
scanner = RFID(pin_irq=None)


try:
    while True:
        #Reader.read returns other information, only concerned with UID
        print("Waiting...")
        (error, tag_type) = scanner.request() #Checking for tag
        if not error:
            (error, uid) = scanner.anticoll()
            if not error:
                uid = uid[:4] #Only read first 4 bytes, same as esp32 MFRC522 library
                uid = ''.join(format(number, '02X') for number in uid) #Get array of decimal numbers into hexidecimal as a string
                decimalUID = int(uid,16) #Python will convert the hex into decimal
                #Convert to string for input, \n will submit after string is typed
                decimalUID = str(decimalUID) + "\n"
                #Typing RFID uid
                typeText(decimalUID)
                #Delay to prevent multiple inputs
                time.sleep(5)
        time.sleep(0.5) #To slow polling rate
except Exception as e:
    print(e)
    #Reset GPIO pins on the RPi
    GPIO.cleanup()
finally:
    #Reset GPIO pins on the RPi
    GPIO.cleanup()
```

Figure 43: RFID scanning script (.py file)

8.1.3.3 Service Configuration

Fig. 44 presents the service file used to configure the RFID scanner service. The essential configurations include:

- **ExecStart**: Defines the python configuration and script to use.
- **Environment Variables**
  - `"DISPLAY=:0" and "XDG_RUNTIME_DIR=/run/user/1000"`: Ensures Wtype has context to allow it to operate on the currently focused Wayland window.
  - `"PATH=/home/Arthur/Desktop/rfidservice/.venv/bin:$PATH"`: Defines the virtual environment to use, with the appropriate libraries.
- **Restart=always**: Safeguards against unexpected crashes by restarting the service.

```
[Unit]
Description=RFID Scanner Service
After=network.target

[Service]
ExecStart=/home/arthur/Desktop/rfidservice/.venv/bin/python /home/arthur/Desktop/rfidservice/input_scanned_rfid.py
WorkingDirectory=/home/arthur/Desktop/rfidservice
User=arthur
Group=arthur
Environment="DISPLAY=:0"
Environment="XDG_RUNTIME_DIR=/run/user/1000"
Environment="PATH=/home/arthur/Desktop/rfidservice/.venv/bin:$PATH"
Restart=always
StandardOutput=journal
StandardError=journal
SyslogIdentifier=rfid-scanner

[Install]
WantedBy=multi-user.target
```

Figure 44: Configuration for the RFID scanning service (.service file)

8.2 Software

8.2.1 SQLite Database

The final database implementation required minimal change from my design in **Fig. 33**, with **Fig. 45** presenting the final Entity Relationship Diagram (ERD). Database interactions are primarily done through SQLAlchemy within the Flask self-checkout, and standard SQL is used for IoT integration with other components.

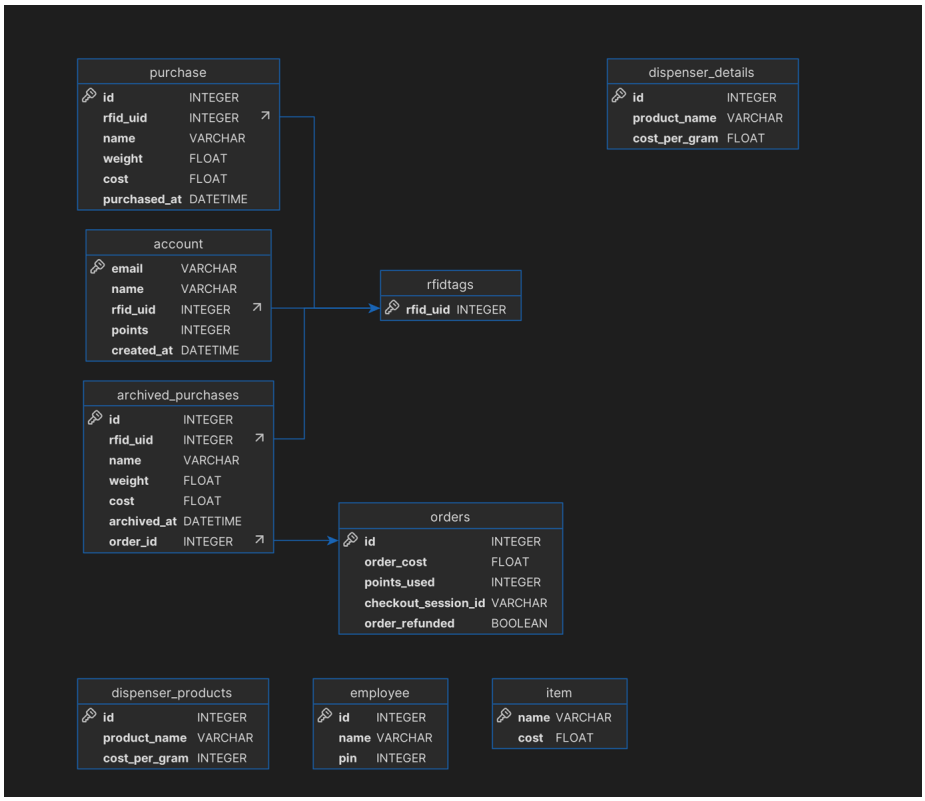


Figure 45: Reverse-engineered ERD using DbVisualiser

8.2.2 Self-Checkout System

Fig. 46 presents the self-checkout as it would appear to customers, displaying on a 16" gesture supported touchscreen with an on-screen keyboard for any input outside of scanning tags. The use of bootstrap ensures the self-checkout adapts to various screen sizes should a store have a pre-existing display of a different size. The full app directory including implementation screenshots are presented in **Appendix D, Figures 1-13**.

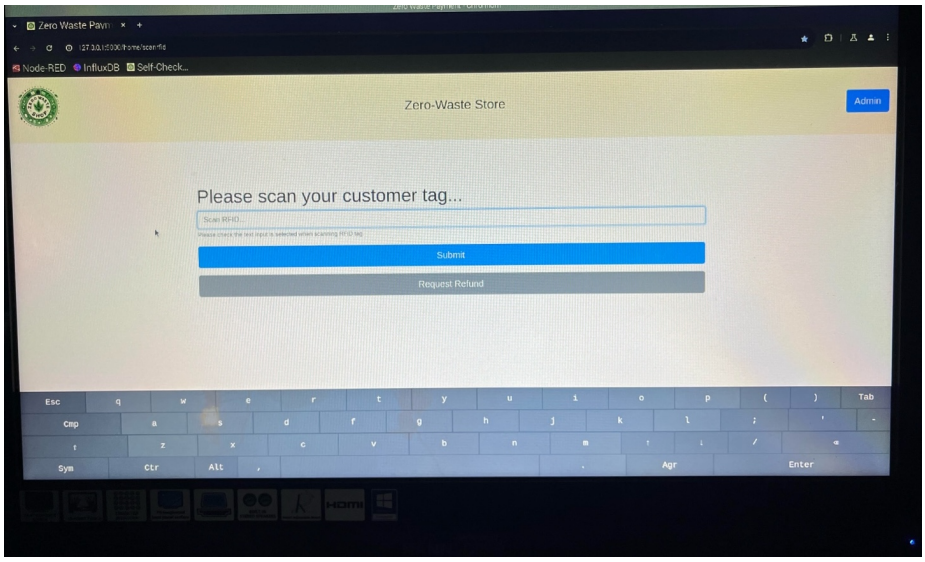


Figure 46: Self-checkout on 16" touchscreen with on-screen keyboard

8.2.2.1 Checkout

Most options on the checkout will prompt the customer for an employee pin, the idea being much of the checkout process should already be present from data received from dispensers. However, options such as navigating to your basket or adding an item can be done without authorisation. Most features are handled using Ajax to create a smoother customer experience, with success/error messages using either flashes or alerts.

8.2.2.2 Customer Accounts

On completing a guest purchase, customers are presented with the option to create an account and keep the tag used for future purchases. On registering, the points for that purchase will be added and an email receipt will be sent using SendGrid. Already registered customers will be automatically recognised based on their tag, streamlining their checkout process, and allowing the use of points for purchases.

8.2.2.3 Payments and Refunds

Payments and refunds are handled using the Stripe API, on successful payment

the `payment-session-ID` is stored within the Order table of the database. This can then be used, alongside backend input validation, to process any requested refunds, and refunded orders are marked as refunded. Any point additions/discounts are handled in the Flask backend.

8.2.2.4 Flask-Admin

Flask-Admin is used with the self-checkout to allow employees to Create Read Update Delete (CRUD) customer accounts, RFID tags, dispenser details, employee pins, and items. I have modified the admin view to require authentication, done through an employee login page using Flask sessions, and overriding the built-in `AdminIndexView` and `ModelView` classes **(Fig. 47)**.

```python
#Adding auth check to main admin page
You, 2 weeks ago | 1 author (You)
class CustomAdminIndexView(AdminIndexView):
    @expose('/') #To change the main admin page
    def index(self):
        #If authorised use normal index view, otherwise redirect to home
        if session.get("AdminAccess"):
            return super().index()
        else:
            flash("You are not authorised to access this page.")
            return redirect(url_for('home.rfidUID'))


#Adding auth check to other admin pages, secure model view built in
#is_accessible and inaccessible_callback are built in with Flask-Admin
#All model views will inherit from the custom view
You, 2 weeks ago | 1 author (You)
class SecureModelView(ModelView):
    def is_accessible(self):
        return "AdminAccess" in session

    def inaccessible_callback(self, name, **kwargs):
        flash("You are not authorised to access this page.")
        return redirect(url_for('home.rfidUID'))
```

Figure 47: Code for requiring authentication on admin views

To maintain the integrity of a customer's purchase history, should an employee change the RFID tag linked to a customer account, a modified `on_model_change` will set all purchases since the account's creation to the new RFID UID **(Fig. 48)**.

Figure 48: Code to update customer purchase history

8.2.3 InfluxDB Dashboards

The final dashboard implementation remains relatively true to the initial wireframes in **Figs. 19 & 20**, with final designs presented in **Fig. 50** and **Figs. 55 & 56**. Both dashboards were developed using the InfluxDB GUI, alongside the Flux programming language, to create queries and graphs which interact with a time-series database. The basic structue of the database is displayed in **Fig. 49**. Key Flux functions included:

- **aggregateWindow**: Used to aggregate data on a given interval, paired with `createEmpty` for smoother graphs.
- **range**: Filter the data used in a query by a given date range.
- **filter**: Filter the rows of data by a condition such as measurement or field.

bucket name	_measurement	_field	tags
dispense_data	dispenses	cost, weight	product_name
capacity_data	capacities	capacity	product_name

Figure 49: Schema for InfluxDB time-series database

8.2.3.1 Capacity Dashboard

Figure 50: Final capacity dashboard

The query for displaying the capacity of a product **(Fig. 51)** uses the `capacity_data` bucket, then filters the data by the capacities measurement and the desired product name, whilst `last()` ensures only the most recent capacity is displayed.

```
from(bucket: "capacity_data")
  |> range(start: 0)
  |> filter(fn: (r) => r._measurement == "capacities") //Filter by capacities measurement
  |> filter(fn: (r) => r.product_name == "Cashew Nuts") //Filter by product name tag
  |> last()  //Only get last submission
```

Figure 51: Capacity query

The capacity dashboard includes integration with an external notification endpoint, namely a `capacity_data` Slack channel. Every hour a check is run on the most recent capacities within that hour. Any capacities lower than 10% trigger an InfluxDB check **(Fig. 52)**. Triggered checks are sent as an alert using a notification rule **(Fig. 53)**, which details the Slack endpoint and structure of the notification(s). While the boilerplate code regarding notifications is mostly handled by InfluxDB, the specific use case required modifications.

```
import "influxdata/influxdb/monitor"
import "influxdata/influxdb/v1"

data =
    from(bucket: "capacity_data")
        |> range(start: -1h)
        |> filter(fn: (r) => r["_measurement"] == "capacities")
        |> filter(fn: (r) => r["_field"] == "capacity")
        |> aggregateWindow(every: 1m, fn: last, createEmpty: false)

option task = {name: "Capcity Check", every: 1h, offset: 0s}

check = {_check_id: "0e847ee0933af000", _check_name: "Capcity Check", _type: "threshold", tags: {}}
crit = (r) => r["capacity"] < 10.0
messageFn = (r) => "Product '${r.product_name}' has a low capacity of ${r.capacity}%."

data |> v1["fieldsAsCols"]() |> monitor["check"](data: check, messageFn: messageFn, crit: crit)
```

Figure 52: Capacity check

```
import "influxdata/influxdb/monitor"
import "slack"
import "influxdata/influxdb/secrets"
import "experimental"

option task = {name: "Capcity Alert Rule", every: 1h, offset: 0s}

slack_endpoint =
    slack["endpoint"](
        url: "SLACK URL HERE",
    )
notification = {
    _notification_rule_id: "0e848155ab2b1000",
    _notification_rule_name: "Capcity Alert Rule",
    _notification_endpoint_id: "0e847f22f6c56000",
    _notification_endpoint_name: "Capacity Alert",
}
statuses = monitor["from"](start: -1h)
crit = statuses |> filter(fn: (r) => r["_level"] == "crit")
all_statuses =
    crit |> filter(fn: (r) => r["_time"] >= experimental["subDuration"](from: now(), d: 1h))

all_statuses
    |> monitor["notify"](
        data: notification,
        endpoint:
            slack_endpoint(mapFn: (r) => ({channel: "", text: "${r._message}", color:
"danger"})),
    )
```

Figure 53: Capacity notification rule

Fig. 54 displays a typical capacity notification, the benefit of using Slack is

cross-platform notifications, as it can notify employees via either web or mobile.

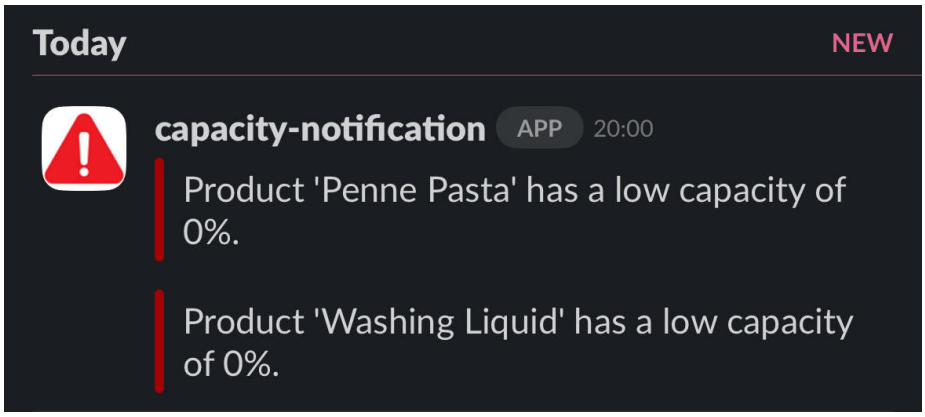


Figure 54: Capcity notification example

8.2.3.2 Business Dashboard

Figure 55: Final business dashboard (1)

Figure 56: Final business dashboard (2)

The final business dashboard captures information on both stock and revenue, focusing on daily targets generated from prior sales, and graphs that can be filtered by the user's given date range. The daily graphs use a hard coded range **(Fig. 57)**, and will not change on user request.

```
import "experimental"
import "date"

startTime = experimental.addDuration(
  to: date.truncate(t: now(), unit: 1d),
  d: 7h
)
stopTime = experimental.addDuration(
  to: date.truncate(t: now(), unit: 1d),
  d: 18h
)
```

Figure 57: Date range for daily graphs (7am-6pm)

8.3 Hardware and Software Communication

Communication between devices and technologies is achieved over Wi-Fi. The RaspberryPi was configured to a static IP to simplify configuration, as using localhost was proving troublesome. The addresses of relevant technologies are

as follows:

- **RaspberryPi**: 192.168.1.100
- **Flask**: 192.168.1.100:5000
- **MQTT**: 192.168.1.100:1883
- **Node-RED**: 192.168.1.100:1880
- **InfluxDB**: 192.168.1.100:8086

8.3.1 MQTT

Appendix E, Figure 1 presents a full list of MQTT topics and their function, **Fig. 58** includes the basic structure of publishing a JSON payload. The PubSubClient Arduino library is used to connect to the MQTT, publish messages, and subscribe to topics. **Fig. 59** is also called throughout my dispenser code to ensure a high uptime and impede connection drops.

```
//Trys to reconnect MQTT if connection drops
if (!client.connected())
{
  reconnect_MQTT();
}
client.loop();

//Details which will send to MQTT
//JSON string payload for MQTT of dispense details
String businessDispensePayload = "{";
//Sending as string
businessDispensePayload += "\"name\": \"" + PRODUCT_NAME + "\"";
//Sending as number
businessDispensePayload += ",\"weight\": " + String(amountDispensed, 2);
//Sending as number
businessDispensePayload += ",\"cost\": " + String(costDispensed, 2);
businessDispensePayload += "}";

//Publishing to MQTT under business dispense topic
if(client.publish(topic_business_dispense, businessDispensePayload.c_str()))
{
  Serial.println("Dispense successfully sent to dashboard.");
}
else //If error sending data message is displayed
{
  Serial.println("Failed to send dispense.");
}
```

Figure 58: Example MQTT publish code

