B, James

|

Realtime Computer Vision For Use in a Retail EPOS

Computer science a2 project

Contents

[Project Analysis 3](#_Toc92699719)

[Project Proposal 5](#_Toc92699720)

[Target Audience 8](#_Toc92699721)

[Similar Products 9](#_Toc92699722)

[Amazon Go 10](#_Toc92699723)

[What I like about this solution 11](#_Toc92699724)

[What I don’t like about this solution 11](#_Toc92699725)

[Amazon Dash Cart 12](#_Toc92699726)

[What I like about the solution 12](#_Toc92699727)

[What I don’t like about the product 13](#_Toc92699728)

[Scan as You Shop Technology 14](#_Toc92699729)

[What I like about this 14](#_Toc92699730)

[What I do not like about this 14](#_Toc92699731)

[Conventional Self-Checkout Systems 16](#_Toc92699732)

[What I like in this 17](#_Toc92699733)

[What I do not like in this solution 18](#_Toc92699734)

[Success Criteria 19](#_Toc92699735)

[Critical features- (must be included for the solution to function) 20](#_Toc92699736)

[Ideal features- (nice to have with some impact on the efficacy of the solution) 21](#_Toc92699737)

[Additional features- (no massive impact on the project but would be nice to have) 22](#_Toc92699738)

[Project Design 23](#_Toc92699739)

[Disassembling the Project 24](#_Toc92699740)

[The Structure of My Solution 25](#_Toc92699741)

[Front End 25](#_Toc92699742)

[Back End 27](#_Toc92699743)

[Flow chart for the UX 29](#_Toc92699744)

[Pseudocode for Key Elements 31](#_Toc92699745)

[Creating a Computer Vision Model 34](#_Toc92699746)

[Protecting Customer and Client Information 36](#_Toc92699747)

[Technological Requirements for my solution 37](#_Toc92699748)

[How my Project will be managed. 39](#_Toc92699749)

[Time Management 40](#_Toc92699750)

[Testing 43](#_Toc92699751)

[Key Variables 44](#_Toc92699752)

[Iteration Review 1 46](#_Toc92699753)

[Testing Iteration 1 52](#_Toc92699754)

[Normal Function Test 52](#_Toc92699755)

[Extreme Excess Items Test 52](#_Toc92699756)

[Camera Error Testing 53](#_Toc92699757)

[Iteration Review 2 54](#_Toc92699758)

[Testing Iteration 2 59](#_Toc92699759)

**THIS IS NOT MY FINAL WRITE UP FOR THIS SOLUTION, THE SUBSEQUENT VERSIONS CONTAINED A LOT OF EMBEDDED INFO WHICH FOR PRIVACY REASONS COULD NOT BE SHARED. THIS VERSION IS LARGELY FINAL OTHER THAN TESTING AND EVALUATION.**

# Project Analysis

## Project Proposal

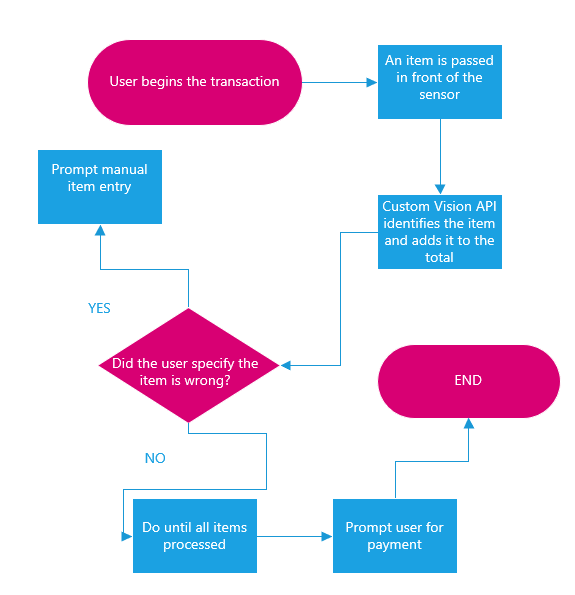
For my project I aim to create a windows forms-based application to recognise items as they are put into a shopping basket. It is designed to replace the need for cashiers and scanning the items and instead a camera (which would be fitted to the basket) will take a picture of the item as it is put in the basket and use Microsoft Azure Custom Vision to identify the item and automatically add the item to a customer’s cart. The UI will aim to be customer focussed to be as easy to use as possible as they are unlikely to be familiar with the technology and so a comforting and informative system will reduce the burden on the client’s staff and maximise the effectiveness of the solution.

At its fundamental basis the Custom Vision tool uses artificial intelligence by teaching the algorithm; building on their prefabricated bases, to recognise the specific items that the customer passes in front of the camera.

Conventional barcodes report a string of numbers or letters that uniquely identify each item. This is traditionally seen as the most effective method however I think I can build on it by circumventing the need and time to locate the barcode and scan it. However, I will return a string to uniquely identify the product which will be referenced against a database.

My product will aim to be an off the shelf style solution that could be deployed to any type of retail store such as a clothes store or a DIY supplier. Similar products are typically only used for conventional food retailers, but I believe I can build on this by allowing stores to create their own computer vision model. By using the web portal already built by Microsoft which features easy step by step hints and I will aim to produce documentation to supplement this.

The program takes the role of the conventional checkout and instead of detecting barcodes it will take a picture of the object as it passes and then add it to the till total. The UI will aim to be as simple as possible to make it easy to train staff. It will use buttons to remove items as well as offering a manual input option as a fall back in case there is an issue with an item or if an item is damaged affecting the recognition capabilities. The program will also feature the standard POS functionality including payment processing.



## Target Audience

My target is small to mid-sized retailers looking to save money by reducing the need for cashiers or complex self-checkouts. I want it to be easily accessible using simple UI/UX so that any age or technical ability of customer could easily make use of the solution without depending on help from IT support which would hinder the overall effectiveness of the product.

As my target audience is the stores themselves, they are unlikely to have programmers and developers in their staff and so I need to focus on making it easily accessible. By using a user-friendly platform for the custom vision component of the project (customvision.ai) the end user can easily create their own models to adapt the solutions to their individual needs and products.

For the store deploying the solution I want to maximise the ease of use and so I will aim to create clear documentation and well commented code for them to use my solution as effectively as possible.

A key use for this product is in shops where customers/staff may not use barcodes and instead must manually search a database for a product. If products are known by many names or are particularly complex, my product reduces the risk of customers being incorrectly charged for their goods as the program can more accurately and quickly identify the product. An example of this might be a bakery which sells many similar looking bread rolls at difference prices. At a self-service checkout a customer might manually input the wrong product either to choose a cheaper option in order to cheat the system or accidentally out of confusion. My program will largely aim to eliminate this mistake.

# Similar Products

## Amazon Go

Amazon have a small chain of store which feature similar object detection technology. They use a range of sensors including cameras, weight sensors and unique QR codes to detect the items as a customer pick them up. They aim to greatly increase the speed at which someone can shop as well as saving costs by not having cashiers.



QR codes act as an identifier for each customer as they enter the store and they are then tracked by cameras so that it can track what they have bought.

The system does not have any visible UI other than a QR code to access the store and the digital receipt sent after the customer has left the store

### What I like about this solution

I like the use of the QR style codes on similar looking products to make it easy for the cameras to recognise the items and this may be beneficial to my own solution. Secondly, I like the lack of need for customers to input their information every time they enter the store and instead, they scan a QR code, this may not be within the scope of my project but using a unique pin may be an effective alternative compared to having customers use chip& pin

### What I don’t like about this solution

The large number and variety of sensors used in the store make it a very expensive product for clients to use. By focussing on a highly accurate computer vision model it will keep the cost for potential clients relatively low, making the solution far more economically viable and easier to install without product redesigns and store modifications. My solution will not rely on the array of cameras and as such I will not be able to implement the automatic recognition as if it were targeted for an item being picked up.

If this technology became more commonplace, I think it would highlight 2 major limiting factors, first the widespread use of tracking technology would invoke fears of surveillance and data harvesting by Amazon as a for-profit company how may be motivated to target users based on their shopping habits. And secondly if someone does not have a phone or forgets it, they are unable to use the store potentially limiting their market.

The lack of UI for transactions I think negatively affects the customer experience. As the technology is very new, I believe people would prefer to see they are being charged correctly and that the program is working rather than just assuming that it is, and users have no way to check their total whilst shopping

## Amazon Dash Cart

Amazon dash cart is an expansion of their Amazon Go “just walk out” technology. It uses a “smart cart” which recognises the items using weight sensors, computer vision and an array of cameras. It allows around 2 bags of shopping as the cart is made to be deliberately small.