```
//Function to reconnect to MQTT broker
void reconnect_MQTT()
{
  //Loops until reconnected to MQTT broker
  while (!client.connected())
  {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Reaching Checkout");
    Serial.println("Attempting MQTT connection...");
    int dotPosition = 0;
    if (client.connect("ESP32Client", mqtt_user, mqtt_password))
    {
      Serial.println("connected");
      //Subscribing to rfid validation topic
      client.subscribe(topic_verify_response);
      //Subscribing to response topic
      client.subscribe(topic_dispenser_response);
    }
    else
    {
      lcd.setCursor(dotPosition,1);
      lcd.print(".");
      dotPosition++;
      Serial.println("Failed, Reason Code=");
      Serial.println(client.state());
      Serial.println(" trying again in 5 seconds...");
      delay(5000);
    }
  }
}
```

Figure 59: MQTT reconnect function

8.3.1.1 Challenges Encountered

Dynamic Publish and Subscribe

The biggest challenge working with MQTT was setting up a publish and subscribe relationship to allow dispensers to get details dynamically, which in turn allowed both verification of RFID tags, and the editing of product de-

tails through the self-checkout admin. A callback function is used to listen on the MQTT server **(Appendix E, Fig. 2)**, utilizing the ArduinoJson library to decode the returned JSON payload. Subscribing to topics including a dispenser's unique ID ensures dispensers do not retrieve the incorrect details, making communication scalable for many dispensers on the same MQTT server. **Appendix E, Figures 3-4** include the code for requesting product details and validating an RFID tag.

8.3.2 Node-RED

Node-RED is responsible for dictating the flow of MQTT messages, depending on their topic. By listening on the MQTT server, JSON payloads are formatted into SQL queries **(Appendix F, Figure 1)**, InfluxDB insert statements **(Appendix F, Figure 2)**, and dynamic MQTT response topics **(Appendix F, Figure 3)**. **Fig. 60** contains all flows and nodes, with their purposes annotated.

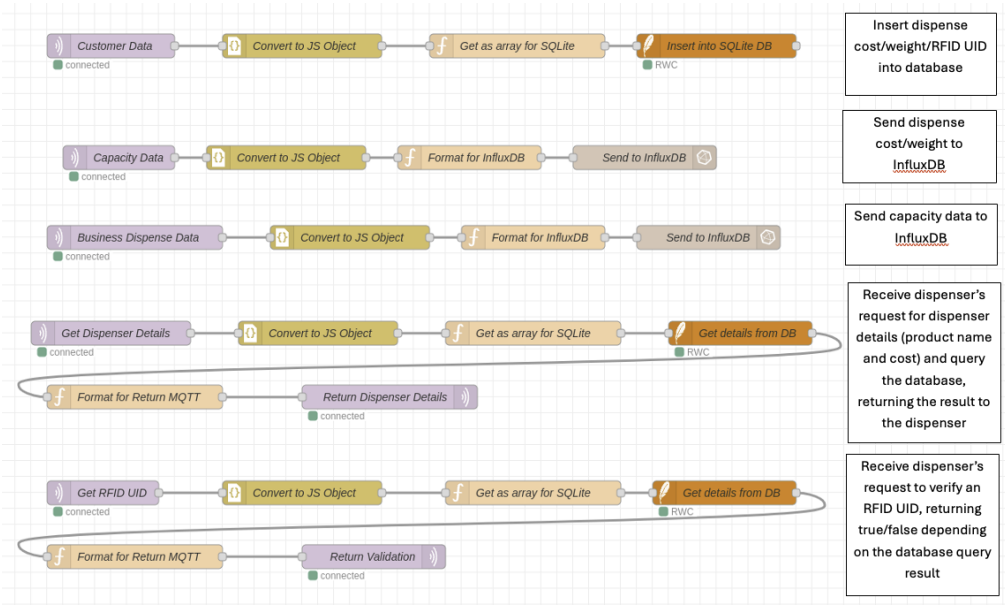


Figure 60: All Node-RED flows annotated with their purpose

8.4 Testing

8.4.1 Dispenser Testing

The testing of the dispensers involved 23 test cases, the results and details of these are shown in **Appendix G, Figure 3**.

8.4.2 Dashboard Testing

The dashboard testing involved 10 test cases, displayed in **Appendix G, Figure 4**. **Appendix G, Figure 1** contains the code used to assist testing, it sends dummy data at random to InfluxDB, which I then cross-referenced between the ESP32's serial and the dashboards for verification.

8.4.3 Self-Checkout System Testing

The testing of my self-checkout involved 50 test cases, captured through 56 unit tests, alongside manual testing. **Appendix G, Figure 5** presents the defined test cases with results, and **Fig. 61** presents the unit test results. The configuration used for the unit tests is shown in **Appendix G, Figure 2**.

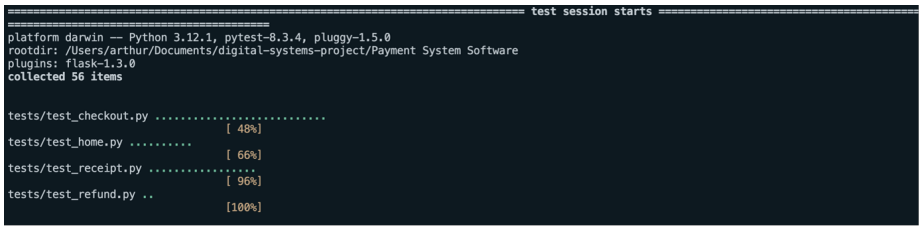


Figure 61: Unit test results

8.4.4 Reflection on Testing

My final test cases differ greatly from the initial test plan **(Appendix B, Figure 7)**, tests were mostly expanded into multiple test cases or removed. Simultaneously, many new tests were identified and created once implementation had completed. No tests failed on the dispensers or the dashboards, perhaps due to incremental testing during implementation having fixed issues prior to final testing. The self-checkout, however, required several changes to how it handled

data validation. These changes were mostly identified through unit testing, in which a variety of parameters were used, including unexpected inputs and edge cases.

Generally, testing was a smooth experience due to both incremental testing, and work in the design phase helping to mitigate unforeseen errors. Unit testing helped to thoroughly test the robustness of my Flask endpoints, discovering issues which required fixing. Some tests were more time-consuming than others, for example, the dashboard tests required a substantial amount of dummy data. Tests that spanned multiple components of the project, such as verifying the flow of messages, also took longer to complete. Additionally, some self-checkout testing was particularly complex due to the involvement of external APIs and multiple redirects.

9 Project Evaluation

9.1 Successes

Overall, I believe the project has been successful; with challenges either completed or worked around. Resulting in a system capable of handling customer transactions, loyalty accounts, product dispensing, capacity monitoring, and business-oriented insights. I believe the project fulfilled all its aims and objectives, alongside meeting all must and should requirements. Additionally, the project also met some could requirements. Unfulfilled requirements will be additional considerations for further work, alongside those outlined in **Chapter 10**.

Appendix H, Figures 1-3 contain a cost estimation for the two dispensers and the self-checkout. I consider the estimated costs to be a significant success of the project, particularly the self-checkout when considering supermarket self-checkouts can cost in the thousands, such as those offered by **NCR Voyix, 2025** [33].

A large success of the project was getting the communication between mixed hardware to a stage which provided accurate measurements of peripherals, whilst effectively guiding the flow of data from dispensers and the self-checkout. I faced many challenges in getting this to function, but the final solution proved its robustness during testing, and the mixed hardware combines to provide an improved customer flow within a zero-waste shop.

Further success of the project stemmed from the depth of my considerations prior to implementation. For example, the literature review appropriately informed my requirements and provided awareness of limitations to either avoid

or resolve within these requirements, which were then a great basis for decisions within subsequent chapters. The depth of my requirements allowed for good design choices as I knew the priority of features to capture, which then allowed for a smoother implementation process.

Furthermore, the Gantt chart within my methodology chapter provided a great background to judge whether I was on-track during every stage of the project, alongside weekly SCRUM stand-ups identifying the week's focus to remain on-schedule.

9.2 Limitations

The biggest limitation within the project is being unable to automatically dispense, this was a gap I identified during my literature review and wanted the project to solve, unfortunately, I was unable to achieve this due to inaccurate measurements with different container materials and sizes. Despite offering a cheaper alternative solution for spillage reduction, automatic dispensing would have helped to further differentiate the project from microcontroller-based dispensing solutions already in place and create more accessibility when dispensing. Perhaps with more expensive peripherals consistently accurate height calculations can be achieved, but this contrasts the aim of affordability.

Additionally, testing the scalability and real-world performance of the system is very difficult without implementing the solution within real zero-waste stores. Some scalability tests have been conducted, but they were limited to a handful of ESP32s sending dummy dispenses. Consequently, it is entirely possible unforeseen issues might be encountered when the project runs on a different network with more dispensers and long-term operation, such as hardware failures or communication issues.

9.3 Reflection

The project was a great learning experience for me, IoT was almost an entirely new concept prior to beginning the project, and my experience using microcontrollers was minimal. The project forced me to push myself and learn many new concepts, upon its completion I became familiar with an entire tech stack to deploy IoT solutions. I also learned a lot regarding circuit design, primarily the limitations of different pins types, and the flow of electricity. I had also not used Ajax prior to developing the self-checkout, and its use allowed for a much smoother customer experience.

If I were to do things differently, I would first ensure I thoroughly research real-world performance of peripherals, helping to prevent initial ambitions from being

unachievable due to overpromised performance of hardware. I would also ensure I have considered power requirements within my designs, this should prevent issues like those I encountered implementing the liquid dispenser. Additionally, I would use an SSD1306 OLED display instead of a 16x2 LCD in the solids dispenser, it was used for the purpose of testing its viability as a cheaper option, but its character limit is very harsh when displaying feedback to users.

Furthermore, I would utilise the React front-end framework for the self-checkout, I believe this offers a cleaner solution than Flask with Ajax. React components would provide more maintainable and reusable code blocks, my Ajax related code ended up being quite verbose by the end of development. React also facilitates easier development of visually appealing animations to the checkout, which could further enhance the customer experience.

Finally, I would have liked to have been more thorough in my adoption of SCRUM Agile. Perhaps utilising a project management tool, such as Jira, for both a cleaner and more documented history of development. This would have also been useful for writing my implementation chapter as the clear blocks in development and most time-consuming tasks would have been easily identifiable within Jira.

10 Further Work and Conclusions

10.1 Further Work

10.1.0.1 Dispenser Improvements

The presentation of the dispensers could be improved by encasing the wires to reduce the potential for water damage and improve safety, particularly on the scale as it will be most susceptible to spillages. Additionally, the wires could be soldered to prevent them from easily coming loose. The surface area of the load cell's casing could also be increased to facilitate easier dispensing when using large containers.

10.1.0.2 Monitor Reserve Stock

Reserve stock could be monitored, perhaps through microcontrollers attached to scales labelled by product name, measuring the remaining weight of reserve stock and sending results to InfluxDB. This could be expanded to track expiry dates and notify employees what products they should push to sell to prevent

waste.

10.1.0.3 MySQL, Containerization, and Edge-Cloud Computing

The SQLite database can be converted to a MySQL database server, providing greater scalability, and allowing for both containerization and cloud hosting, required for the further work proposed below.

Docker could then be used to containerize Flask, MySQL, Node-RED, Mosquitto, and InfluxDB, ensuring the project functions the same on any machine with minimal setup. In a real-world context this also promotes a standardized configuration across all stores.

Alternatively, a hybrid edge and cloud approach could be implemented. For example, Node-RED, MQTT, and InfluxDB could be containerized locally within each store. Whilst Flask and MySQL are hosted in the cloud, delivering an expanded self-checkout app which can handle cross-store data management and payment processing. This approach would preserve local processing speeds of dispensing data, while improving data security and scalability by keeping sensitive data off-site, and off-loading some computation to the cloud **(Shrivastava, 2024)** [43]. This would perhaps be most appropriate for a chain of zero-waste stores, as it will keep them connected while maintaining a level of separation between the business dashboards and dispenser communications.

10.1.0.4 Artificial Intelligence (AI) Forecasting

AI models could be applied to dispense and capacity data to forecast times products might deplete and predict peak operating hours. The main goal of these insights is to optimize staffing and task distribution, while reducing product shortages. 94% of retailers have reported reduced operating costs from AI adoption **(Cummings, 2025)** [9], so its inclusion could yield similar results.

10.2 Conclusions

In conclusion, the aim to develop a prototype IoT smart dispensing system for zero-waste stores was achieved. Completing remaining requirements, along with proposed future work, would help to reach a fully refined solution. Consistent work and feedback from meetings with my supervisor **(Appendix H, Figure 4)** helped to ensure timely completion of the project's main functionalities and resolve blocks in development.

References

[1] Alanazi, M.F., Shahein, M.I., Alsharif, H.M., Alotibi, S.M., Alanazi, A.O., Alanazi, A.O., Alharbe, U.A., Almfalh, H.S.S., Amirthalingam, P., Hamdan, A.M., Veeramani, V.P., Mohamed, S.H.P., Ali, M.A.S. (2022) Impact of Automated Drug Dispensing System on Patient Safety. ***Pharm Pract (Granada)*** [online]. 20(4). [Accessed 20 October 2024].

[2] AliExpress (2025) ***AliExpress***. Available from: `https://www.aliexpress.com` [Accessed December 2024].

[3] AutomatedSteven (2023) Arduino Automated Water Dispenser. ***YouTube*** [video]. 9 February. Available from: `https://youtube.com/shorts/DIon7ecjTdo?feature=shared` [Accessed 5 March 2025].

[4] Baldini, E., Chessa, S. and Brogi, A. (2023) Estimating the Environmental Impact of Green IoT Deployments. ***Sensors*** [online]. 23(3). [Accessed 01 November 2024].

[5] Bottle (2024) ***Bottle: Python Web Framework***. Available from: `https://bottlepy.org/docs/dev/` [Accessed November 16 2024].

[6] Capital One Shopping (2024) ***Self-Checkout Adoption & Theft Statistics***. Available from: `https://capitaloneshopping.com/research/self-checkout-statistics#:~:text=Retailer%20Self-Checkout%20Adoption%20Statistics%201%2075.5%25%20of%20self-checkout,systems%20market%20represents%2044.4%25%20of%20the%20global%20market` [Accessed on 05 November 2024].

[7] Circuit Digest (2018) Automatic Water Dispenser Project Using Arduino. ***YouTube*** [video]. 14 August. Available from: `https://www.youtube.com/watch?v=5F8OYaEcO2Y` [Accessed 5 March 2025].

[8] Costa, F., Geonovesi, S., Borgese, M., Michel, A., Dicandia, F.A. and Manara, G. (2021) A Review of RFID Sensors, the New Frontier of Internet of Things. ***Sensors*** [online]. 21(9). [Accessed 21 October 2024].

[9] Cummings, A.R. (2025) AI in Retails: Use Cases, Examples & Adoption. ***Shopify: Retail Tips & Trends*** [blog]. 11 March. Available from: https://www.shopify.com/uk/retail/ai-in-retail [Accessed 05 March 2025].

[10] Deloitte (2020) ***Impact of COVID-19 on Cybersecurity***. Available from: `https://www.deloitte.com/ch/en/pages/risk/articles/impact-covid-cybersecurity.html` [Accessed on 05 November 2024].

[11] Do, Q.H., Hosseyni, P., Küsters, R., Schmitz, G., Wenzler, N. and Würtele, T. (2022) A Formal Security Analysis of the W3C Web Payment APIs: Attacks and Verification. ***IEEE Symposium on Security and Privacy (SP)*** [online]. pp 215-234. [Accessed 16 November 2024].

[12] Electronics Champ (2022) Can Arduino Measure Height of Tall Buildings? ***YouTube*** [video]. 1 October. Available from: `https://www.youtube.com/watch?v=DZfnqeiszpc` [Accessed 5 March 2025].

[13] Elijah, O., Rahman, T.A., Orikumhi, I., Leow, C.Y. and Hindia M.N. (2023) An Overview of Internet of Things (IoT) and Data Analytics in Agriculture: Benefits and Challenges. ***IEEE Internet of Things Journal*** [online]. 5(5). [Accessed 16 October 2024].

[14] Fung, E.Y., Leung, B., Hamilton, D., Hope, J. (2009) Do Automated Dispensing Machines Improve Patient Safety? ***The Canadian Journal of Hospital Pharmacy*** [online]. 62(6). [Accessed 20 October 2024].

[15] Ghimire, D. (2020) Comparative Study on Python Web Frameworks: Flask and Django [online]. ***Beng, Metropolia University of Applied Sciences***. [Accessed 06 November 2024].

[16] Holden, D. (2024) The Rise of Self-Checkouts and How They've Changed Retail. ***The Payments Association*** [online]. 26 September. Available from: `https://thepaymentsassociation.org/article/the-rise-of-self-checkouts-and-how-theyve-changed-retail/` [Accessed on 14 November 2024].

[17] Houghton, A. (2024) Are Supermarkets Going Cashless? The Full List Ditching Cash Payments in 2024. ***TimeOut*** [online]. 10 September. Available from: `https://www.timeout.com/uk/news/full-list-of-uk-supermarkets-ditching-cash-payments-this-year-from-tesco-to-asda-09102` [Accessed on 05 November 2024].

[18] Jara, A.J., Zamora, M.A. and Skarmeta, A.F.G. (2011) An Internet of Things Based Personal Device for Diabetes Therapy Management in Ambient Assisted Living (AAL). ***Personal and Ubiquitous Computing*** [online]. 15, pp.431-440. [Accessed 21 October 2024].

[19] Jia, X., Feng, Q., Fan, T. and Lei, Q. (2012) RFID Technology and its Applications in Internet of Things (IoT). ***2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*** [online]. pp. 1282-1285. [Accessed 21 October 2024].

[20] Jungle Culture (2019) ***Are Zero Waste Shops Expensive or Cheaper?***. Available from: `https://jungleculture.eco/blogs/news/are-zero-waste-shops-expensive` [Accessed 01 November 2024].

[21] Kaur, M., Sandhu, M., Mohan, N. and Sandhu, P.S. (2011) RFID Technology Principles, Advantages, Limitations & Its Applications. ***International Journal of Computer and Electrical Engineering*** [online]. 3(1), pp. 1793-8163. [Accessed 21 October 2024].

[22] Kavya, K.K., Akunuru, K.A., Pravalika, B.P., Krishna, B.S.K. and Radhakrishna, K.R. (2021) Automatic Water Dispenser Using Arduino. ***International Journal of Creative Research Thoughts*** [online]. Available from: `https://www.researchgate.net/publication/357420134_AUTOMATIC_WATER_DISPENSER_USING_ARDUINO` [Accessed 15 October 2024].

[23] Kemenes, P. (2024) Is Stripe safe? How secure is Stripe for payments? ***Wise Business*** [blog]. 19 July. Available from: `https://wise.com/us/blog/is-stripe-safe` [Accessed November 15 2024].

[24] Khan, C. (2023) I Long to Destroy Self-Checkout Machines – and at Last, There's a Glimmer of Hope. ***The Guardian*** [online]. 14 November. Available from: `https://www.theguardian.com/commentisfree/2023/nov/14/destroy-self-checkout-machines-supermarket-boycott` [Accessed on 10 November 2024].

[25] Kondapalli, J.K., Senepu, V.R., Kothapalli, B.S., Peketi, S.P.R. and Kukatla, V.D.N. (2019) Automatic Pet Feeder Using Internet of Things. ***Journal of Emerging Technologies and Innovative Research*** [online]. 6(4). [Accessed 15 October 2024].

[26] Kulothungan, S., Kamalakannan, K. and Thirugnanam, P. (2018) Agriculture Robot for Irrigation and Automation. ***Bulletin of Pure & Applied Sciences-Geology*** [online]. 37(1). [Accessed 15 October 2024].

[27] Kumar, S. (2023) 10 Beneficial Reasons to Shop at Zero Waste Stores. ***Chasing Verde*** [blog]. 16 October. Available from: `https://www.chasingverde.com/10-beneficial-reasons-to-shop-at-zero-waste-stores/` [Accessed 01 November 2024].

[28] Mack, I. (2024) A Beginner's Guide to Shopping at a Zero Waste Store. ***Party Kit Network*** [blog]. 7 September. Available from: `https://www.partykitnetwork.org/post/a-beginner-s-guide-to-shopping-at-a-zero-waste-store` [Accessed 01 November 2024].

[29] Mauladi, K.F., Laut Mertha Jaya, I.M. and Esquivias, M.A. (2022) Exploring the Link Between Cashless Society and Cybercrime in Indonesia. ***Journal of Telecommunications and the Digital Economy*** [online]. 10(3), pp.58-76. [Accessed 05 November 2024].

[30] Medrano-Galarza, C., LeBlanc, S.J., Jones-Bitton, A., DeVries, T.J., Rushen, J., Marie de Passillé, A. and Haley, D.B. (2017) Producer Perceptions of Manual and Automated Milk Feeding Systems for Dairy Calves in Canada. ***Canadian Journal of Animal Science*** [online]. 98(2), pp. 250-259. [Accessed 20 October 2024].

[31] Morchid, A., Jebabra, R., Ismail, A., Khalid, H.M., Alami, R.E., Qjidaa, H. and Jamil, M.O. (2024) IoT-enabled Fire Detection for Sustainable Agriculture: A Real-Time System Using Flask and Embedded Technologies. ***Results in Engineering*** [online]. 23. [Accessed 16 November 2024].

[32] Munoz-Ausecha, C., Ruiz-Rosero, J. and Ramirez-Gonzalez, G. (2021) RFID Applications and Security Review. ***Computation*** [online]. 9(69). [Accessed 21 October 2024].

[33] NCR Voyix (2025) ***NCR Voyix Self-Checkout - Where Innovation Meets Effortless Transactions***. Available from: `https://www.ncrvoyix.com/retail/self-checkout` [Accessed 15 March 2025]

[34] Nesbo, E. (2023) Will Self-Checkout Really Eliminate Jobs?. ***Make Use Of*** [online]. 30 January. Available from: `https://www.makeuseof.com/will-self-checkout-really-eliminate-jobs/` [Accessed 05 November 2024].

[35] Niranjanamurthy, M. (2014) E-Commerce: Recommended Online Payment Method – Paypal. ***International Journal of Computer Science and Mobile Computing*** [online]. 3(7), pp.669-679. [Accessed 10 November 2024].

[36] Ogbeiwi, O. (2017) Why Written Objectives Need to Be Really SMART. ***British Journal of Healthcare Management*** [online]. 23(7). [Accessed 01 December 2024].

[37] Philip, J., Abraham, F.M., Giboy, K.K., Feslina, B.J. and Rajan, T. (2020) Automatic Medicine Dispenser Using IoT. ***International Journal of Engineering Research & Technology*** [online]. 9(8), pp. 342-349. [Accessed 16 October 2024].

[38] Profitable Venture (2024) ***How to Start a Zero Waste Store Business That Makes Money***. Available from: `https://www.profitableventure.com/start-zero-waste-store-business/` [Accessed 01 March 2025].

[39] Random Nerd Tutorials (2025) ***Arduino with Load Cell and HX711 Amplifier (Digital Scale)***. Available from: `https://randomnerdtutorials.com/arduino-load-cell-hx711/#calibrate-load-cell` [Accessed 10 March 2025].

[40] Re-Gen Waste (2023) ***The Benefits of Zero Waste Shops***. Available from: `https://regenwaste.com/re-gen-waste-news/the-benefits-of-zero-waste-shops/` [Accessed 01 November 2024].

[41] Rochemont, S. (2018) Environmental Sustainability of a Cashless Society. ***Institute of Faculty of Actuaries*** [online]. 21. [Accessed 05 November 2024].

[42] Shapsough, S., Takrouri, M., Dhaouadi, R. and Zualkernan, I.A. (2021) Using IoT and Smart Monitoring Devices to Optimize the Efficiency of Large-Scale Distributed Solar Farms. ***Wireless Networks*** [online]. 27, pp. 4313-4329. [Accessed 01 November 2024].

[43] Shrivastava, A. (2024) Edge Computing vs Cloud Computing: Transforming Strategies & Benefits. ***Scrobits*** [online]. 8 August. Available from: `https://www.scrobits.com/blog/edge-computing-vs-cloud-computing-transforming-strategies-benefits` [Accessed 05 March 2025].

[44] Smith, A.D. (2010) Retail-based Loyalty Card Programmes and CRM Concepts: An Empirical Study. ***International Journal of Innovation and Learning*** [online]. 7(3), pp.303-330. [Accessed 15 November 2024].

[45] Somers, J. (2024) Zero-Waste Stores Aren't as Good as People Think. ***Lifehacker*** [online]. 21 June. Available from: `https://lifehacker.com/money/zero-waste-stores-arent-as-good-as-people-think` [Accessed 01 November 2024].

[46] Stripe (2024) ***The Market Leader for Platform and Marketplace Payments***. Available from: `https://stripe.com/gb/connect` [Accessed 10 November 2024].

[47] TechTarget (2023) ***MoSCoW Method***. Available from: `https://www.techtarget.com/searchsoftwarequality/definition/MoSCoW-method` [Accessed 04 December 2024].

[48] Too Good to Go (2024) ***Save Good Food from Going to Waste***. Available from: `https://www.toogoodtogo.com/` [Accessed 15 November 2024].

[49] Townley, G. (2023) Going Green is Good for Business. ***Companies House*** [blog]. 21 July. Available from: `https://companieshouse.blog.gov.uk/2023/07/21/going-green-is-good-for-business/` [Accessed 15 November 2024].

[50] Ulyanida, S., Supriyanto, A., Suciyati, S. and Junaidi, J. (2022) Automatization of Weight and Height Measurement Using Ultrasonic Sensors HC-SR04 and Load Cell Based on Arduino UNO at Integrated Services Posts (Posyandu). ***Journal of Energy Material and Instrumentation Technology*** [online]. 3(4), pp. 127-137. [Accessed 16 October 2024].