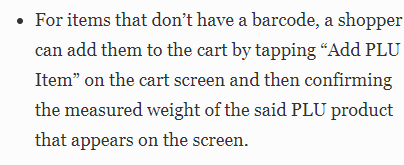
This cart is close to what I wish to design, however I feel like I can build upon the technology by instead of focusing on building solely for a food retailer to instead creating a universal product that is more dynamic.

### What I like about the solution

The use of clear images in the cart UI makes it intuitive for customers to use and the reduced reliance on masses of sensors compared to Amazon Go makes it a simpler solution for deploying to stores. I will aim to also use clear images for my UI to improve the end user experience.

### What I don’t like about the product

The use of scales in the cart is likely to be problematic as if a customer leans on the cart or places an unexpected item inside it then customers may be charged the wrong price, causing problems for the client.



Forbes 2020

A major drawback is that some items in the store which do not have barcodes must be manually entered using an onscreen display. This is a major limitation as it hinders the “magic” of the solution; the ease and intelligence of using it. I want to bypass this by relying on physical captures of the product itself and not just the barcode. While my solution will have a fall back to manually identify the product, I will instead aim to have it as a more intuitive search system that can be easily used.

## Scan as You Shop Technology

Major retail supermarkets have recently implemented “scan as you shop” which allows customers to scan products as they pick them up to automatically add them to their virtual cart. It relies on a customer manually scanning the barcode on each item and essentially acts as a very stripped-down version of a typical checkout system and once done shopping the total is transferred to a more typical checkout system where the payment is processed.



### What I like about this

The solution is easy and use friendly, it has audible feedback to make it easy for users to understand when an object has been scanned. I will also use the audible feedback for when an Item has been detected.

### What I do not like about this

This relies on being able to find barcodes and so on particularly difficult items such as Cadburys Crème Eggs the barcode is nearly impossible to scan and so must be set aside till the end of the shop where it then inconveniences customers and staff at the checkout. By using custom vision my product avoids this issue.



A further limitation is that theft and shoplifting both intentional and unintentional can easily occur with this solution. A customer may forget to scan an item as they place this in their cart or they may stow it discreetly to hope to skip the service checks which are used to target this issue but act as a further cause of grief inhibiting the use of the system.

My project’s scope will be limited and instead to focus on building on the self-checkout ideas used in this. The product will rely on the items being passed in front of a checkout camera to add them to the total.

## Conventional Self-Checkout Systems

This technology has been around for a long time and allows the customer to act as their own cashier, scanning their own products and manually inputting payment either through chip and pin or cash systems. This however is limited again by the conventional use of barcodes and responsiveness issues in some stores with earlier systems. Weight sensors are used as a theft deterrent to detect when an unexpected item is placed in the bagging areas. Members of store staff must manually authorise payments where the user is purchasing an age restricted item or triggers the weight sensors to think an extra item is in the bag.



An earlier example of a self-checkout system, where these remain in use, they are a source of user frustration as they are usually slow and prone to errors.

A more modern example using newer technology and often not accepting cash, reducing cost and bulk. These can be very effective and easy for the user to use.

### What I like in this

The system is already commonplace making it easy for stores to adopt and train staff to use. The modern systems are very responsive to use and are significantly faster to use than traditional cashiers. They also encourage the use of card or device payments and as this is much faster it helps to benefit the client by allowing them to process more customers in a shorter amount of time increasing profits. The UI is clear and intuitive making it easy to use even for first time customers. Although it is not particularly attractive, it does include the required functionality and each button is clearly labelled.

### What I do not like in this solution

The theft deterrence in the scales commonly flags accidents such as a customer leaning on the scales, acting as a large inconvenience, I will omit this from my solution and instead focus my scope solely upon the EPOS aspect of the product. Furthermore, the tills may be slowed down by unfamiliar users who do not know where the barcodes are located on their items or how to use the UI. Which is why I am using computer vision and will aim to make my UI as customer friendly as possible.

# Success Criteria

## Critical features- (must be included for the solution to function)

* Recognise items and add them to a virtual list.

This is fundamental to my project and includes taking images of the products (scanning), sending them to the computer vision model and deciphering the result for use in the program

* Allow manual item entry as a fall-back if the recognition is not functioning

This is required as the custom vision is a new technology and given my limited time scope, I won’t be able to build a perfect custom vision model. Manual entry will aid bug testing and dealing with damaged or unexpected products

* Process payment

The till will need to be able to process payments as this is a basic feature of POS. Given the increase in a cashless society I will aim to only process card payments. For the purpose of designing the solution I will not be able to connect to a payment processor, but my program will collect the card information (and potentially encrypt it).

## Ideal features- (nice to have with some impact on the efficacy of the solution)

* Sending digital receipts to customers once the transaction is complete

This would require connecting to an email API to generate a virtual receipt based on a customer’s basket, total and email address. (API SendGrid would be promising for this)

* Customer database to collect rewards etc

This would not include input functionality as that would be a different program, but customers would be able to enter an identifier and pin to use saved payment info or access rewards

* Storing previous transactions into a database

Key for sending email receipts and allows analysis of sales e.g. stock replenishment and managing waste.

## Additional features- (no massive impact on the project but would be nice to have)

* Saving card information based on customer database

This would present a challenge as this information must be encrypted but would potentially be a good feature to add if there was time as it would increase customer convenience.

* Admin backend to debug tills when in use

This would make it easy to test the solution including resetting a transaction or restarting the program.

# Project Design

# Disassembling the Project

My Project will be writing in C# based on a Windows Form using Visual Studio 2019. To complete the project in the given timeframe I will break it down into 4 iterations and use planning tools including GANTT and PERT charts to help me keep within the required timeframe.

At its most basic form my project captures an image, identifies an object in the image and then returns what it is. Within Iteration 1 these will be my primary goals, alongside producing the computer vision model necessary for this project. I have broken this into further detail below.

More complex features including a database to store customer information and payment processing I will aim to add in the last iteration but are not a priority as they do not play a particularly significant role in the success of my solution. [See: Required features of my solution](#_Required_Features_of)

I will perform security and reliability evaluation at the end of every iteration in order to look for bugs and vulnerabilities to correct in the next iteration. As my product is used in a commercial environment and handles payment it needs to be protected from hacking and should be reliable so that the program is quicker and easier to use than conventional POS as laid out in the Project Analysis.

## The Structure of My Solution

Below I have broken down the logical flow of my program as a user would experience it (based on all iterations being fully met).

### Front End

The user would first press a touch to start button that would load the main program. This would show a screen with the camera view, a total of what they have scanned in an itemised list and include prompts for manual item entry.

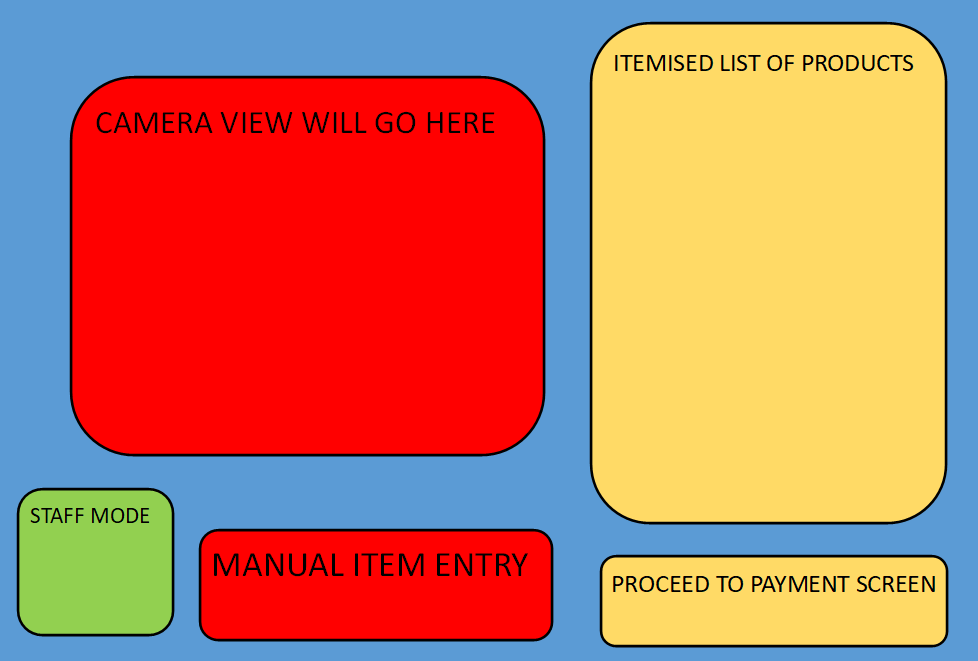


Figure 1 shows a layout for how I intend to design the main screen including all buttons and features I anticipate needing (Does not represent desired colours or style)

The manual item entry button will open a keyboard allowing the user to manually search the product database if their item is not scanning correctly and acts as a fall-back if the camera fails. I intend to use the Microsoft Windows On Screen Keyboard which is present on all recent windows OS’s including embedded. If this is not possible, I will design my own keyboard based on samples online. (This will rely on Oledb and SQL to search the products).