[51] Wang, S., Zeng, S., Yang, Y. and Jian, L. (2022) Application of Automatic Dispensing Machine in Outpatient Pharmacy and its Advantage and Disadvantage Analysis. ***4th International Conference on Frontiers of Biological Sciences and Engineering*** [online]. 2511(1). [Accessed 20 October 2024].

[52] Zhadan, A. (2023) Analysis: Is AI Coming for Retail Jobs?. ***Charged Retail*** [online]. 22 March. Available from: `https://www.chargedretail.co.uk/2023/03/22/is-ai-coming-for-retail-jobs/` [Accessed 10 November 2024].

[53] Zhai, C., Zou, Z., Zhou, Q., Mao, J., Chen, Q., Tenhunen, H., Zheng, L. and Xu, L. (2016) A 2.4-GHz ISM RF and UWB Hybrid RFID Real-Time Locating System for Industrial Enterprise Internet of Things. ***Enterprise Information Systems*** [online]. 11(6), pp. 909-926. [Accessed 21 October 2024].

A Methodology

A.1 Gantt Chart Tasks

TASK	PROGRESS	START	END
Project Preparation			
Gather details and order the hardware required	0%	30/9/24	8/10/24
Begin collecting initial background research	0%	30/9/24	2/10/24
Consider the initial aims & objectives of the project	0%	30/9/24	4/10/24
Draft of project proposal	0%	7/10/24	12/10/24
Finalise project proposal ready for submission	0%	13/10/24	24/10/24
Report			
Gather references required for literature review	0%	7/10/24	14/10/24
Draft of literature review	0%	15/10/24	30/10/24
Final draft of literature review	0%	31/10/24	15/11/24
Generate requirements based off of the literature review	0%	16/11/24	24/11/24
Expand requirements, rank using MoSCoW and classify as functional/non-functional	0%	25/11/24	30/11/24
Finalise requirements	0%	1/12/24	20/1/25
Draft of introduction	0%	25/11/24	20/2/25
Create design diagrams (use case, circuit diagrams, ERD diagram, etc.)	0%	1/12/24	15/2/25
Methodology section draft	0%	20/1/25	10/2/15
Design section draft	0%	15/1/25	15/2/25
Implementation draft	0%	1/2/25	15/3/25
Finalise methodology, design and implementation section	0%	15/3/25	25/3/25
Project evaluation draft	0%	26/3/25	5/4/25
Further work and conclusions draft	0%	26/3/25	5/4/25
Final draft of introduction	0%	6/4/25	10/4/25
Final draft of future work and conclusions	0%	6/4/25	10/4/25
Format report appropriately (table of contents, table of figures, references, etc.)	0%	10/4/25	15/4/25
Finalise report ready for submission	0%	16/4/25	25/4/25
Smart Dispensing System			
Gather details and order the required hardware	0%	15/10/24	5/11/24
Set up microcontroller and test different hardware	0%	6/11/24	12/11/24
Dispensing system can read RFID tags	0%	13/11/24	20/11/24
Dispensing system weighs dispensed product	0%	13/11/24	20/11/24
System can measure contents left in reserve	0%	15/1/25	1/2/25
System dispenses based on a containers height	0%	21/11/24	1/2/25
System can send dispenses to external systems	0%	21/11/24	1/2/25
System can send reserve capacity and dispense details to the business dashboard	0%	2/2/25	8/2/25
Test the finished dispensing system	0%	9/2/25	1/3/25
Prepare system for viva	0%	25/4/25	5/5/25
Checkout System			
RFID tag can be scanned and identified by the system	0%	10/10/24	15/10/24
RFID specific baskets can be viewed	0%	16/10/24	20/10/24
Basket contents can be modified (add/delete purchases, add items)	0%	21/10/24	27/10/24
System can take customer payments via card	0%	28/10/24	3/11/24
Receipts displayed to users with the option to send email confirmation	0%	4/11/24	1/2/25
Features for customer loyalty such as paying with points and RFID tag specific customer accounts	0%	4/11/24	24/11/24
Admin pages (modifying customer accounts, adding new available items)	0%	25/11/24	15/12/24
Security features (requiring employee ID to delete/cancel purchases and access admin pages)	0%	25/11/24	1/2/25
Test the finished checkout system	0%	2/2/25	1/3/25
Prepare system for viva	0%	25/4/25	5/5/25
Business Dashboard			
Reserve capacity is correctly displayed on the dsahboard	0%	3/2/25	16/2/25
Dashboard displays interactable graphs of customer shopping trends (e.g. amounts dispensed by hour/week/month)	0%	17/2/25	2/3/25
Test the finished business dashboard	0%	3/3/25	10/3/25
Prepare dashboard for viva	0%	25/4/25	5/5/25

A.2 Risk Register

Risk ID	Description	Impact	P	I	S	Triggers	Mitigation Strategy
R1	I am late submitting the project	I will fail the module	2	5	10	• Ineffective time-management • Illness • Not meeting with supervisor	• Utilise Gantt chart to keep on schedule • Meet consistently with supervisor
R2	System works as needed but isn't accessible for zero waste stores	The effectiveness of the project will be diminished	3	4	12	• Poor research into cost-effective solutions • Using high-cost hardware	• Ensure hardware remains low-cost • Monitor the price as features are implemented
R3	I focus too heavily on one side of the project (hardware/software) but neglect the report	My report will suffer in quality if I have less time to complete it	3	3	9	• Neglecting work I find less enjoyable • Perfectionism in specific sections	• Ensure I am working consistently accross all sections of the project • Utilise Gantt chart to keep on schedule
R4	The delivery of the hardware for the project is delayed	It will give me tighter deadlines to adhere to	4	3	12	• External issues with suppliers • Certain hardware being hard to source	• Order hardware as soon as it is clear it's needed • Order spares where viable

Risk ID	Description	Impact	L	S	Score	Triggers	Mitigation Strategy
R5	I require new hardware mid-development	It will give me tighter deadlines to adhere to	4	4	16	• Shallow research into hardware • Unforseen new requirements • Overpromising on hardware performance online	• Be verbose during requirement analysis • Focus on other areas of the project if waiting for parts
R6	The hardware cost of the project is too expensive	I will not be able to purchase particular components	2	4	8	• Lower cost components not being accurate enough • No available alternatives to the expensive hardware	• Check datasheets for components • Adjust hardware to focus on achievable goals at a lower cost • Adjust project requirements to work around requiring expensive hardware
R7	The project fails to comply with regulations (e.g., PCI compliance)	Businesses would be fined, lose reputation	2	5	10	• Not researching approriate handling of card payments	• Ensure the project uses approved payment transaction handling • Take extra care when developing payment related features

Risk ID	Description	Impact	L	S	Score	Triggers	Mitigation Strategy
R8	The project handles customer data without following GDPR laws	Can result in huge fines and even prison time	2	5	10	• Not knowing how to approriately handle customer data	• Ensure customer data is handled approriately • Ask for permission to store any data
R9	The system has multiple bugs upon completion	Project effectiveness diminished, possibly unusable	3	4	12	• Developing without testing • Using unstable/deprecated tools	• Ensure iterative testing throughout development • Create a thorough test plan • Utilise unit testing
R10	The concepts in creating a hardware project are difficult for me to grasp	Greater challenge in development and the risk of a sub-par project	4	3	12	• Compex concepts • No prior experience • Short time to learn	• Study consistently • Do not get discouraged
R11	Working with liquids and electricity could raise safety issues	Components could be damaged or cause electric shocks	3	4	12	• Poor precautions taken • Liquid near components • spillages	• Separate liquids from components • Turn off system when working on electrical parts

Risk ID	Description	Impact	L	S	Score	Triggers	Mitigation Strategy
R12	Testing is limited compared to real-world use	System could encounter unforeseen errors in production	4	4	16	• Difficulty replicating production environment	• Be as thorougoh as possible during testing • Attempt to capture scalability of the project in testing • Consider scalability in development • Consider common customer actions carefully
R13	System corruption without appropriate backups	Project setback due to potential data or config loss	2	4	8	• Storage drive corruption • Hardware damage	• Use Git for version control • Maintain backups of configurations and services
R14	Poor quality human-computer interaction (UI/UX)	Customer experience is harmed, reducing effectiveness	3	4	12	• Poor design planning • Ignoring accessibility guidelines	• Develop accessible designs • Thorough planning including wireframes

Risk ID	Description	Impact	L	S	Score	Triggers	Mitigation Strategy
R15	Project scope grows too large for time/resources	Risk of an unfinished/unpolished large-scale project	3	3	9	• Requirement creep • Poor time managements	• Set clear project guidelines early • Discuss scope changes with supervisor

P = Risk Probability (Scored between 1-5)

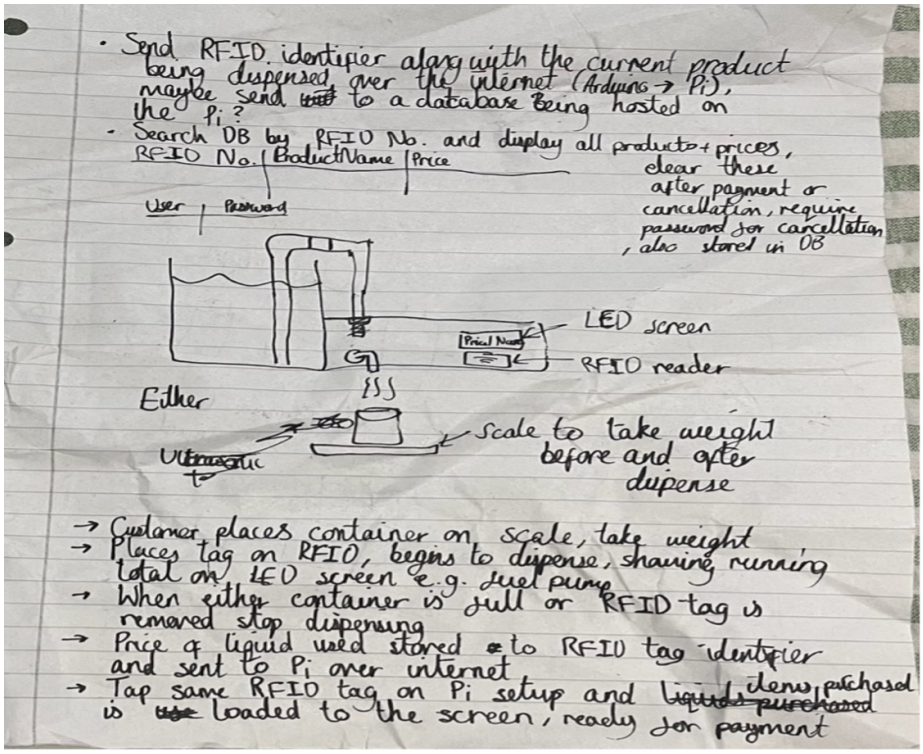
I = Risk Impact (Scored between 1-5)

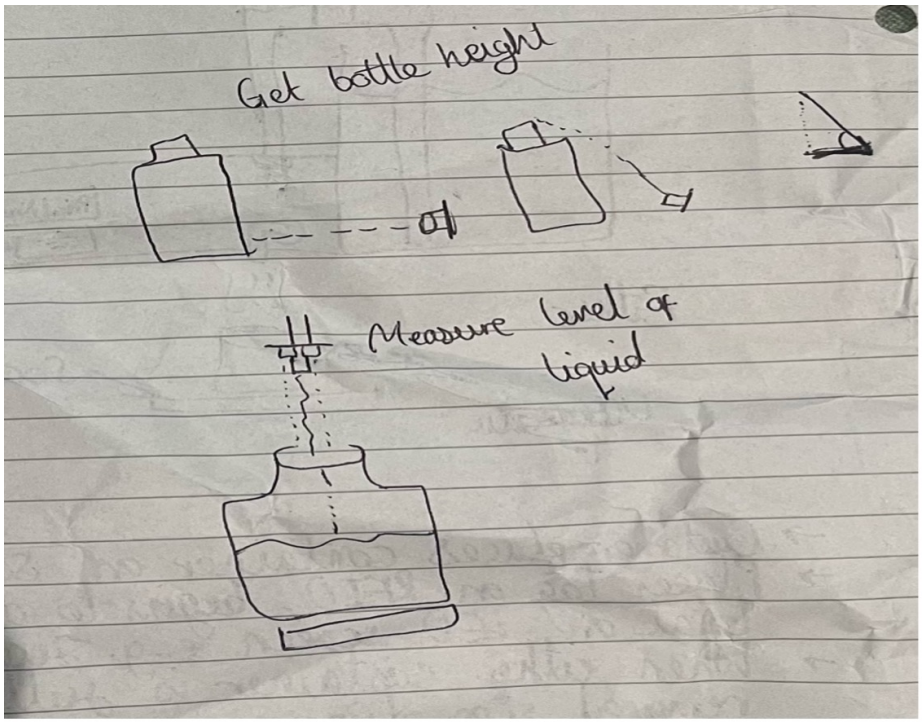
S = Score

$$Score = Risk\ Probability \times Risk\ Impact$$

B Design

B.1 Initial Paper Plans





B.2 Microcontroller Comparison

Name	Clock Speed	RAM	Flash Memory	Key Features	Cost	Comments	Plan to Use?
Arduino Uno	16MHz	2KB	32KB	• N/A	£2.69	• No built in Wi-Fi functionality • Small number of pins	No
Pi Zero W	1GHz	512MB	microSD	• Wi-Fi • Bluetooth • 40 GPIO pins • HDMI output	£15.10	• Perhaps overkill for the project's requirements • Higher cost than ESP32	No

Name	Clock Speed	RAM	Flash Memory	Key Features	Cost	Comments	Plan to Use?
ESP32 WROOM-32	240MHz	520KB	4MB	• Wi-Fi • Bluetooth • 34 GPIO pins • Low power mode	£2.69	• Low cost • Power efficient • Plenty of GPIO pins • Fast processing speed • Built-in Wi-Fi capability	Yes

Data obtained from respective microcontroller's datasheet, with costs based on prices shown at **AliExpress (2025)**[2].

B.3 Distance Sensor Comparison

Name	Voltage	Key Features	Cost	Comments	Plan to Use/Test?
HC-SR04	3.3-5v	• Works with most surfaces and environments • Good accuracy	£0.86	• Low cost • Potential to be effective in measuring capacities of both liquids and solids	Yes

Name	Voltage	Key Features	Cost	Comments	Plan to Use/Test?
VL53L1X	3.3-5v	• Very high accuracy	£2.69	• Relatively high cost • Low measuring angle and high accuracy • Potential to be effective in detecting container heights and liquid level • Might struggle with glass containers due to using infrared technology	Yes
JSN-SR04T	5v	• Waterproof Waterproof	£2.18	• Relatively high cost • Similar to ultrasonic but waterproof • Only measures distances less than 20cm	No

Data obtained from respective peripheral's datasheet, with costs based on prices shown at **AliExpress (2025)**[2].

B.4 Display Comparison

Name	Type	Resolution	Cost	Comments	Plan to Use?
SSD1306	OLED	128x64	£1.69	• Low cost • Can display many characters at one time • Can display images • Will not suffer the visibility issues of the 1601 • Contrast of text is good for accessibility • Most approriate choice for this project	Yes
ILI9341	TFT LCD	240x320	£4.15	• Perhaps overkill for the project's requirements • High resolution • Much more expensive than the SSD1306 • Can display images	No
GC9A01	IPS LCD	240x240	£2.55	• Perhaps overkill for the project's requirements • Round display could hinder displaying of text • Can display images	No

Name	Type	Resolution	Cost	Comments	Plan to Use?
1602	LCD	16x2 characters	£0.75	• Lowest cost and power requirements • Cannot display images but isn't essential for my project • Limited space to show text is a large drawback • Will consider this for the solids dispenser to verify its approriateness in real-world use	Yes

Data obtained from respective peripheral's datasheet, with costs based on prices shown at **AliExpress (2025)**.

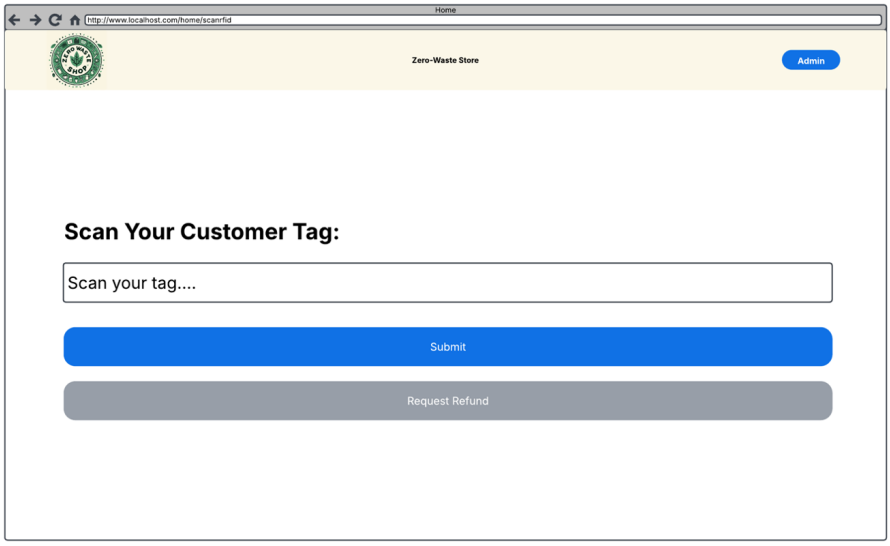
B.5 Water Pump Comparison

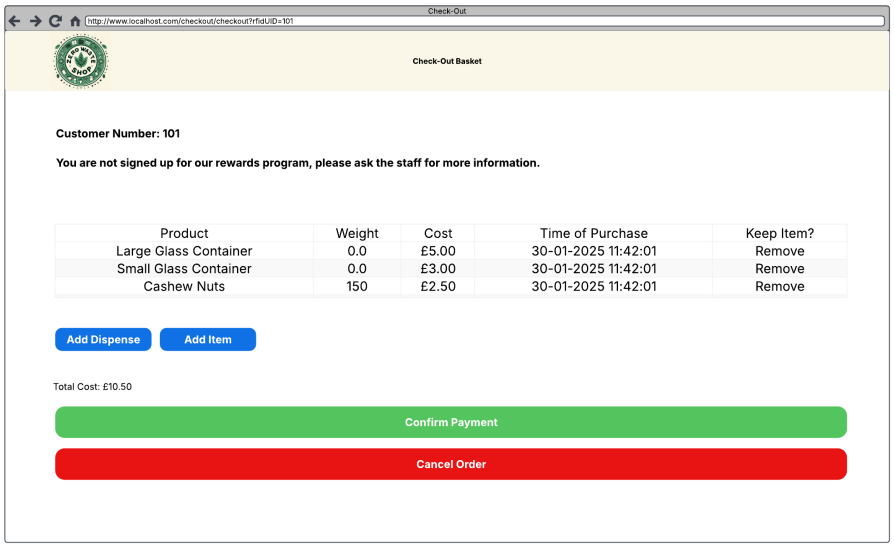
Name	Key Features	Cost	Comments	Plan to Use?
5V Submersible Pump	• Small pump • Low flow rate • Lower noise level • Low voltage requirement	£0.44	• Will dispense slowly • Might struggle with more viscous liquids • Liquid travels through the pump itself which could cause blockages • Must be submerged in the liquid to pump	No

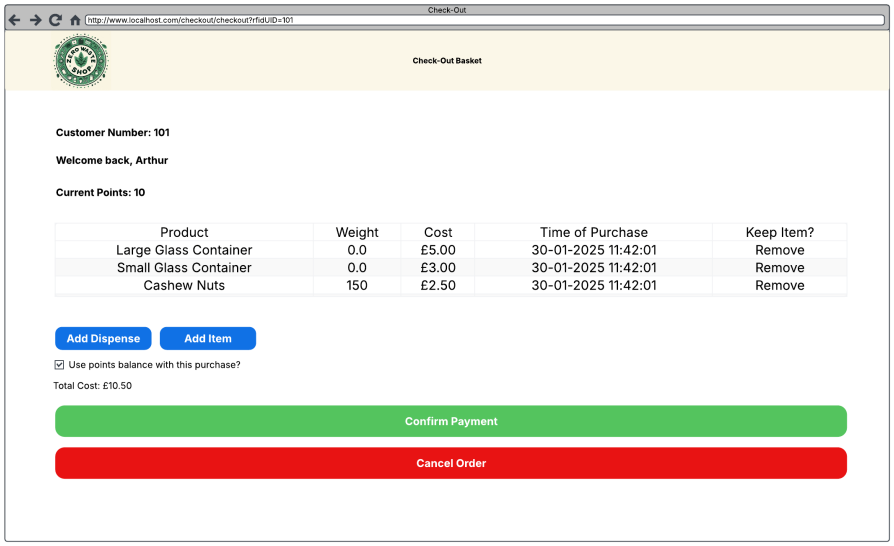
Name	Key Features	Cost	Comments	Plan to Use?
12V Peristaltic Pump	• Bigger pump • Higher flow rate • Louder noise level • High voltage requirement	£3.40	• Will allow for quicker displensing of liquid, which will improve dispensing times • Doesn't need to be submerged to pump liquid, making separation from sensitive electronics simple • 24V pump might be beyond project requirements, 12V is a nice middle ground • Will create more complex power requirements within the liquid dispenser circuit • I believe the benefits outweight the additional costs and complexity	Yes

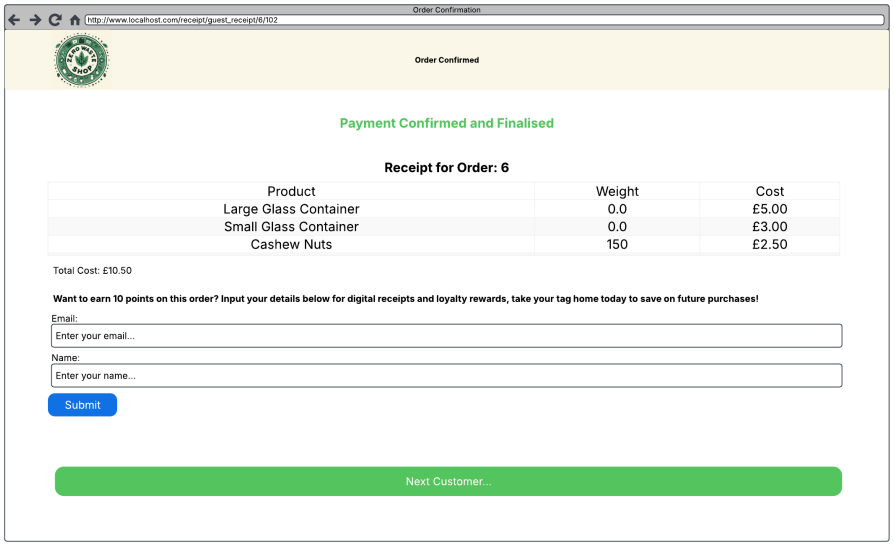
Data obtained from respective peripheral's datasheet, with costs based on prices shown at **AliExpress (2025)**.

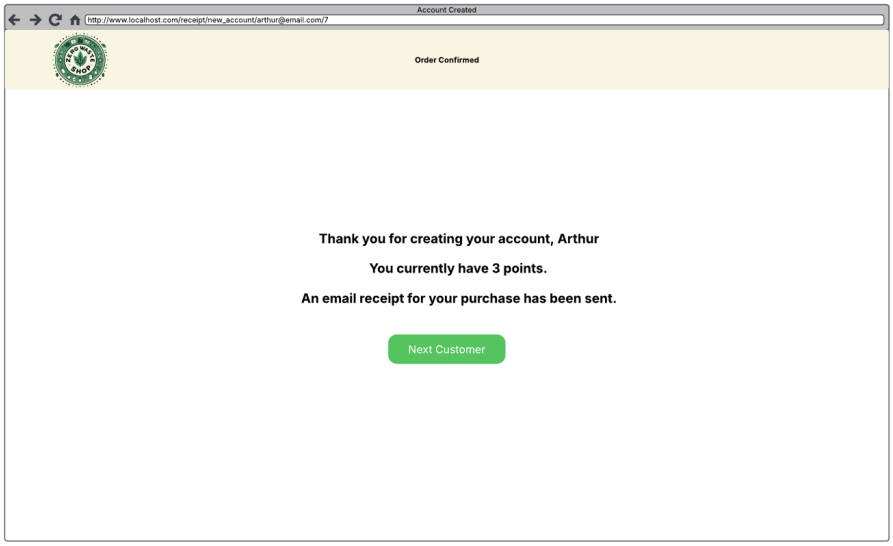
B.6 Self-Checkout Mock-Ups

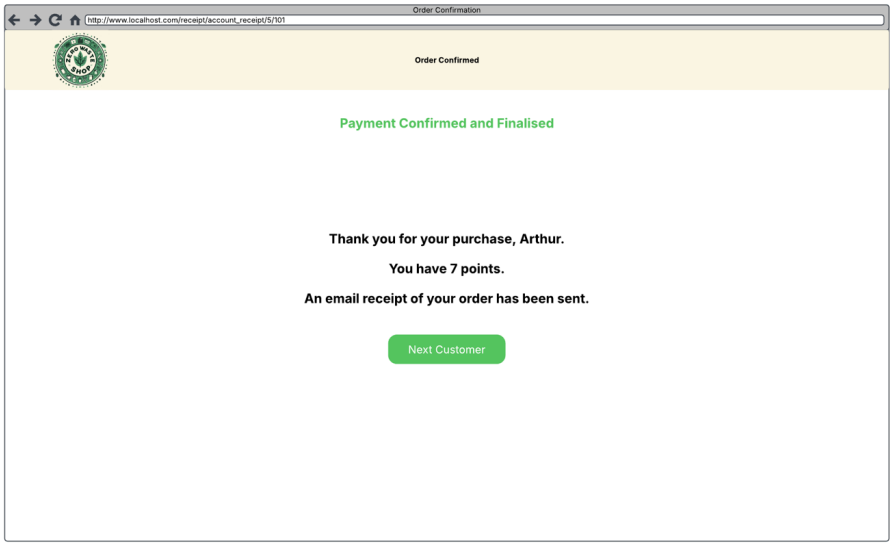


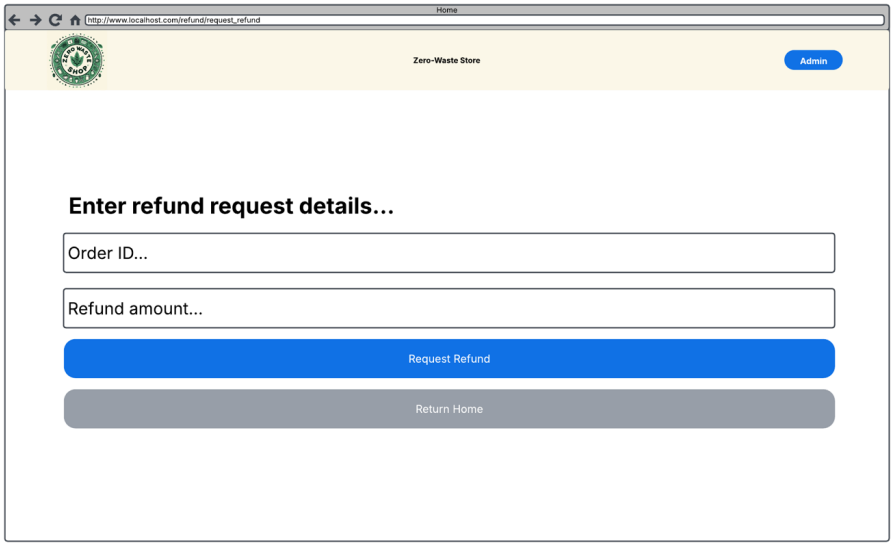



Check-Out
http://www.localhost.com/checkout/checkout?rfidUID=101
Check-Out Basket
Customer Number: 101
Welcome back, Arthur
Current Points: 10
Product | Weight | Cost | Time of Purchase | Keep Item?
Large Glass Container | 0.0 | £5.00 | 30-01-2025 11:42:01 | Remove
Small Glass Container | 0.0 | £3.00 | 30-01-2025 11:42:01 | Remove
Cashew Nuts | 150 | £2.50 | 30-01-2025 11:42:01 | Remove
Add Dispense
Add Item
Use points balance with this purchase?
Total Cost: £10.50
Confirm Payment
Cancel Order


Order Confirmation
http://www.localhost.com/receipt/guest_receipt/6/102
Order Confirmed
Payment Confirmed and Finalised
Receipt for Order: 6
Product | Weight | Cost
Large Glass Container | 0.0 | £5.00
Small Glass Container | 0.0 | £3.00
Cashew Nuts | 150 | £2.50
Total Cost: £10.50
Want to earn 10 points on this order? Input your details below for digital receipts and loyalty rewards, take your tag home today to save on future purchases!
Email:
Enter your email...
Name:
Enter your name...
Submit
Next Customer...


Account Created
http://www.localhost.com/receipt/new_account/arthur@email.com/7
Order Confirmed
Thank you for creating your account, Arthur
You currently have 3 points.
An email receipt for your purchase has been sent.
Next Customer


Order Confirmation
http://www.localhost.com/receipt/account_receipt/5/101
Order Confirmed
Payment Confirmed and Finalised
Thank you for your purchase, Arthur.
You have 7 points.
An email receipt of your order has been sent.
Next Customer



B.7 Initial Test Plan

Test Case ID	Priority	Test Case Type	Test Description	Test Step	Expected Result
1	High	Functionality	All links redirect to the correct page.	Test the destination of every hyperlink.	Hyperlinks redirect the user to the intended pages.
2	High	Security	Customers cannot remove an item from their sale without employee approval.	Ensure removing an item from a sale prompts an employee pin and that pin is correctly validated.	Attempting to remove an item will require an employee pin that is validated against the database.
3	High	Security	Customers cannot cancel their sale without employee approval.	Ensure cancelling a sale prompts an employee pin and that pin is correctly validated.	Attempting to cancel a sale will require an employee pin which is validated against the database.

Test Case ID	Priority	Test Case Type	Test Description	Test Step	Expected Result
4	Medium	Usability	The system will only read one RFID when multiple are present.	Hold multiple RFID tags to the reader and ensure there are no collisions.	Only one RFID UID will be read due to anti-collision and a delay between reading an RFID tag.
5	High	Functionality	The system calculates dispensed weight accurately (±10%).	Dispense a known weight or verify dispensed weight with an external scale against the systems reading.	The weight is accurately calculated.
6	High	Functionality	Dispense details are correctly passed to the checkout system.	Dispense known values and verify the details are sent over as expected on the checkout system.	The dispense details are correctly sent.
7	High	Security	Payments are successfully and securely processed by Stripe API.	Try different types of card details, including fake and real, to ensure expected behaviour, verify payment using the Stripe dashboard.	Payments details are correctly verified and charged.
8	High	Functionality	Unauthorised access to the admin panel is not allowed.	Attempt to access the admin panel without requiring an employee pin.	Access will be blocked until a valid pin is entered.

Test Case ID	Priority	Test Case Type	Test Description	Test Step	Expected Result
9	High	Security	Unauthorised access to the business dashboard is not allowed.	Attempt to access the business dashboard without logging-in, and with incorrect username and password.	Presented with log-in page and denied access.
10	High	Functionality	Capacity details are correctly passed to the capacity dashboard from the dispensing system.	Test a known capacity and ensure it displays/is saved appropriately within the capacity dashboard.	Details are correctly sent and displayed.
11	Medium	Functionality	If the dispensing system fails to calculate an appropriate height it will default to manual dispensing.	Test heights which are infeasible (< MIN HEIGHT and > MAX HEIGHT) and see if it manually dispenses.	Height calculation error will be displayed to the user and manual dispensing will begin.
12	Low	Usability	The current price visually updates correctly when a dispense has been changed.	Check the displayed current price after adding item to a dispense of a known cost, then check the same when removing items of a known cost.	Price will be displayed correctly based on the value of the item added to the dispense.
13	Medium	Security	An incorrect RFID UID cannot be used with the self-checkout.	Try to use the checkout with an invalid RFID UID.	RFID UID is validated against the database accordingly.

Test Case ID	Priority	Test Case Type	Test Description	Test Step	Expected Result
14	High	Security	Refunds are correctly validated by the checkout system before sending the refund request to Stripe.	Try requesting a refund with an invalid order ID, a refund with a negative refund amount, a refund with an amount higher than the order cost - (points used/100).	Refunds are correctly verified and refunded, any errors are correctly presented to the user.
15	High	Functionality	Stripe API successfully receives refund requests.	Send a refund request using the request refund page and verify the result using the Stripe dashboard.	The refund request is visible on the Stripe dashboard.
16	High	Functionality	When a customer cancels an order those items are removed from the database and their basket is cleared.	Create an order, cancel the order, verify the change in the database and scan the RFID tag again to see if the dispenses are still displaying visually.	The order will be cancelled, dispenses will be removed from the database, and it will no longer display in the RFID's basket.
17	High	Security	A customer cannot sign-up using an email that is already taken.	Verify a customer account email in the database then attempt to sign-up using this email.	Error will be displayed about the email already being registered.

Test Case ID	Priority	Test Case Type	Test Description	Test Step	Expected Result
18	High	Security	A customer cannot sign-up using an RFID UID that is already taken.	Verify a customer account RFID UID in the database then attempt to sign-up using this RFID UID.	Error will be displayed about the RFID UID already being registered.
19	Medium	Functionality	A customer cannot confirm payment with an empty basket.	Attempt to confirm payment whilst having no active purchases.	Error will be displayed about having no purchases in the basket.
20	High	Functionality	Real-time dispensing data is sent and displayed correctly on the business dashboard.	Send known dummy data to the business dashboard using the same ESP32 code as the dispensers.	Data is displayed correctly.
21	High	Usability	Test if multiple dispensers can connect under the same MQTT/self-checkout/dashboard.	Have multiple ESP32 running simultaneously and sending data through MQTT.	The dispenses are all recorded on both the self-checkout and dashboards.
22	Low	Functionality	Test the dispensers go into sleep mode outside of business hours.	Monitor a dispenser outside of 9:00am-5:30pm to ensure it goes into sleep mode and wakes up when it is back in business hours.	The dispenser remains in sleep mode until business hours resume.
23	High	Functionality	System correctly notifies employees of low capacity.	Send low stock levels to capacity dashboard and check for notifications.	Low capacity gives a warning to employees through the dashboard interface.

Test Case ID	Priority	Test Case Type	Test Description	Test Step	Expected Result
24	Medium	Usability	Customers with an account get email receipts on purchase confirmation.	Confirm a purchase with an account.	Email is sent with the correct purchase details.
25	Medium	Usability	Ensure the MQTT connection is stable on the dispensers throughout the entire dispensing flow.	Monitor MQTT connection and have multiple checks the connection is still present throughout the code.	MQTT connection has a 99% uptime.

B.8 Diagram-Requirements Mapping

Design Diagram	Related Requirements
Self-Checkout Wireframes	**All self-checkout wireframes relate to: MF1, CNF2**
Fig. 10 ('Home')	MF8
Fig. 11 (Guest view of the 'Checkout')	MF3, MF7, WF1
Fig. 12 (Guest view of the 'Receipt')	MF3, MF8, MF9
Fig. 13 ('Account Created')	MF8, SF3
Fig. 14 (Account view of the 'Checkout')	MF3, MF7, MF8, MF9, SF3, WF1
Fig. 15 (Account view of the 'Receipt')	MF3, SF3
Fig. 16 ('Refund Request')	MF3
Fig. 17 (Wireflow)	MF2, MF8, MF9, MNF2
Self-Checkout Mock-Ups	**All self-checkout mock-ups relate to: MF1**
Fig. 18 (Self-Checkout Site Map)	MF9, SF1, MFN1, MNF3
Dashboard Wireframes	**All dashboard wireframes relate to: SF2**

Design Diagram	Related Requirements
Fig. 19 ('Capacity Dashboard')	MF4
Fig. 20 ('Business Trends Dashboard')	MF5
Use Case Diagrams	**N/A**
Fig. 21 (Dispensing)	MF1, MF4, MF6, SF2
Fig. 22 (Self-Checkout)	MF2, MF3, MF7
Fig. 23 (Dashboard)	MF4, MF5, SF2
Figs. 24-26 (Customer Flow Diagram)	MF1, MF8, SF3, WNF2
Fig. 27 (System Architecture Diagram)	MF1, MF2, MNF1, MNF2, MNF3, WNF2
Fig. 28 (Data Flow Diagram)	MF1, MF2, MF4, MF5, SF1, MNF1, MNF3, WNF2
Fig. 29 (Dispensing State Diagram)	MF1, MF2, MF6, MF8, SF2, CF3, MNF1, MNF3, MNF4, CNF3, SNF2
Fig. 30 (Class Diagram)	MF3, MF8
Sequence Diagrams	**N/A**
Fig. 31 (Dashboard Sequence)	MF4, MF5, SF2
Fig. 32 (Self-Checkout Sequence)	MF2, MF3, MF7, MF8, MF9, SF3, CF3, MNF2, MNF3
Fig. 33 (ERD)	MF2, MF3, MF7, MF8, MF9, SF1, SF3, MNF1, MNF3
Circuit Diagrams	**All circuit diagrams relate to: CF3, SNF1, SNF2, WNF1**
Fig. 34 (RaspberryPi)	MNF1, MNF3
Fig. 35 (Solid Dispenser)	MF4, SF2, MNF3, MNF4

C Automatic Dispenser

C.1 SWOT Analysis of Automatic Dispensing

Strengths

- Assists user in filling container.
- Less dispenser interactions, improving ease-of-use.
- Potential to reduce spillages as automatic dispensing prevents the container from overflowing.

Weaknesses

- Limited accuracy due to variation in containers could lead to an increase in spillages rather than a decrease.
- Infrared-based sensors struggle with detecting glass containers.
- Increases the cost of the dispenser due to the additional peripherals required.
- Increases the likelihood of the dispenser failing due to the increase in sensitive peripherals.

Opportunities

- Use higher-cost sensors which provide much greater accuracy.
- A working automated dispenser could have uses outside of a zero-waste shop, such as in factories.

Threats

- To achieve consistent results, the cost could become greater than the aims of the project.
- The large variation in containers makes it very difficult to capture all possible dispenses.

C.2 Height Calculation Code

Below is extracted key code for calculating the height.

LIBRARIES USED