The camera view allows users to see what the camera sees, helping them to better position items as they pass them over to more easily get accurate and efficient use of my solution.

The itemised list of payments will show the users total and is a fundamental requirement in any self-serve EPOS. It will show what product has been scanned and its price. This will also help to build user confidence with the new technology as it instantly assures them that the program is working.

The Staff button will require a pin to access and will allow items to be removed or prices to be overridden in case the prices are incorrect in the database or the store is running a promotion.

Finally, the proceed to payment button load the payment window for a customer to pay.

Once on the payment screen the user will either be able to enter a member code to pay using stored card information or will be able to manually enter their card information. This will use encryption to ensure all information is stored safely. For the purpose of demonstration and testing my product will not hold any real card information and will not connect to any payment API’s.

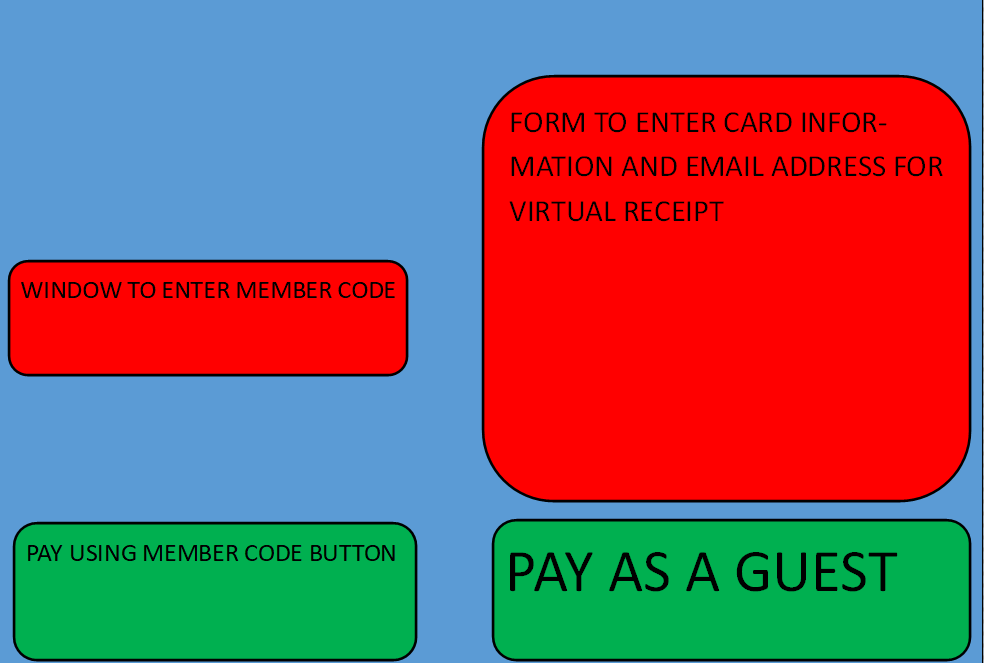


Figure 2 This mock-up shows how I intend to layout my payment screen (Does not represent desired colours or style)

The form to enter card information will consist of textboxes prompting all relevant information. I may include a diagram to help users identify the relevant information on their card in order to enter it.

The payment via member code would likely be supported using a changing code like [two factor authentication](https://authy.com/what-is-2fa/) whereby codes change every 30 seconds to prevent theft from shoulder surfing and hacking however due to time limitations this is out of the scope of my project and as such will not be included. Instead, the client might choose to incorporate this into their own apps.

When the user presses the pay button it would encrypt the card information and confirm that the payment has been successful or display a relevant error.