```
#include <mpu6050_FastAngles.h> //Library used for gyroscope
#include <Math.h> //For height calculation
#include <ESP32Servo.h> //Servo
```

PIN SETUP

```
//IR Sensor
#define IR_SENSOR_PIN 15
//Servo motor
#define SERVO_PIN 33

#define SDA_PIN 21
#define SCL_PIN 22
```

CONSTANT VARIABLES

```
    //Max angle of servo for height calculation
    const int MAX_SERVO_ANGLE = 90;
    //Error margin for height calculation
    const int ERROR_MARGIN = 2;
    //Delay between each movement of servo
    const int MOVE_DELAY = 200;
    //Delay between dispenses
    const float DISPENSE_DELAY = 15000;
    //The distance from the ultrasonic sensor to
    //the scale minus a few cm to allow for the base of containers
    const int BASE_DISTANCE = 18;
```

CODE INSIDE SETUP FUNCTION

```
    //IR Sensor
    pinMode(IR_SENSOR_PIN, INPUT);


    myServo.attach(SERVO_PIN);
    myServo.write(0);

    //Gyroscope
    mpu.begin(MPU_MODE_250);
    mpu.setComplementaryFactor(0.98);
    mpu.setKalmanQangle(0.001);
    mpu.setKalmanQbias(0.003);
    mpu.setKalmanRmeasure(0.03);

    display.clearDisplay();
    display.setCursor(0,0);
    display.println("Gyroscope ready.");
    display.display();
    delay(1000);
```

HEIGHT CALCULATION

```
    //Calculating height of the object, adjacent = opposite/tan(radians)
    //Storing ultrasonic sensor distance for size of opposite
    opposite = ultrasonic1.read();
    display.clearDisplay();
    display.setCursor(0, 0);
    display.println("Measured Opposite:");
    display.print(opposite);
    display.println("cm");
    display.display();
    delay(2000);

    int IRValue = digitalRead(IR_SENSOR_PIN);
```

```
    //Get initial angle before movement
    float initialAngle = mpu.getAngle('X', KALMAN);

    //Runs until IR sensor detects container
    while(IRValue == HIGH)
    {
    //Slowly moving servo and getting the X angle
    for (int angle = 0; angle <= MAX_SERVO_ANGLE; angle++) {
        myServo.write(angle); //Set servo to angle
        //Delay between movements to give time for sensor readings
        delay(MOVE_DELAY);
        //Read current value
        IRValue = digitalRead(IR_SENSOR_PIN);
        display.clearDisplay();
        display.setCursor(0, 0);
        //Gyroscope reads at 90 (+- a couple degrees) to begin,
        //so use this to get the angle from 0 degrees
        display.println((initialAngle-mpu.getAngle('X', KALMAN))+90);
        display.display();
        if(IRValue == LOW) //Detects container
        {
        delay(200);
        //Store current angle of gyroscope
        servoAngle = (initialAngle-mpu.getAngle('X', KALMAN))+90;
        //Stop for loop, do not store angle as previous angle will be
            ↪ more accurate
        break;
        }
        delay(100);
    }
    }

    //Once angle has been found calculate height
    //Converting the angle to radians, error margin of 2cm
    servoRadians = servoAngle * (M_PI / 180.0);
    height = opposite * tan(servoRadians);
    height = height - ERROR_MARGIN;
```

IF HEIGHT INVALID

```
    //If height calculated is higher than distance
    //between scale and ultrasonic OR less than 0
    if(height>=BASE_DISTANCE || height<=0)
    {
        display.clearDisplay();
        display.setCursor(0,0);
        display.println("Height calculation incorrect, will dispense
            ↪ manually...");
        display.display();
        dispenseAutomatically = false;
```

```
    }

    //If sensor above container reads height below
    //the base distance, query user to adjust their container

    //Distance lower than the base distance suggests
    //the sensor is hitting the side of the container
    //not the inside this only runs if user is automatically dispensing
    if(ultrasonic2.read()<BASE_DISTANCE)
    {
        while (true)
        {
            float currentHeight = ultrasonic2.read();

            //If distance is reading appropriately
            if(currentHeight>BASE_DISTANCE)
            {
            display.clearDisplay();
            display.setCursor(0,0);
            display.println("Thank you for adjusting your container...");
            display.println("Currently Detected Distance:");
            display.print(currentHeight);
            display.println("cm");
            display.display();
            delay(5000);
            break;
            }

            display.clearDisplay();
            display.setCursor(0,0);
            display.println("Currently Detected Distance:");
            display.print(currentHeight);
            display.println("cm");
            display.println("Please adjust your container...");
            display.display();

            delay(100);
        }
    }
```

OPTION TO MANUALLY DISPENSE

```
//Only query user if they have not already decided to manually dispense
if(dispenseAutomatically)
{
    //Display calculated height to dispense to
    display.clearDisplay();
    display.setCursor(0,0);
    display.println("Dispensing to Height:");
    display.println(height);
```

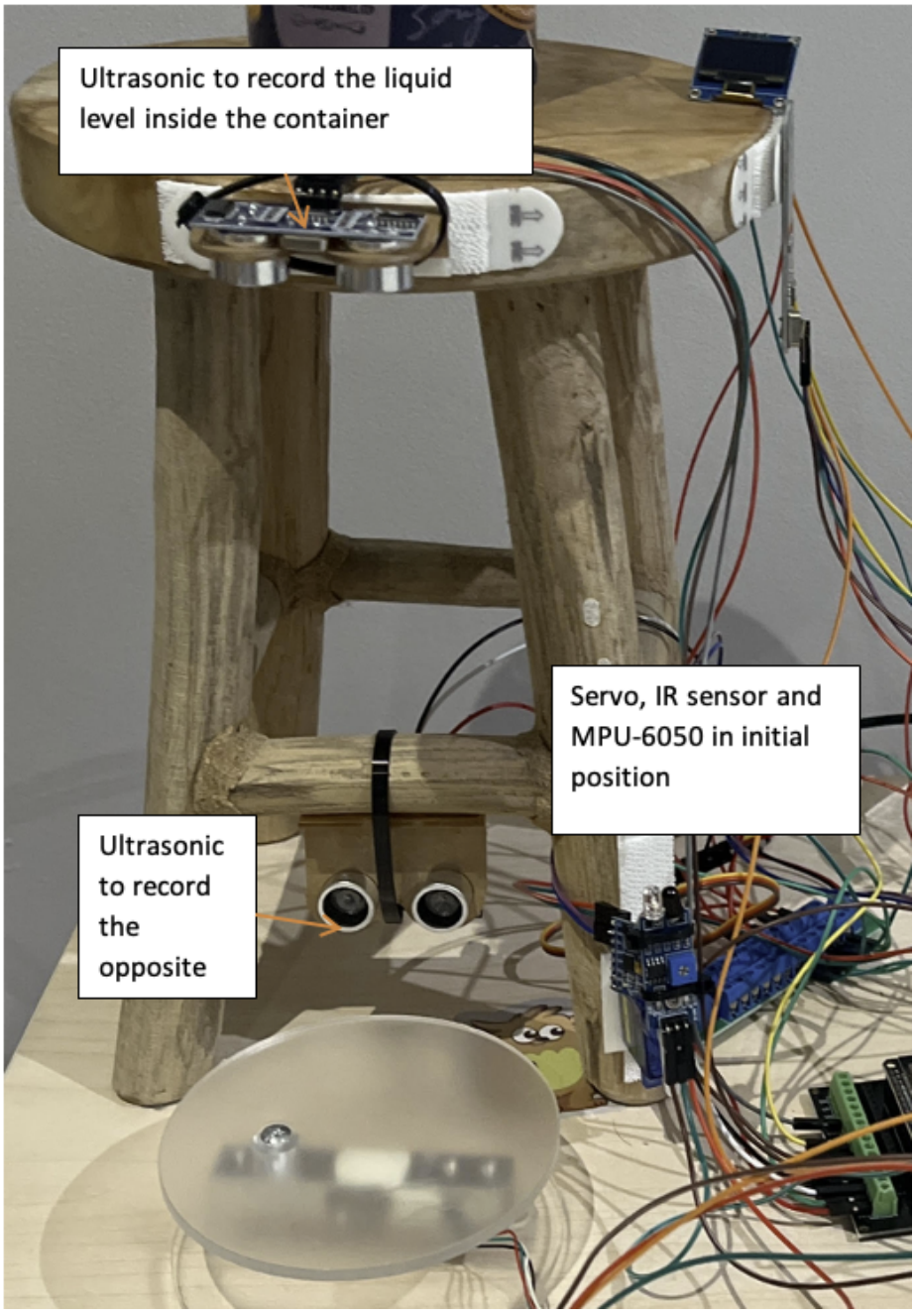
```
    display.println("If you would like to manually dispense please press
        ↪ the button within 10 seconds, otherwise do nothing...");
    display.display();

    unsigned long esp32Time = millis(); //Current time since esp32
        ↪ started running
    while(true) //Ask if customer wishes to manually or automatically
        ↪ dispense
    {
        if(digitalRead(BUTTON_PIN) == LOW) //If button pressed
        {
            display.clearDisplay();
            display.setCursor(0,0);
            display.println("Button press detected, dispensing manually...
                ↪ ");
            display.display();
            delay(1500);
            dispenseAutomatically = false;
            break;
        }

        //Compare time at start of loop to
        //current time against the decision time
        if(millis() - esp32Time > DECISION_TIME)
        {
            //Continue with automatic dispensing
            display.clearDisplay();
            display.setCursor(0,0);
            display.println("Continuing with automatic dispensing...");
            display.display();
            delay(1500);
            break;
        }

        delay(100);
    }
}
```

C.3 Automatic Dispensing Function

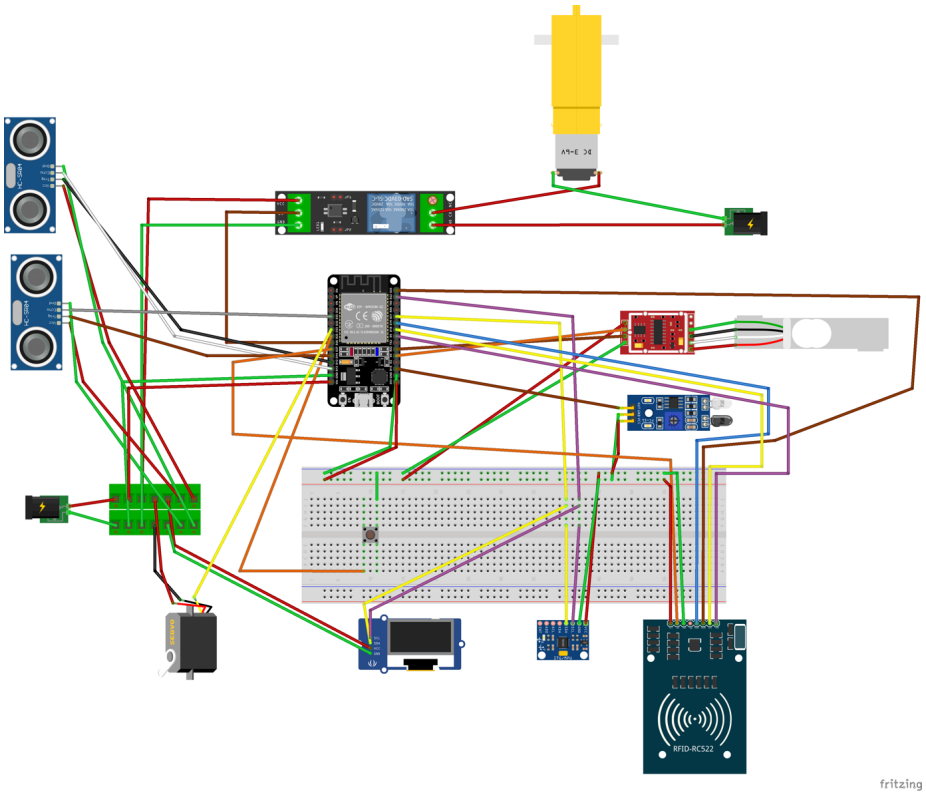
```
void AutomaticDispense(float &amountDispensed, float &costDispensed,
    ↪ float height)
{
    digitalWrite(RELAY_PIN, HIGH); //Turn on pump and read liquid level
    float initialRead = ultrasonic2.read();

    while(ultrasonic2.read()<=(initialRead - height))
    {
```

```
        if(scale.is_ready())
        {
            //Read scale
            amountDispensed = scale.get_units(10);
        }
        costDispensed = amountDispensed * COST_PER_GRAM; //Cost

        display.clearDisplay();
        display.setCursor(0, 0);
        display.println("Amount Dispensed:");
        display.println(String(amountDispensed)+"g");
        display.println("Total Cost:");
        display.println(String(costDispensed/100));
        display.print("Liquid Height: ");
        display.print(ultrasonic2.read());
        display.println("cm");
        display.println("To stop dispensing early press button...");
        display.display();

        if(digitalRead(BUTTON_PIN) == LOW) //When button pressed stop
            ↪ dispensing
        {
            delay(50);
            break;
        }
        delay(100);
    }
    digitalWrite(RELAY_PIN, LOW); //Turn off pump
}
```

C.4 Automatic Dispenser Layout

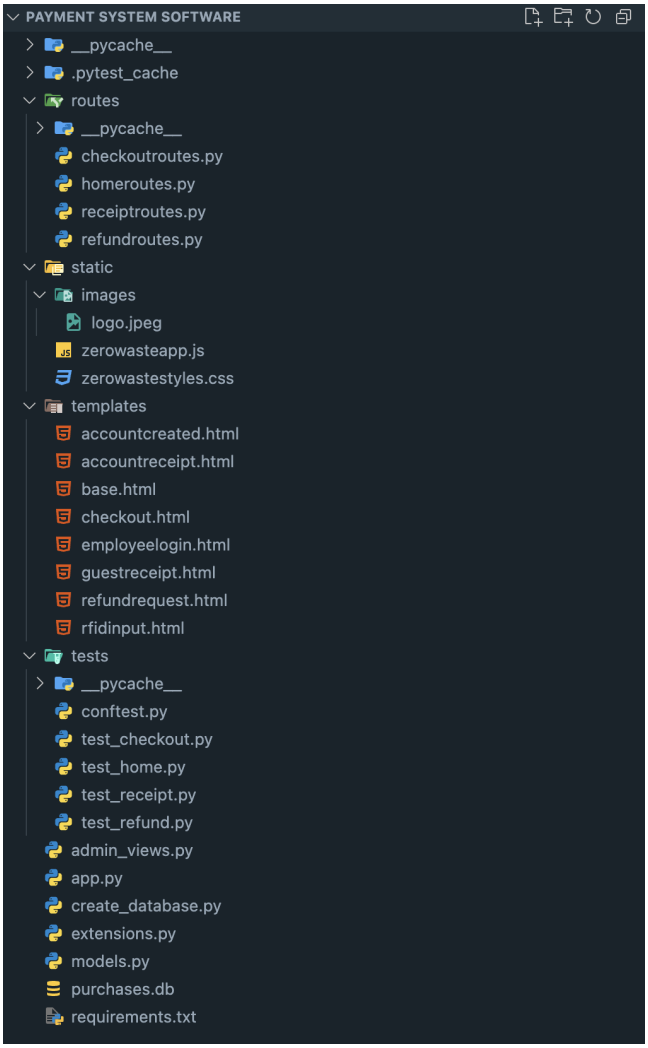


C.5 Automatic Dispenser Circuit Diagram

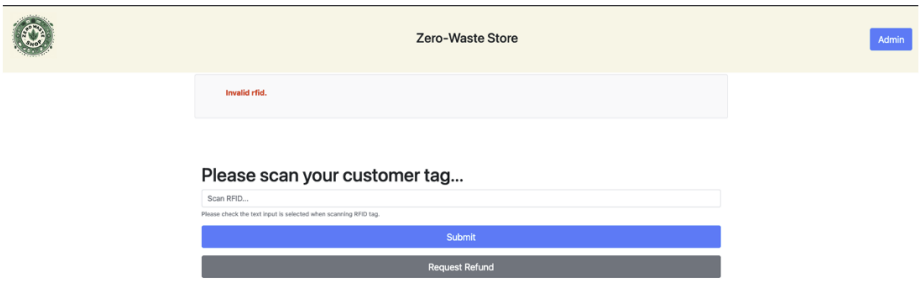


D Self-Checkout

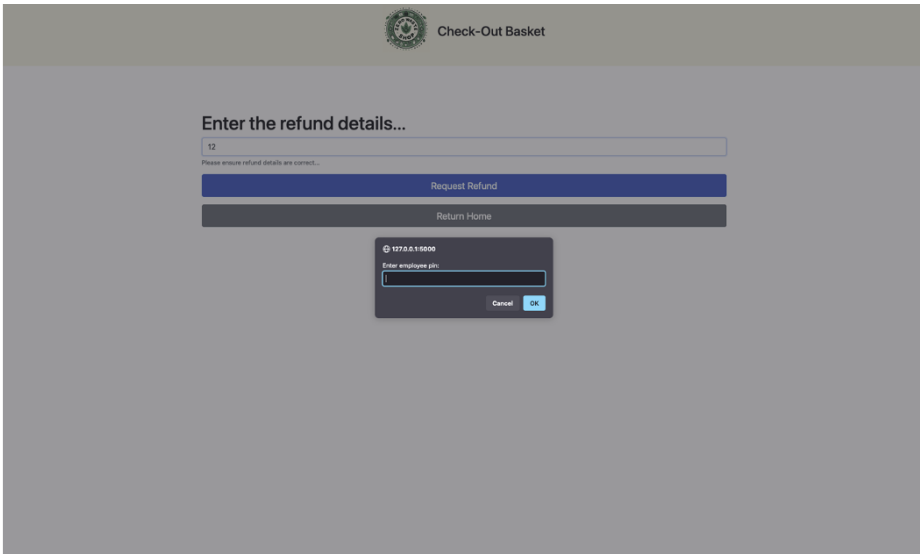
D.1 App Directory



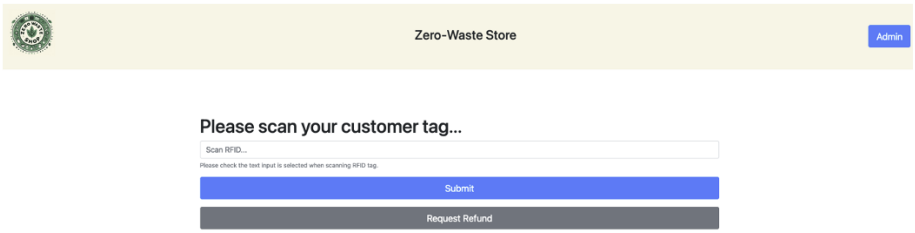
D.2 Home Page with Example Flash Message



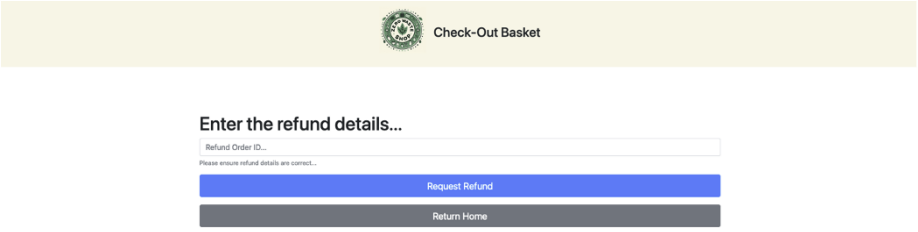
D.3 Example Employee Pin Prompt



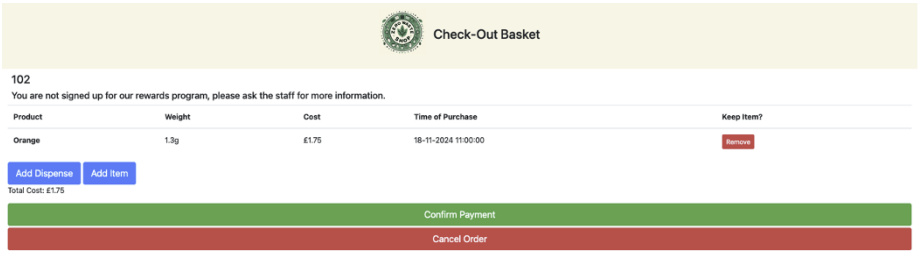
D.4 /home/scanrfid



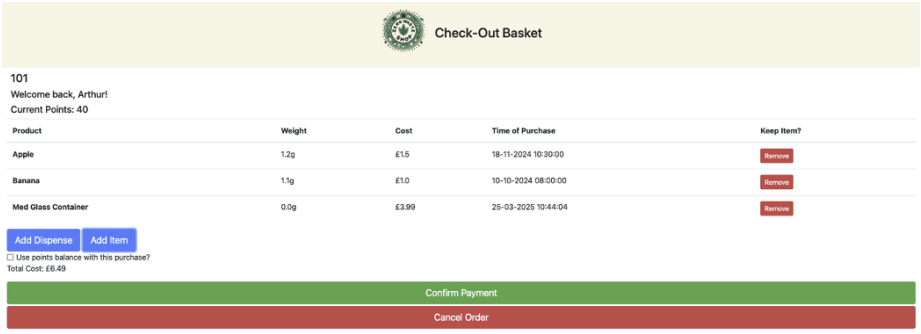
D.5 /refund/refund_request



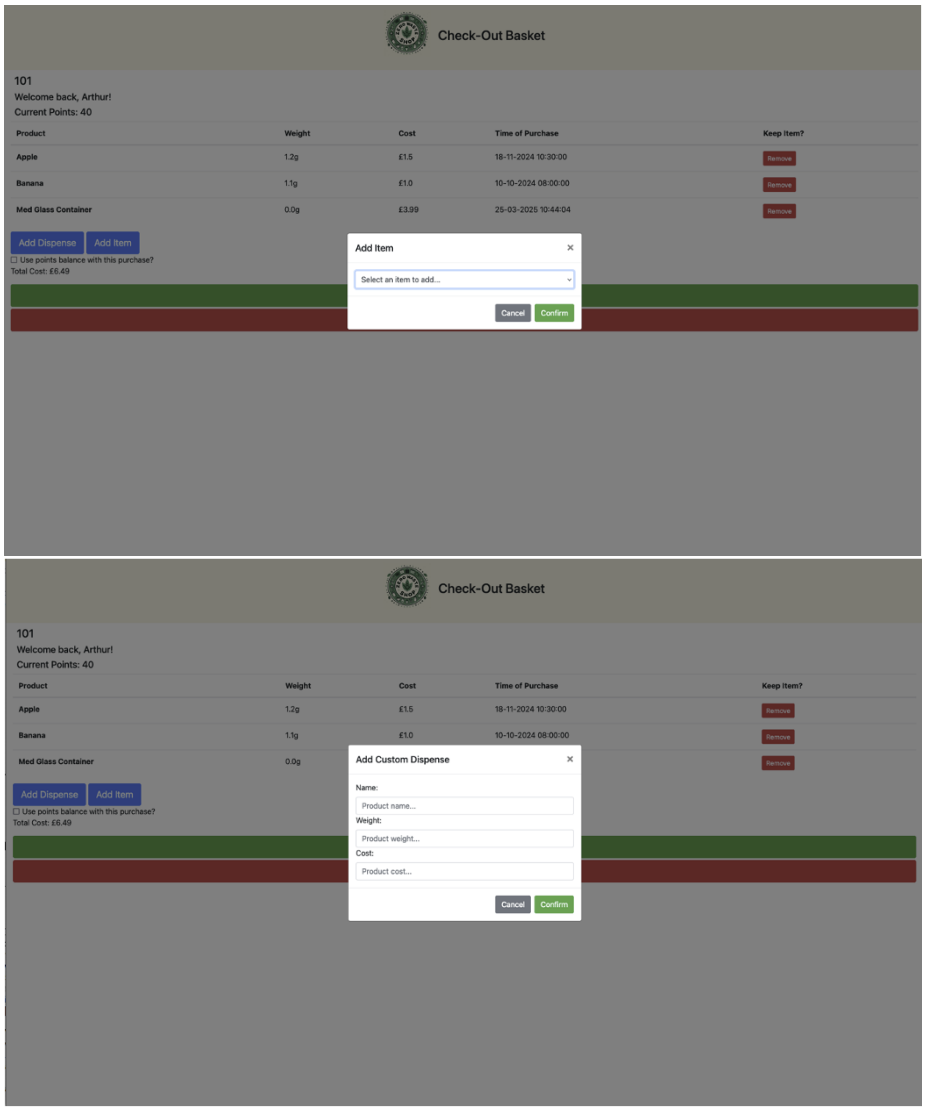
D.6 /checkout/checkout Guest



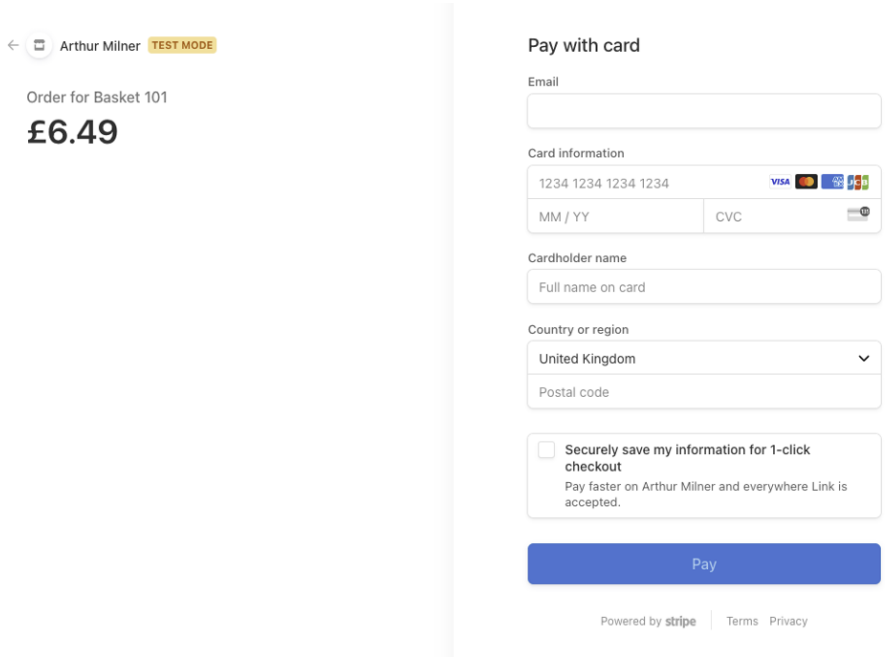
D.7 /checkout/checkout Account



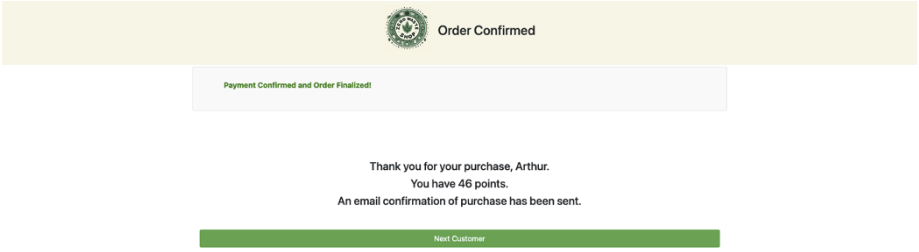
D.8 /checkout/checkout Add Item/Dispense



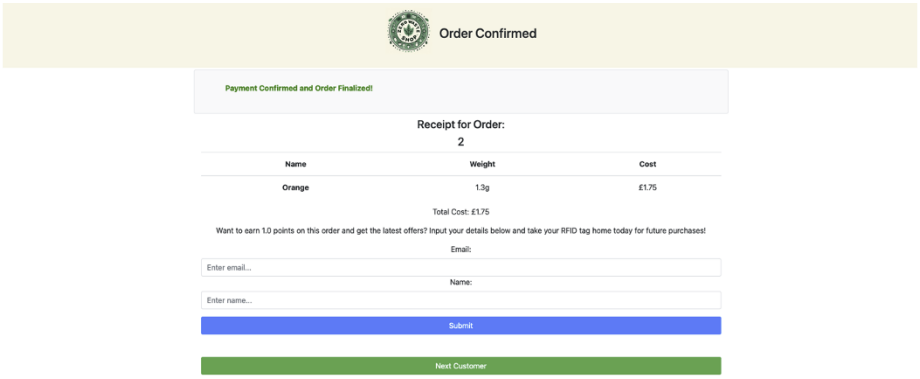
D.9 Stripe Payment Session



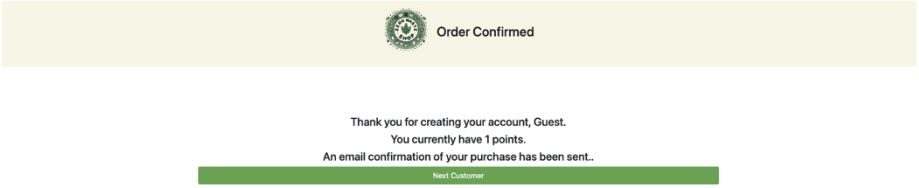
D.10 /receipt/account_receipt/..



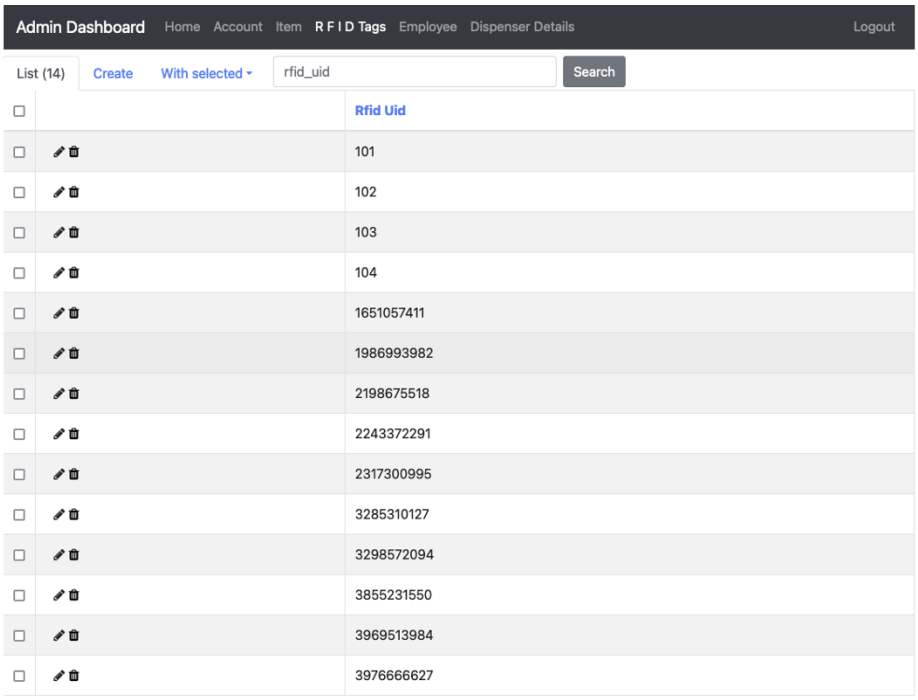
D.11 /receipt/guest_receipt/..



D.12 /receipt/new_account/..



D.13 Flask-Admin Example View



E MQTT

E.1 List of MQTT Topics

Topic	Topic Purpose
`shop/customer`	Topic used to send dispense details from the dispensers to the SQLite database. `(rfid_uid, product_name, cost, weight)`
`shop/business/capacity`	Topic used to send product capacity details from the dispensers to InfluxDB for use in the capacity dashboard and capacity notifications. `(product_name, capacity)`

Topic	Topic Purpose (Payload Parameters)
shop/business/dispense	Topic used to send product dispense details from the dispensers to InfluxDB for use in the business dashboard. (product_name, cost, weight)
dispenser/details	Topic to request dispenser details, request is sent from a dispenser with its ID in the payload. (dispenser_id)
dispenser/response/1	Topic which is subscribed to on the dispensers, the topic ID depending on the dispenser, in this case it would be a dispenser ID of 1. Topic is published to in response to sent requests under the dispenser/details topic. (product_name and cost_per_gram)
dispenser/verify	Topic to request validation of a given RFID UID, request is sent from a dispenser when an RFID tag is detected. (dispenser_id, rfid_uid)
dispenser/verify/1	Topic which is subscribed to on the dispensers, the topic ID depending on the dispenser, in this case it would be a dispenser ID of 1. Topic is published to in response to sent requests for validation under the dispenser/verify topic, returning true for a valid tag and false for an invalid tag. (rfid_valid)

E.2 MQTT Callback Function

```
1 //Callback function for receieving MQTT messages
2 //Adapted from https://randomnerdtutorials.com/esp32-mqtt-publish-
      ↪ subscribe-arduino-ide/
3 void callback(char* topic, byte* message, unsigned int length)
4 {
5     //If topic matches dispenser ID
6     if(String(topic) == String(topic_dispenser_response))
7     {
```

```
        //Get the message
        String JSONMessage;
        for (unsigned int i = 0; i < length; i++)
        {
            JSONMessage += (char)message[i];
        }
        //Using ArduinoJson library to parse the JSON string message
        //200 represents the bytes set aside for parsing
        StaticJsonDocument<200> doc;
        //Checks if the message is correctly deserialised
        DeserializationError error = deserializeJson(doc, JSONMessage);

        if (!error) //If no error
        {
            //Set product_name and cost_per_gram from parsed JSON to the
                ↪ ESP32 variables
            if (doc.containsKey("product_name") && doc.containsKey("
                ↪ cost_per_gram"))
            {
            PRODUCT_NAME = doc["product_name"].as<String>();
            COST_PER_GRAM = doc["cost_per_gram"].as<float>();
            Serial.println("Received updated product details:");
            Serial.print("Name: ");
            Serial.println(PRODUCT_NAME);
            Serial.print("Cost per gram: ");
            Serial.println(COST_PER_GRAM);
            }
        }
        else
        {
            while(true)
            {
            display.clearDisplay();
            display.setCursor(0,0);
            display.println("Error getting product details...");
            display.setCursor(0,1);
            display.println("Please reboot dispenser...");
            delay(1000);
            }
        }
    }

    //If topic matches dispenser ID
    if(String(topic) == String(topic_verify_response))
    {
        //Get the message
        String JSONMessage;
        for (unsigned int i = 0; i < length; i++)
        {
            JSONMessage += (char)message[i];
```

```
56          }
57
58          //Using ArduinoJson library to parse the JSON string message
59          //200 represents the bytes set aside for parsing
60          StaticJsonDocument<200> doc;
61          //Checks if the message is correctly deserialised
62          DeserializationError error = deserializeJson(doc, JSONMessage);
63
64          if (!error) //If no error
65          {
66              //If reponse is valid rfid set valid rfid to true
67              if (doc.containsKey("rfid_valid"))
68              {
69              //Extract the value
70                  bool isValidRFID = doc["rfid_valid"].as<bool>();
71
72                  if(isValidRFID)
73                  {
74                  rfidValid = true;
75                  display.clearDisplay();
76                  display.setCursor(0,0);
77                  display.println("Valid tag...");
78                  display.display();
79                  delay(1000);
80                  }
81                  else
82                  {
83                  rfidValid = false;
84                  display.clearDisplay();
85                  display.setCursor(0,0);
86                  display.println("Invalid tag...");
87                  display.display();
88                  delay(1000);
89                  }
90              }
91          }
92          else //If mqtt error
93          {
94              display.clearDisplay();
95              display.setCursor(0,0);
96              display.println("Error validating tag...");
97              delay(1000);
98          }
99      }
100 }
```

E.3 Getting Dispenser Details

```
//Subscribing to response topic
client.subscribe(topic_dispenser_response);
//Sending request for dispenser product details on ID 1
String detailsPayload = "{ \"dispenser_id\": 1 }";
client.publish(topic_dispenser_details, detailsPayload.c_str());
//Wait for MQTT response, times out after 10 seconds and presents error
unsigned long startTime = millis();
//Whilst PRODUCT_NAME still not set
while (PRODUCT_NAME == "" && (millis() - startTime < 10000))
{
    if (!client.connected())
    {
        reconnect_MQTT();
    }
    client.loop();
    delay(100);
}

//Halt if failed to grab details
while(true)
{
    if(PRODUCT_NAME == "" || COST_PER_GRAM == 0.0)
    {
        display.clearDisplay();
        display.setCursor(0,0);
        display.println("Couldn't get product details, please reboot...")
            ↪ ;
        display.display();
        delay(1000);
    }
    else
    {
        display.clearDisplay();
        display.setCursor(0,0);
        display.print("Product: ");
        display.println(PRODUCT_NAME);
        display.print("Cost Per Gram: ");
        display.println(COST_PER_GRAM);
        display.display();
        delay(3000);
        break;
    }
}
```

E.4 Validating RFID

```
unsigned long decimalUID = 0; //To store the RFID ID
bool detectedContainer = false; //Track if container is on scale
```

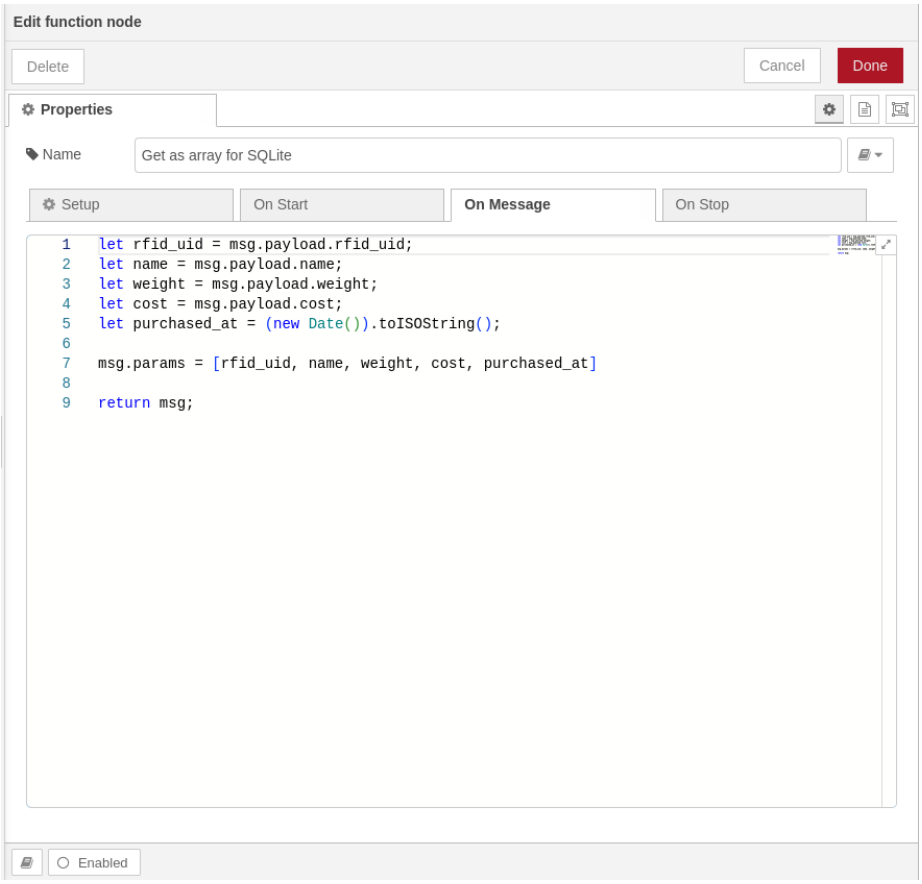
```
3
4  display.clearDisplay();
5  display.setCursor(0, 0);
6  display.println("Place container and scan a tag...");
7  display.display();
8
9  //Loops until an rfid tag is scanned or container is removed
10 while (true) {
11     if(rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial())
12     {
13
14     decimalUID = 0;
15     for (byte i = 0; i < (rfid.uid.size); i++)
16     {
17         decimalUID = decimalUID * 256 + rfid.uid.uidByte[i];
18     }
19
20     rfid.PICC_HaltA();
21     rfid.PCD_StopCrypto1();
22
23     //After tag is read check container is on scale
24     while(!detectedContainer)
25     {
26         //Read ToF value
27         if (tof.dataReady())
28         {
29         int tofDist = tof.distance();
30         tof.clearInterrupt();
31         //If container is detected as on the scale
32         if(tofDist < CONTAINER_DETECTION)
33         {
34             display.println("Container detected...");
35             display.display();
36             delay(1000);
37             detectedContainer = true;
38         }
39         display.clearDisplay();
40         display.setCursor(0, 0);
41         display.println("Checking for container...");
42         display.println("Distance:");
43         display.println(tofDist);
44         display.display();
45         }
46
47         delay(100);
48     }
49
50     if (!client.connected())
51     {
52         reconnect_MQTT();
```

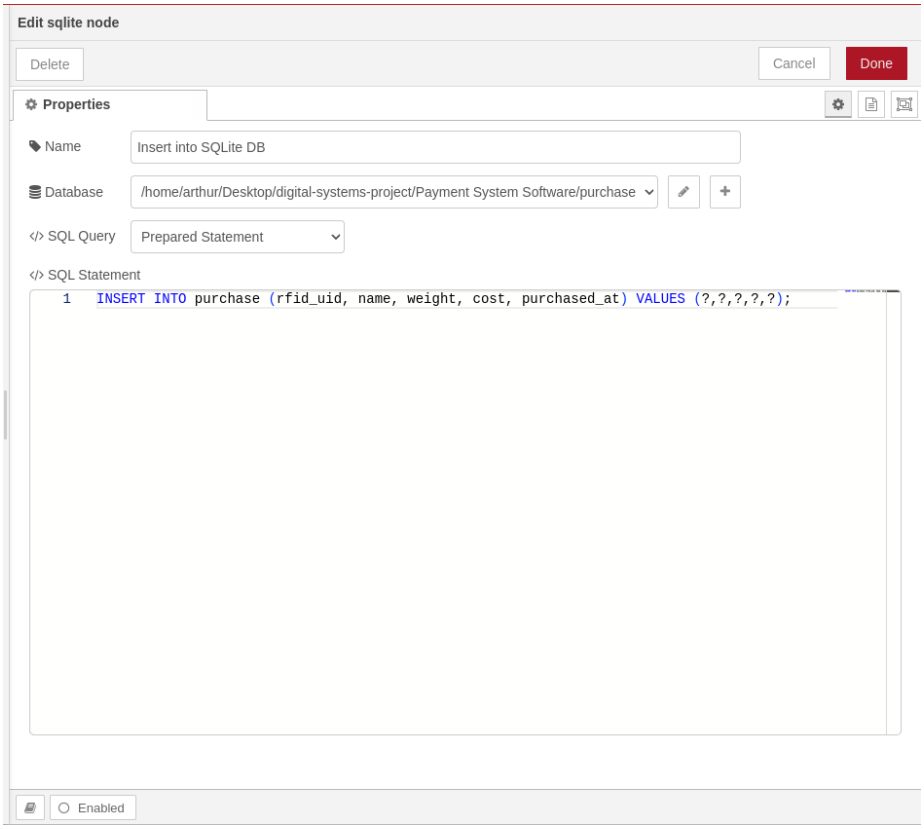
```
    }
    client.loop();

    //Set rfid valid to false before beginning validation
    rfidValid = false;

    //Sending rfid validation request with dispenser id
    String rfidValidPayload = "{ \"rfid_uid\": " + String(decimalUID) + "
        ↪ ,
    \"dispenser_id\": 1 }";
    if(client.publish(topic_verify_rfid, rfidValidPayload.c_str()))
    {
        Serial.println("RFID verification message sent successfully");
    }
    else
    {
        Serial.println("Failed to send RFID verification message");
    }

    //Wait for MQTT response, times out after 10 seconds and presents
        ↪ error
    display.clearDisplay();
    display.setCursor(0, 0);
    display.println("Verifying tag...");
    display.display();
    //While rfid is not valid or not yet timed out listen for response
    unsigned long startTime = millis();
    while (rfidValid==false && (millis() - startTime < 10000))
    {
        if (!client.connected())
        {
        reconnect_MQTT();
        }
        client.loop();
        delay(200);
    }

    //Break out while loop if rfid valid
    if(rfidValid)
    {
        display.clearDisplay();
        display.setCursor(0, 0);
        display.println("Tag validated!");
        display.display();
        delay(2500);
        break;
    }
    else //Display error then wait for another tag
    {
        display.clearDisplay();
```

```
101         display.setCursor(0, 0);
102         display.println("Error validating or an invalid tag was scanned
                ↪ ...");
103         display.display();
104         delay(2500);
105     }
106     }
107     delay(100);
108 }
109 
110 //Set rfid valid to false again
111 rfidValid = false;
```

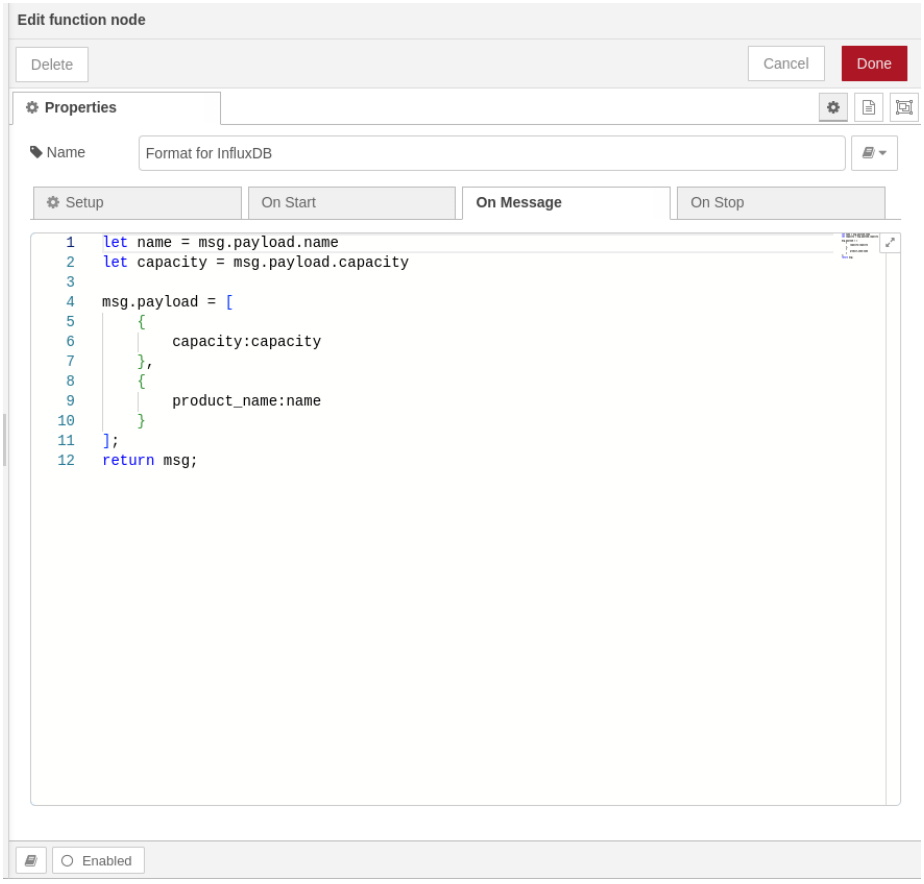
F Node-RED

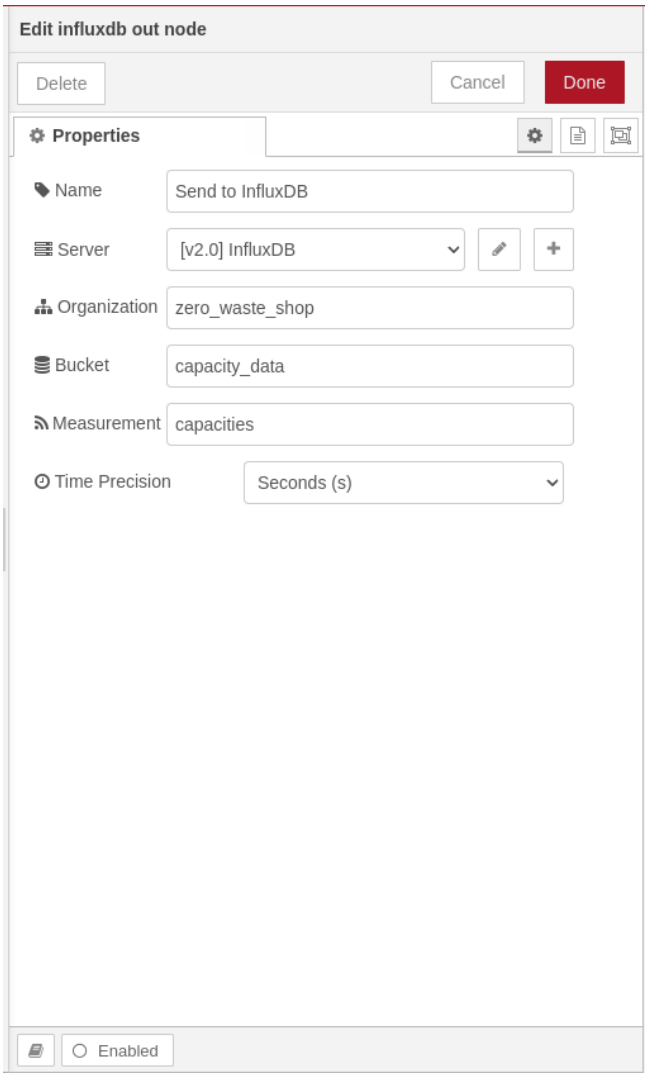
F.1 SQL Formatting



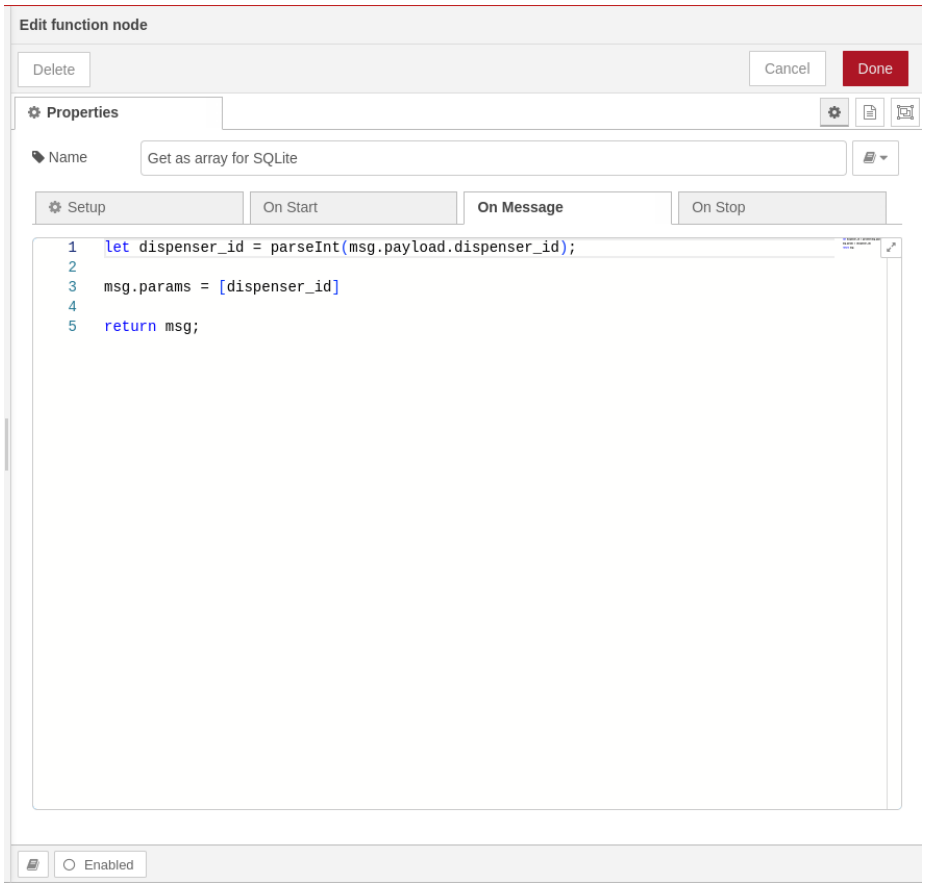

Edit sqlite node
Delete
Cancel
Done
Properties
Name
Insert into SQLite DB
Database
/home/arthur/Desktop/digital-systems-project/Payment System Software/purchase
SQL Query
Prepared Statement
SQL Statement
1 INSERT INTO purchase (rfid_uid, name, weight, cost, purchased_at) VALUES (?,?,?,?,?);
Enabled

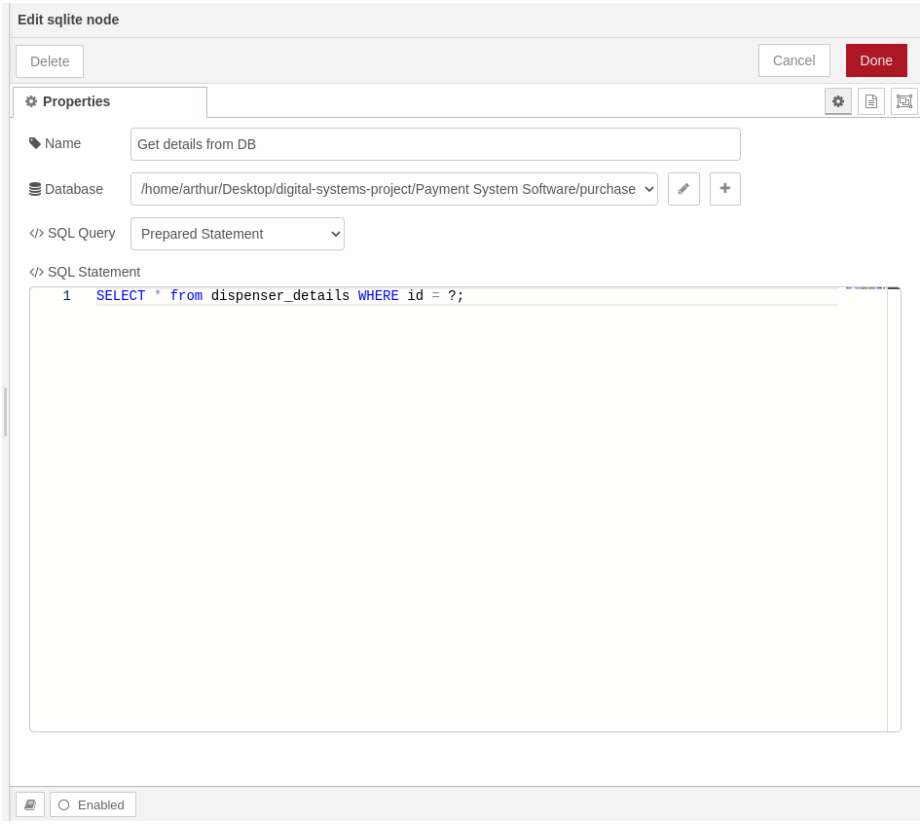
F.2 InfluxDB Formatting

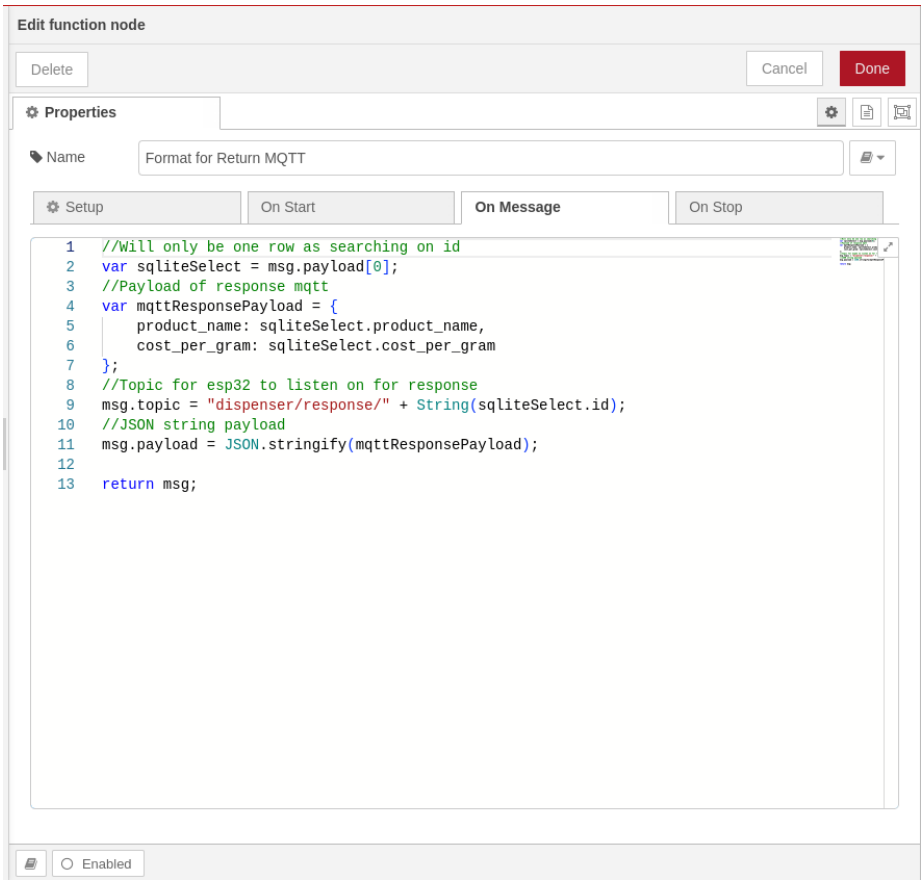



Edit influxdb out node
Delete
Cancel
Done
Properties
Name
Send to InfluxDB
Server
[v2.0] InfluxDB
Organization
zero_waste_shop
Bucket
capacity_data
Measurement
capacities
Time Precision
Seconds (s)
Enabled

F.3 Dynamic MQTT Response Topic




Edit sqlite node
Delete
Cancel
Done
Properties
Name
Get details from DB
Database
/home/arthur/Desktop/digital-systems-project/Payment System Software/purchase
SQL Query
Prepared Statement
SQL Statement
1 SELECT * from dispenser_details WHERE id = ?;
Enabled



G Testing

G.1 InfluxDB Dummy Data Code

```
#include <WiFi.h>
#include <PubSubClient.h> //MQTT

//WiFi details
const char* ssid = "******";
const char* password = "*******";

//MQTT broker details
const char* mqtt_server = "192.168.1.100"; //IP of Raspberry Pi/MQTT
    ↪ server
const int mqtt_port = 1883; //MQTT port
```

```
11 const char* mqtt_user = "****"; //MQTT username
12 const char* mqtt_password = "****"; //MQTT password
13
14 //MQTT topics
15 //Topic for customer basket details
16 const char* topic_customer = "shop/customer";
17 //Topic for product capacity details
18 const char* topic_business_capacity = "shop/business/capacity";
19 //Topic for product dispense details
20 const char* topic_business_dispense = "shop/business/dispense";
21
22 WiFiClient espClient; //Wifi setup
23 PubSubClient client(espClient); //MQTT connection/setup
24
25 //Delay between publishing dummy data
26 const float DISPENSE_DELAY = 100000;
27 //Array of product names to randomly choose from
28 const char* PRODUCT_NAMES[] = {"Peanuts", "Cereal", "Olive Oil", "
       ↪ Chickpeas", "Green Peas", "Lentils", "Penne Pasta", "Washing
       ↪ Powder", "Sesame Oil", "Cornflakes"};
29 //Number of items in array
30 //sizeof(PRODUCT_NAMES) returns the number of bytes
31 //(one word should be 4 bytes)
32 //then sizeof(PRODUCT_NAMES[0]) will get the number of bytes
33 //in one word which will be the same for all words
34 //so sizeof(PRODUCT_NAMES) / sizeof(PRODUCT_NAMES[0]) will
35 //give the number of words in the array
36 const int PRODUCT_COUNT = sizeof(PRODUCT_NAMES) / sizeof(PRODUCT_NAMES
       ↪ [0]);
37 //Delay between sending dummy data
38 const unsigned long PUBLISH_DELAY = 100000;
39
40 //Connects ESP32 to WiFi
41 void setup_wifi() {
42     delay(10);
43     Serial.println();
44     Serial.print("Connecting to ");
45     Serial.println(ssid);
46
47     WiFi.begin(ssid, password);
48     while (WiFi.status() != WL_CONNECTED)
49     {
50     delay(500);
51     Serial.println(".");
52     }
53
54     Serial.println();
55     Serial.println("WiFi connected");
56     Serial.println("IP address: ");
57     Serial.println(WiFi.localIP());
```

```
}

//Function to reconnect to MQTT broker
void reconnect_MQTT() {
    //Loops until reconnected to MQTT broker
    while (!client.connected()) {
    Serial.println("Attempting MQTT connection...");
    if (client.connect("ESP32Client", mqtt_user, mqtt_password)) {
        Serial.println("connected");
    } else {
        Serial.println("failed, rc=");
        Serial.println(client.state());
        Serial.println(" try again in 5 seconds");
        delay(5000);
    }
    }
}

void setup()
{
    Serial.begin(115200);
    //Wi-fi and MQTT
    setup_wifi();
    client.setServer(mqtt_server, mqtt_port);
}

void loop()
{
    //Gets random index to select product name from
    int productNameIndex = esp_random() % PRODUCT_COUNT;
    String productName = PRODUCT_NAMES[productNameIndex];
    //Random value between 0-100 for capacity % full
    float percentFull = esp_random() % 101;
    //Random value for amount dispensed (between 0.01 and 200.00)
    float amountDispensed = ((esp_random() % 20000) + 1) / 100;
    //Random value for cost dispensed (between 0.01 and 20.00)
    float costDispensed = ((esp_random() % 2000) + 1) / 100;

    //Details which will send to MQTT
    //JSON string payload for MQTT of capacity details
    //Random product name and capacity between 0-100 sent
    String businessCapacityPayload = "{";
    businessCapacityPayload += "\"name\":\"" + productName + "\",";
    businessCapacityPayload += "\"capacity\":" + String(percentFull, 2);
    businessCapacityPayload += "}";

    //JSON string payload for MQTT of dispense details
    //Random product name, amount, and cost
    String businessDispensePayload = "{";
    //Sending as string
```

```
    businessDispensePayload += "\"name\": \"" + productName + "\"";
    //Sending as number
    businessDispensePayload += ",\"weight\": " + String(amountDispensed,
        ↪ 2);
    //Sending as number
    businessDispensePayload += ",\"cost\": " + String(costDispensed, 2);
    businessDispensePayload += "}";

    unsigned long startTime = millis();
    while(millis() - startTime < PUBLISH_DELAY)
    {
    //Trys to reconnect if connection drops
    if (!client.connected())
    {
        reconnect_MQTT();
    }
    client.loop();
    }

    //Publishing to MQTT under business dispense topic
    if(client.publish(topic_business_dispense, businessDispensePayload.
        ↪ c_str()))
    {
    Serial.println("Dispense successfully sent to dashboard.");
    }
    else //If error sending data message is displayed
    {
    Serial.println("Failed to send dispense.");
    }

    //Publishing to MQTT under business capacity topic
    if(client.publish(topic_business_capacity, businessCapacityPayload.
        ↪ c_str()))
    {
    Serial.println("Capacity successfully sent to dashboard.");
    }
    else //If error sending data message is displayed
    {
    Serial.println("Failed to send capacity.");
    }
}
```

G.2 Unit Test Configuration

Test database used within unit tests:

```
@pytest.fixture
def client():
```

```
    #Use in-memory database for testing
    app = createApp({
        'SQLALCHEMY_DATABASE_URI': 'sqlite:///:memory:'
    })
    app.config['TESTING'] = True
    with app.test_client() as client:
        with app.app_context():
            db.create_all() #Create the database files for test
            #Adding dummy data and all valid rfid uid
            dummyData = [
            RFIDTags(rfid_uid=101),
            RFIDTags(rfid_uid=102),
            RFIDTags(rfid_uid=103),
            RFIDTags(rfid_uid=104),
            RFIDTags(rfid_uid=3855231550),
            RFIDTags(rfid_uid=3285310127),
            RFIDTags(rfid_uid=2317300995),
            RFIDTags(rfid_uid=3298572094),
            RFIDTags(rfid_uid=1986993982),
            RFIDTags(rfid_uid=2243372291),
            RFIDTags(rfid_uid=3976666627),
            RFIDTags(rfid_uid=2198675518),
            RFIDTags(rfid_uid=1651057411),
            RFIDTags(rfid_uid=3969513984),
            Purchase(rfid_uid=101, name="Apple", weight=1.2, cost=1.50,
                ↪ purchased_at=datetime(2024, 11, 18, 10, 30)),
            Purchase(rfid_uid=101, name="Banana", weight=1.1, cost=1.00,
                ↪ purchased_at=datetime(2024, 10, 10, 8, 00)),
            Purchase(rfid_uid=102, name="Orange", weight=1.3, cost=1.75,
                ↪ purchased_at=datetime(2024, 11, 18, 11, 00)),
            Purchase(rfid_uid=103, name="Milk", weight=2.0, cost=3.50,
                ↪ purchased_at=datetime(2024, 11, 18, 11, 15)),
            Purchase(rfid_uid=103, name="Bread", weight=0.5, cost=2.00,
                ↪ purchased_at=datetime(2024, 11, 18, 11, 20)),
            Purchase(rfid_uid=104, name="Eggs", weight=1.0, cost=2.50,
                ↪ purchased_at=datetime(2024, 11, 18, 12, 00)),
            DispenserDetails(id = 1, product_name = 'Washing Liquid',
                ↪ cost_per_gram = 1.1),
            DispenserDetails(id = 2, product_name = 'Penne Pasta',
                ↪ cost_per_gram = 0.5),
            DispenserDetails(id = 3, product_name = 'Vegetable Oil',
                ↪ cost_per_gram = 4),
            DispenserDetails(id = 4, product_name = 'Cornflakes',
                ↪ cost_per_gram = 0.6),
            Item(name='Small Glass Container', cost=2.50),
            Item(name='Med Glass Container', cost=3.99),
            Item(name='Large Glass Container', cost=5.00),
            Account(email='arthurmilner01@gmail.com', name="Arthur",
                ↪ rfid_uid=101, points=40, created_at=datetime(2024, 10,
                ↪  17, 9, 00)),
```

```
            Employee(name='John', pin=2522945794),
            #! FOR TESTING RECEIPT ROUTE ONLY
            Account(email='arthur2.milner@live.uwe.ac.uk', name="Arthur",
                ↪ rfid_uid=103, points=40, created_at=datetime(2024, 10,
                ↪  17, 9, 00)),
            Order(id=1, order_cost=15.00, points_used=0,
                ↪ checkout_session_id="N/A", order_refunded=False),
            ArchivedPurchase(rfid_uid=103, name="Shampoo", weight=2.0,
                ↪ cost=10.00, order_id=1),
            ArchivedPurchase(rfid_uid=103, name="Hand Soap", weight=1.0,
                ↪ cost=5.00, order_id=1),
            #! FOR TESTING GUEST RECEIPT AND REFUND ROUTES ONLY
            Order(id=2, order_cost=15.00, points_used=0,
                ↪ checkout_session_id="N/A", order_refunded=True),
            ArchivedPurchase(rfid_uid=104, name="Shampoo", weight=2.0,
                ↪ cost=10.00, order_id=2),
            ArchivedPurchase(rfid_uid=104, name="Hand Soap", weight=1.0,
                ↪ cost=5.00, order_id=2)
            ]
            db.session.bulk_save_objects(dummyData)
            db.session.commit()
            yield client
            db.drop_all() #Drop database after each test
```

Configuring app.py to use the test database when running unit tests:

```
def createApp(testConfig=None):
   app = Flask(__name__)
   app.config['SECRET_KEY'] = 'secretkey123'

   #For unit testing
   if testConfig is None:
      #Creating the flask alchemy database
      sqlitePath = os.path.join(os.path.dirname(os.path.abspath(__file__)
          ↪ ), 'purchases.db')
      app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///'+sqlitePath
   else:
      app.config.update(testConfig)
```

G.3 Dispenser Test Cases

Test Case ID	Related Dispensers	Test Description	Test Step	Expected Result	Result

Test Case ID	Related Dispensers	Test Description	Test Step	Expected Result	Result
D-1	Solid & Liquid	Unsuccessful initialization of the scale should halt the program and display an error.	Disconnect the wires of the scales and boot the dispensers.	The dispensers halt at the initialization for the scale.	Pass
D-2	Solid & Liquid	On boot the dispenser attempts to connect to the Wi-Fi and a message is displayed to signify loading and on successful connection.	Boot dispenser and observe the messages displayed on the OLED screen.	Wi-Fi connection loading message displays including a message when Wi-Fi is connected.	Pass
D-3	Solid & Liquid	On boot the dispenser configures the MQTT connection and a message is displayed to signify loading and on successful connection.	Boot the dispenser with the MQTT server offline and online, observing the feedback on the OLED screen. On successful connection add confirmation the dispenser is connected to the MQTT server by publishing to a topic subscribed to by a terminal window.	When the MQTT is offline the dispensers halt and display the connecting message until the MQTT is online. The published message will appear in the subscribed terminal window.	Pass

Test Case ID	Related Dispensers	Test Description	Test Step	Expected Result	Result
D-4	Solid & Liquid	A message for calibrating the scale is displayed on boot and waits for the employee's button press, the calibration factor should be similar across multiple calibrations.	Observe the calibration message and button press is registered correctly, repeatedly calibrate the scale and make note of the calibration factors.	The calibration message and button press is registered correctly, the calibration factor is similar across multiple scale calibrations.	Pass
D-5	Solid & Liquid	On boot messages are displayed at each peripheral's initialization.	Boot the dispensers and observe the messages are displayed as expected.	Messages are displayed as expected.	Pass
D-6	Solid & Liquid	On boot after initializing the Wi-Fi and MQTT connection the dispensers get their product name and cost per gram from the SQLite database dependant on their dispenser ID. The product name and cost per gram is displayed on successful fetching of the data, otherwise an error is displayed and the dispensers halt.	Add debug nodes to the Node-RED flow expected to trigger in fetching dispenser details, verifying the JSON payload and formatting are correct. Boot dispensers and observe the product name and cost per gram is as expected based on the dispenser ID, changing the dispenser ID manually in the code. Attempt valid and invalid dispenser IDs.	Valid dispenser ID returns the correct details based on what is stored within the SQLite database, invalid IDs will display an error and halt the dispenser.	Pass

Test Case ID	Related Dispensers	Test Description	Test Step	Expected Result	Result
D-7	Solid & Liquid	The capacity data recorded by the dispensers is sent to MQTT/Node-RED in the expected format, and inserted into InfluxDB successfully.	Add debug nodes to the flow expected to trigger from capacity messages and verify the JSON payload and formatting, triggering the flow with known data. Once the flow completes verify the data is sent correctly in InfluxDB.	The JSON payload is as expected and is formatted correctly throughout the flow. The capacity data is inserted into the database correctly.	Pass
D-8	Solid & Liquid	The dispense data recorded by the dispensers is sent to MQTT/Node-RED in the expected format, and added to the SQLite database.	Add debug nodes to the flow expected to trigger from dispense messages and verify the JSON payload and formatting throughout the flow. Send dispenses with known details then verify SQLite database changes are as expected after the message is published.	The JSON payload is as expected and is formatted correctly throughout the flow. The dispense is inserted into the database correctly.	Pass

Test Case ID	Related Dispensers	Test Description	Test Step	Expected Result	Result
D-9	Solid & Liquid	The business data recorded by the dispensers is sent to MQTT/Node-RED in the expected format, and inserted into InfluxDB successfully.	Add debug nodes to the flow expected to trigger from business messages and verify the JSON payload and formatting, triggering the flow with known data. Once the flow completes verify the data is sent correctly in InfluxDB.	The JSON payload is as expected and is formatted correctly throughout the flow. The business data is inserted into the database correctly.	Pass
D-10	Solid & Liquid	Scanned tag UID is recorded by the dispensers and sent to MQTT/Node-RED in the expected format, and validated based on the contents of the RFID Tags table in the database.	Add debug nodes to the flow expected to trigger from scanning an RFID tag, verifying the payload and formatting at each node. Once the flow returns verify it returns true for a valid tag and false for an invalid tag on the topic the dispenser is subscribed to. Ensure return true continues to dispensing and false requests another tag to be scanned.	Scanning a valid tag will send the tag to MQTT, then to Node-RED, which then returns whether the tag is valid under a topic including the dispenser's ID. If the return is false the dispenser requests another tag to be scanned, if true the dispenser continues to dispensing.	Pass

Test Case ID	Related Dispensers	Test Description	Test Step	Expected Result	Result
D-11	Solid & Liquid	The weight read by the scale is within ±10% accuracy of the actual weight.	Calibrate the scale as normal and attempt dispense using known weights of 5G, 10G, 20G, 50G, 100G and 500G.	The scale reads the expected weight within ±10% accuracy.	Pass
D-12	Solid & Liquid	The RFID UID read by the dispensers RFID scanners is in the same format as the UID read by the checkout RFID scanner.	Scan an RFID on the dispensers and make a note of the read UID, scan that same RFID on the checkout and make a note of the read UID.	The UID read will be the same on both the dispensers and the checkout.	Pass
D-13	Solid	All messages are displayed reasonably within the 16x2 character array of the LCD screen and enough time is given to read displayed feedback.	Check all messages display for an appropriate amount of time and are legible.	The messages clearly display for an appropriate amount of time.	Pass
D-14	Liquid	All messages are displayed reasonably and enough time is given to read displayed feedback.	Check all messages display for an appropriate amount of time and are legible.	The messages clearly display for an appropriate amount of time.	Pass

Test Case ID	Related Dispensers	Test Description	Test Step	Expected Result	Result
D-15	Solid & Liquid	The calculated product capacity is accurate within ±10%.	Test different capacity levels on both dispensers, measuring the capacity and comparing the result to the ultrasonic's distance reading and calculated capacity %. Test 25%, 50%, 75% and 100% capacity.	The capacity is calculated within ±10% accuracy.	Pass
D-16	Liquid	Liquid stops dispensing when the container is removed from the scale to prevent spillages.	Test dispensing liquid without removing the container to ensure it doesn't trigger unexpectedly, and ensure when a container is removed dispensing ceases and the correct weight/cost of the dispense is recorded.	Dispensing continues as normal until it detects a container is removed, in which case the cost and weight before removal are recorded.	Pass

Test Case ID	Related Dispensers	Test Description	Test Step	Expected Result	Result
D-17	Liquid	The time of flight sensor on the liquid dispenser handles glass containers as expected.	Attempt dispenses using various sizes and types of glass containers, observing the recorded container distance on the time of flight sensor, comparing this to the actual distance.	The time of flight sensor detects an accuracy which is enough to appropriately trigger the stopping of dispenses on container removal.	Pass
D-18	Solid	When capacity is low the red LED should activate.	Record >20% and <20% capacities and observe the LED state.	The LED is on when capacity is <20% and off when capacity is >20%.	Pass
D-19	Solid & Liquid	Pressing the button during dispensing stops the dispense and records the current weight and cost.	Attempt a dispense and press the button, observing the recorded weight/cost and whether the dispense stops.	The dispense stops when the button is pressed and records the current dispense weight/cost.	Pass
D-20	Liquid	After scanning an RFID tag a check is made to ensure a container is on the scale.	Attempt to scan a tag with a container off the scale and attempt to scan a tag with a container on the scale.	Scanning a tag without a container on the scale will read the tag, then prompt the user to place a container on the scale. Scanning a tag with a container on the scale will proceed with tag validation as normal.	Pass

Test Case ID	Related Dispensers	Test Description	Test Step	Expected Result	Result
D-21	Solid & Liquid	When expecting a subscribe response from the MQTT server if a response is not received within 10 seconds an error is displayed.	Observe the fetching of dispenser details and the verification of an RFID tag when the MQTT server doesn't send a valid response.	Failure to fetch the dispenser details results in requiring a reboot, failure to validate the RFID tag results in re-prompting the user to scan a tag.	Pass
D-22	Solid & Liquid	If a dispense records inaccurate weight and cost it doesn't send to the checkout and the user is asked to seek help at checkout.	Attempt a dispense with a weight <0.01kg and >5kg, then attempt a dispense with a cost of <£0.01 and >£200.	Any dispenses reading outside the boundaries are not sent and an error is displayed to the user.	Pass
D-23	Solid & Liquid	On the completion of a dispense an appropriate message and time period is given to allow the customer to remove their container.	Attempt dispenses and observe the time allotted for the removal of the container, alongside the reading of the dispense complete message.	An appropriate amount of time is given to remove the container and the message is clear in informing the user to do so.	Pass

G.4 Dashboard Test Cases

Test Case ID	Related Dashboard	Test Description	Test Step	Expected Result	Result

Test Case ID	Related Dashboard	Test Description	Test Step	Expected Result	Result
DB-1	Business	The dashboard cells for today's revenue and stock by hour remain on the current day's data even if the user changes the date range using the GUI.	Ensure the charts for today's dispense data are populated, then attempt to change various date ranges, observing if any changes are seen in the graphs.	The graphs do not change when adjusting the date range.	Passed
DB-2	Business	The dashboard cells which display revenue and stock within the requested date range appropriately filter the displayed data.	Ensure the charts for any given date range are correctly filtering the data by various date ranges, observing the legibility is somewhat preserved.	The graphs change on user request and the x ticks/y ticks change appropriately.	Passed
DB-3	Business	The target stock/revenue is calculated from the previous days.	Send dispenses for a full day, then on the next day observe that the targets are correctly calculated from the previous day. Verify this using data explorer and filtering data by the previous day to get yesterday's total revenue/stock.	The targets are created using the previous day's totals.	Passed

Test Case ID	Related Dashboard	Test Description	Test Step	Expected Result	Result
DB-4	Business	If there were no dispenses on the previous day, target stock-/revenue falls back to default values.	Send no data for a full day, then check the dashboard and observe the target revenue and stock.	The fallback values are used for target revenue and stock.	Passed
DB-5	Business	The cells for today's totals are appropriately colour coded on the day's performance.	Observe the colour at the start of the day when the totals are 0, then observe the colour when the totals cross different thresholds. (£40, £180, £200 and 3000g, 5000g, 10000g)	The total's colour changes based on the day's performance.	Passed
DB-6	Business & Capacity	The data being displayed is accurate and correlates to the data being sent from the dispensers.	Send dummy data from a dispenser using the same code as production, verifying whether the data displayed/stored in InfluxDB is accurate through visual verification and verification using the data explorer. Also attempt this using real dispenses to ensure the dummy data result is replicated in real dispensing.	The data displayed is as expected based on the data sent from the dummy data code, and the actual dispensers sending real dispense data.	Passed

Test Case ID	Related Dashboard	Test Description	Test Step	Expected Result	Result
DB-7	Capacity	The capacities being displayed are accurate and correlate to the data being sent from the dispensers.	Send dummy capacities to the dashboard, verifying the capacities displayed correlate to the correct product and the displayed % is as expected. Then repeat this using capacities sent from the real dispensers, again observing the displayed capacity is as expected.	The capacities correctly map to the right product with the correct %.	Passed
DB-8	Capacity	Every hour the capacities sent within that hour are flagged if they are <10%, and a message regarding that product and its capacity is sent over Slack.	Send dummy and real capacities to the dashboard, before the hour time frame check the most recent capacities from the last hour which are <10% in the data explorer to see what notifications are expected. Observe notifications send as expected and a notification doesn't repeat every hour for capacities that are already flagged.	Notifications send as expected on the hour, repeats are not seen unless the low capacity has been recorded again.	Passed

Test Case ID	Related Dashboard	Test Description	Test Step	Expected Result	Result
DB-9	Capacity	Ensure the messages sent to Slack contain the correct information regarding a product and its capacities, and the message is sent with a warning flag.	Send capacities <10%, noting the % number and product name, verify on the hour that the notifications contain the expected number and product name.	Notifications contain the correct product name and capacity %.	Passed
DB-10	Business & Capacity	The dashboards can handle and display data coming from multiple dispensers at the same time.	Send dispenses from the liquid, solid and using the dummy data code at the same time. Once sent verify the dashboard handles this data appropriately and graphs still appear as expected, and capacities are correct.	The dashboard handles simultaneous dispenses and no errors are seen.	Passed

G.5 Self-Checkout Test Cases

Test Case ID	Test Description	Test Step	Expected Result	Result
SC-1	Scanning or inputting invalid RFID details should not redirect to checkout.	Scan and input invalid RFID details on the /home/scan-rfid page.	An error is displayed regarding the input/scanning of an invalid tag.	Pass

Test Case ID	Test Description	Test Step	Expected Result	Result
SC-2	Scanning or inputting valid RFID details should redirect customer to the appropriate checkout.	Scan and input valid RFID details on the /home/scanrfid page and verify the viewed basket is correct.	Redirected to the correct basket.	Pass
SC-3	Flask sessions should be cleared when /home/scanrfid is visited.	Create expected sessions and ensure they are cleared when visiting /home/scanrfid.	AdminAccess and checkoutSessionID sessions are cleared to prevent unauthorized access and incorrect details.	Pass
SC-4	Inputting a valid employee pin will allow access and redirect the user to the admin panel.	Pass a valid employee pin to the /home/employee_login page and do the same through the front-end.	The employee is logged-in and can access the admin dashboard.	Pass
SC-5	Inputting an invalid employee pin will not allow access to the admin panel.	Pass an invalid employee pin to the /home/employee_login page and do the same through the front-end.	The user is not permitted access to the admin panel.	Pass
SC-6	The route correctly determines a valid or invalid pin depending on the given input.	Attempt to use both valid and invalid pins and observe the response.	Valid pin allows access to the route attempting to be accessed whilst an invalid pin prevents access.	Pass
SC-7	Purchases are displayed correctly according to the RFID basket being viewed.	Navigate to the basket of an RFID tag with purchases.	Purchases are correctly displayed.	Pass

Test Case ID	Test Description	Test Step	Expected Result	Result
SC-8	A valid purchase ID can be removed from a customer's basket.	Pass a valid purchase ID to the remove_purchase route and attempt to remove a purchase through the front-end.	The purchase is removed from the customer's basket.	Pass
SC-9	An invalid purchase ID will display appropriate error messages.	Pass an invalid purchase ID to the remove_purchase route.	An error is displayed and no changes are made to the customer's basket.	Pass
SC-10	Valid item details will be added to a customer's basket.	Pass valid item and RFID details to the add_item route and attempt to add an item through the front-end.	The item is added to the customer's basket.	Pass
SC-11	Invalid item details will not be added to a customer's basket.	Pass various combinations of invalid item and RFID details to the add_item route.	An error is displayed and no changes are made to the customer's basket.	Pass
SC-12	Valid dispense details will be added to a customer's basket.	Pass valid dispense and RFID details to the add_item route and attempt to add a dispense through the front-end.	The dispense is added to the customer's basket.	Pass
SC-13	Invalid dispense details will not be added to a customer's basket.	Pass various combinations of invalid dispense and RFID details to the add_item route.	An error is displayed and no changes are made to the customer's basket.	Pass
SC-14	Valid customer tag with active purchases can be cancelled.	Pass valid RFID details with active purchases to the /checkout/cancel route and attempt to use the cancel button on the front-end.	Customer is redirected to the home page and a success message is displayed, the purchases have been removed from the database.	Pass

Test Case ID	Test Description	Test Step	Expected Result	Result
SC-15	Invalid customer tag cannot be cancelled.	Pass invalid RFID details to the /checkout/cancel route.	Customer is redirected to the home page and an error message regarding invalid tag details is displayed.	Pass
SC-16	Valid customer tag with no active purchases cannot be cancelled.	Pass valid RFID details with no active purchases to the /checkout/cancel route and attempt the same using the cancel button on the front-end.	Customer is redirected to the home page and an error message regarding no active purchases is displayed.	Pass
SC-17	Valid account customer tag with active purchases can confirm their purchase and use their account points.	Pass valid account RFID details with active purchases and using points to the /checkout/checkout_session/ route and attempt the same using the confirm button on the front-end.	Customer will be redirected to Stripe payment API with the correct order cost (PURCHASES COST - ACCOUNT POINTS).	Pass
SC-18	Valid guest customer tag with active purchases cannot confirm their purchase when trying to use account points.	Pass valid guest RFID details with active purchases and using points to the /checkout/checkout_session/ route.	An error is displayed informing the customer guest accounts cannot use loyalty points.	Pass

Test Case ID	Test Description	Test Step	Expected Result	Result
SC-19	Valid customer tag without active purchases cannot confirm their purchase.	Pass valid RFID details without any active purchases to the /checkout/checkout_session/ route and attempt the same using the confirm button on the front-end.	An error is displayed regarding the customer having no purchases in their cart.	Pass
SC-20	Valid guest/account tag with active purchases can confirm their purchase without using account points.	Pass valid guest and account RFID details with active purchases and not using points to the /checkout/checkout_session/ route and attempt the same using the confirm button on the front-end.	Customer will be redirected to Stripe payment API with the correct order cost.	Pass
SC-21	A negative cost cannot be passed to the Stripe checkout session.	Pass a negative number as the cost and create a Stripe checkout session.	Redirection to checkout with an error.	Pass
SC-22	When a customer cancels payment their basket remains as it was before confirming.	Confirm a valid customer basket then cancel payment on the Stripe checkout session.	Redirection to checkout with customer purchases unchanged.	Pass
SC-23	When a customer confirms payment the Stripe checkout session ID is kept in Flask a session to store within the database on successful payment.	Confirm a valid customer basket then check the checkout session ID is being stored within Flask sessions.	The checkout session ID is being stored within a Flask session.	Pass

Test Case ID	Test Description	Test Step	Expected Result	Result
SC-24	Upon successful payment confirmation of an order guest customers are redirected to the guest receipt.	Pay for an order on the Stripe API using test card details and verify the correct redirect is in place.	Customer is redirected to the guest receipt page.	Pass
SC-25	Upon successful payment confirmation of an order account customers are redirected to the account receipt and an email receipt copy is sent.	Pay for an order on the Stripe API using test card details and verify the correct redirect is in place and an appropriate email copy of the receipt is sent.	Customer is redirected to the account receipt page and an email is sent to the email assigned to their account.	Pass
SC-26	Upon successful payment confirmation of account orders, the account's points are correctly updated.	Pay for an order on the Stripe API using test card details and verify the database changes.	Database is changed with the correct order/purchase details.	Pass
SC-27	Upon successful payment of both account and guest orders the purchases are archived in the database and an order is created with appropriate details, including the Stripe checkout session ID.	Pay for an order on the Stripe API using test card details and verify the database changes.	Database is changed with the correct order/purchase details and Stripe checkout session ID matches ID on Stripe dashboard.	Pass
SC-28	Attempting to view the account receipt page with a non-existing order but a valid account tag will redirect to checkout and no email confirmation should be sent.	Send requests to receipt/account_receipt using a valid account RFID and an invalid order number.	Redirected to /checkout/checkout basket of the RFID with an error message.	Pass

Test Case ID	Test Description	Test Step	Expected Result	Result
SC-29	Attempting to view the account receipt page with a valid order and tag combination, but the tag does not belong to an account, should redirect to the checkout and there is no email attempt.	Send requests to receipt/account_receipt using a valid order and guest tag.	Redirected to /checkout/checkout and no email attempt is made.	Pass
SC-30	Attempting to view the account receipt page using a valid account tag and a valid order which is attached to another account should redirect the customer to checkout without attempting to send an email.	Send requests to receipt/account_receipt using a valid account tag and a valid order that belongs to a different tag.	Redirect to /checkout/checkout and no email attempt is made.	Pass
SC-31	The account receipt of a valid account tag and order displays the correct confirmation message and the email confirmation also contains the right order details.	Send requests to /receipt/account_receipt using valid account tags under different emails with valid orders, verifying order details are as expected in the sent emails.	The receipt information is correct alongside the information within the email.	Pass
SC-32	Attempting to access the guest receipt with a valid account and order will redirect to /receipt/account_receipt and send an email confirmation.	Send requests to /receipt/guest_receipt with valid account tag and order. Verify the redirect is as expected and the email confirmation is accurate.	The redirect and email confirmation are correct.	Pass

Test Case ID	Test Description	Test Step	Expected Result	Result
SC-33	Accessing the guest receipt with valid guest tag and order combination displays the order contents as expected.	Send requests to /receipt/guest_receipt with valid guest tag and order, then verify the displayed information is correct.	The receipt displays with the correct information.	Pass
SC-34	On successful sign-up the confirmation message contains the appropriate and correct information.	Sign-up on the guest receipt page and confirm the displayed information.	The receipt displays with the correct information.	Pass
SC-35	When signing up with valid credentials on the guest receipt page the created account has the correct amount of points attached.	Sign-up on the guest receipt page and confirm in the database that the account has the expected points amount attached to it.	The correct points are attached to the account.	Pass
SC-36	When signing up with valid credentials on the guest receipt page the created account has that orders dispenses attached to it.	Sign-up on the guest receipt page and confirm in the database that the account's attached orders are as expected.	The order/dispenses are attached to the account.	Pass
SC-37	When signing up with valid credentials on the guest receipt page the created account has the RFID tag used when completing the order attached to it.	Sign-up on the guest receipt page and confirm in the database that the account's RFID is as expected.	The account's RFID is the tag used for the purchase.	Pass

Test Case ID	Test Description	Test Step	Expected Result	Result
SC-38	When signing up with valid credentials on the guest receipt page the email used in creating the account receives a confirmation email.	Sign-up with an email you have access to and verify the details of the email confirmation.	The email is sent and the details are accurate.	Pass
SC-39	When signing up with invalid credentials no account is created and the relevant error is displayed.	Attempt to sign-up using an email which is already in use, then attempt to sign-up using an invalid email string.	No sign-up is complete and an error regarding the email being invalid is displayed.	Pass
SC-40	/refund/ validate_refund correctly determines invalid/valid refund requests and displays the appropriate error message.	Attempt to request a refund for a valid order which has not been refunded, a valid order which has been refunded, and an invalid order.	The invalid refunds display an error message and the request goes no further, valid refund gets passed onto /refund/refund_session.	Pass
SC-41	A validated refund request on an order which only required Stripe payment is processed using the checkout session ID which gets payment intent ID.	Request a refund for multiple orders which only used Stripe/currency payment and verify in Stripe dashboard the refund is processed on the correct order.	The refund is processed on the correct order.	Pass
SC-42	A validated refund request on an order which only used points refunds the appropriate amount of points to the user.	Send multiple valid refund requests using different point amounts, verifying within the database that the points amount is correct.	The correct amount of points are maintained and the refund is shown on Stripe dashboard.	Pass

Test Case ID	Test Description	Test Step	Expected Result	Result
SC-43	A validated refund request on an order which required Stripe payment and points is processed using the checkout session ID which gets payment intent ID, and the appropriate point amount is refunded.	Send multiple valid refund requests using different money and point amounts, verifying within the database that the points amount is correct and within Stripe dashboard that the refund request is sent.	The correct amount of points are maintained and the refund is shown on Stripe dashboard.	Pass
SC-44	On successfully processing a refund an email confirmation is sent if the tag belongs to an account.	Observe the app sends an email with the correct details after processing a refund.	The email confirmation is sent to the correct email with the correct refund details.	Pass
SC-45	After successful refund processing a message is displayed and the app redirects to the home page.	Observe the app after successfully processing a refund.	Redirected to /home/scanrfid with the appropriate success message.	Pass
SC-46	After successful refund processing the order which is refunded has order_refunded set to true.	Verify the order_refunded field is set to true after refunding multiple orders with different combinations of points and cost.	The order_refunded field is set to true after successfully processing a refund.	Pass
SC-47	Accessing the admin panel requires authorization.	Attempt to access the admin panel (/admin) without logging in.	Redirected to the home page and an error is displayed.	Pass
SC-48	Hitting logout on the admin panel removes the login session/authorization.	Log-in to the admin panel, hit the logout button, then attempt to access /admin without logging in.	Access will be denied and user will be redirected to the home page with an error message.	Pass

Test Case ID	Test Description	Test Step	Expected Result	Result
SC-49	When an employee changes the RFID UID attached to a customer account their current and archived purchase history is changed to the new UID.	Change the UID of an account with both active and archived purchases/orders.	The UID attached to the account's purchases/archived purchases is changed to the new UID, this change is only made on purchases made after the account's creation.	Pass
SC-50	Employee changes on Flask-Admin are appropriately reflected within the database.	CRUD account details, item details, RFID tag details, employee details, and dispenser details on the admin panel.	The changes are reflected in the database.	Pass

H Evaluation, Further Work, and Conclusions

H.1 Liquid Dispenser Cost Estimation

Hardware	Cost
ESP32	£2.69
12V Peristaltic Pump	£3.40
SSD1306	£1.69
HC-SR04	£0.86
VL53L1X	£2.69
5KG load cell with HX711	£1.31
MFRC522	£0.73
Total Cost	**£13.37**

Costs based on prices shown at **AliExpress (2025)**[2], buying multiple units would further reduce prices.

H.2 Solids Dispenser Cost Estimation

Hardware	Cost
ESP32	£2.69
1602 LCD	£0.75
HC-SR04	£0.86
5KG load cell with HX711	£1.31
MFRC522	£0.73
Total Cost	**£6.34**

Costs based on prices shown at **AliExpress (2025)**[2], buying multiple units would further reduce prices.

H.3 Self-Checkout Cost Estimation

Hardware	Cost
Raspberry Pi 4B 4GB RAM	£48.39
16" Touchscreen	£72.98
MFRC522	£0.73
10pcs 13.56MHz RFID tags	£0.79
Total Cost	**£122.89**

Costs based on prices shown at **AliExpress (2025)**[2], buying multiple units would further reduce prices.

H.4 Supervisor Meeting Logs

Date	Progress So Far	Topics Discussed	Next Steps
03/10/24	Pitched my project idea to get feedback and see if it was appropriate to go ahead with.	Logistics of the project, software/hardware you might use, possible challenges.	Order the hardware required and begin collecting background research for literature review.
07/10/24	Gathered a reasonable amount of background literature and am waiting for hardware to arrive.	How to write an effective literature review, discussing our project ideas with the group for feedback.	Continue gathering literature and begin writing a draft of literature review.

Date	Progress So Far	Topics Discussed	Next Steps
09/10/24	Begun writing my literature review as I had gathered plenty of sources.	Concerns I have for my project, such as power concerns with having a lot of peripherals.	Create project proposal for review next week, continue work on literature review.
23/10/24	Created my project proposal and prepared it for review.	Approval of the project proposal and how I can improve it before submission.	Submitting project proposal, begin testing hardware now it has arrived and continue work on literature review.
30/10/24	Made changes to project proposal following last week's discussion, continued working on my literature review and was able to start work on the hardware side of my project.	Submitting project proposal, discussed our progress on the literature review as a group and whether we should start with literature review or the software/hardware aspects of the project.	Begin working on literature review and hardware side by side and submit project proposal.
13/11/24	Started working on the hardware but realised I needed more parts, so I am waiting on those to arrive. This allowed me to focus on my literature review and I was able to finish a first draft.	Common mistakes in literature reviews, further hardware considerations to prevent me needing to wait for another shipment.	Refine my literature review and begin working on the hardware once parts arrive.
20/11/24	Made some good progress on the hardware, peripherals now receive enough power and the IoT side of the project has progressed nicely with the MQTT server working.	How to stay on top of the workload and balancing other modules. Discussed possible solutions to some issues I was experiencing with the hardware and what alternatives are on the market.	Continue working on the hardware and begin work on the software side of the project (the checkout system).
04/12/24	Steady progress on both the hardware and software.	Evaluating the current progress of my project, next steps in writing my report.	Continue work on hardware and software, consider next steps for report.

Date	Progress So Far	Topics Discussed	Next Steps
18/12/24	Steady progress on both the hardware and software.	Email communication regarding aims, objectives and requirements. Also, communication regarding project-in-progress.	Continue working on project over the Christmas break, primarily focusing on the project-in-progress deliverables.
15/01/25	Finalising project-in-progress submission.	Email communication regarding concerns with project in progress, such as requiring a state diagram.	Develop state diagram and improve design of my poster. Also record the video submission.
07/02/25	Project-in-progress finished, beginning write-up of design chapter. Implementation progressing steadily.	Discussed whether my proposed set of diagrams is comprehensive enough to present my project. Also discussed possible topics to cover in my design chapter.	Finish design chapter and continue with implementation.
26/02/25	Design chapter draft finalized.	Asked a few final queries before moving to implementation chapter.	Take feedback on board, finish majority of project implementation, then begin implementation chapter.
06/03/25	Progressing on implementation and write-up.	Discussed whether removing automatic dispensing is the right option.	Conduct SWOT analysis on whether automatic dispensing in its current state is viable to remain in the project. Finish implementation and write-up.
19/03/25	Finished draft of implementation chapter and almost all implementation.	Queried whether parts of my implementation chapter could be strengthened, such as my discussion of tests.	Finalize implementation chapter and implementation. Finally, write project's evaluation and conclusions.