### Back End

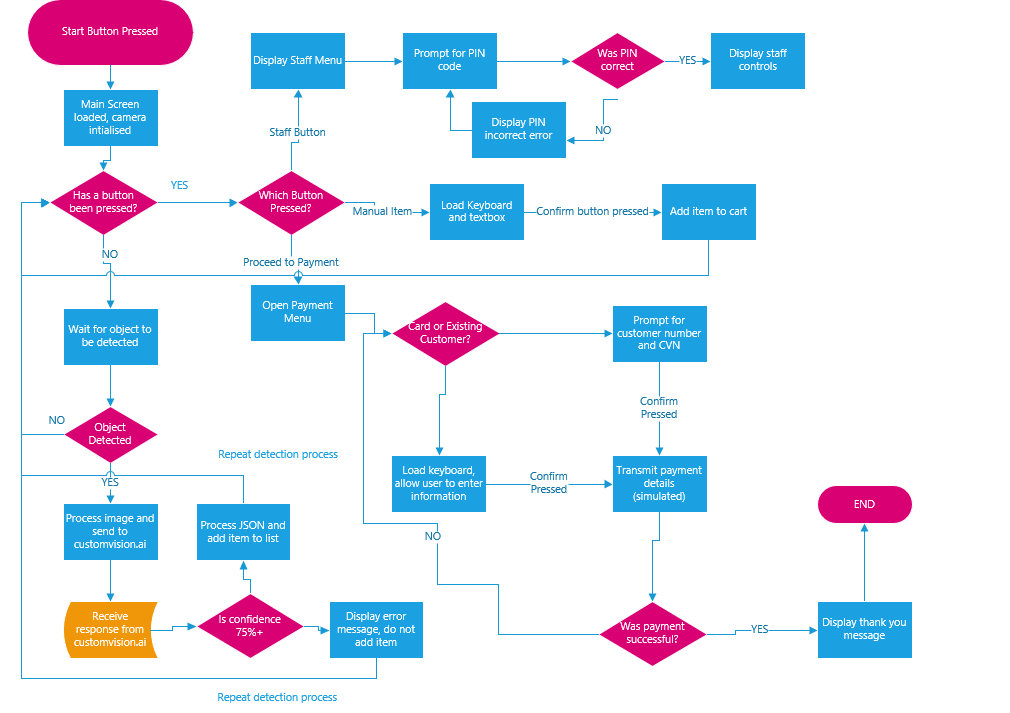
Each time an item is passed in front of the sensor (which will be simulated by a button for the purpose of demonstration) the program will capture an image and run a command line script to transmit the image to the Microsoft customvision.ai cloud platform

This script uses demos from Microsoft documentation adapted for my context to quickly and accurately determine the item in the image. I will verify the accuracy using a simple comparison operation using the accuracy extracted from the custom vision query. I will aim for an 80% threshold and if the confidence is below this then I will prompt for the image to be scanned again.

The backend will also include [Oledb database queries](https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms713643(v=vs.85)) for product and customer databases allowing me to run SQL queries, such as finding the price of a product based on the tag identified.

The backend for payment will not be functional, allowing clients to implement their own code to meet their needs and relevant security requirements in different countries.

## Flow chart for the UX



[Link to the flow chart](file:///\\eastnorfolk.ac.uk\Students\Home\start2020\57073\Computer%20Science%20Coursework\User%20experience%20flowchart.vsdx)

This chart shows the majority of all the functions and systems utilised in my project. I will use this in the development and testing of my project to ensure that I include all features and to make the user experience and code as efficient as possible by only loading relevant elements and making sure a user can flow through the program effectively. Prompts such as back buttons will be available on a range of screens, but these have been abstracted to simplify the flowchart somewhat.

## Pseudocode for Key Elements

<PSEUDOCODE STARTS>

When sensor detects motion //This is simulated by a button for demonstration.

String imagefilepath = C:/…

Capture image

Save image to imagefilepath

ReadImage(image)

End

ReadImage(image)

OpenProgram(sendimage) //This code opens the separate program which sends the image to the //customvision.ai

ProgramParameters = (imagefilepath)

String ProgramResponse = (Programexitcode)

Return output to main program.

End process

<PSEUDOCODE END>

This Code is an abstraction of the main image detection script. It detects and saves the image and then opens the image sending script. When that script gets a response, it automatically closes, and this program reads the response into the string ProgramResponse

<PSEUDOCODE START>

Processcardinfo{

Runprogram(onscreenkeyboard)

String Cardnumber = encrypt(cardinfotextbox)

String Nameoncard = nameoncardtextbox

String expiry date = expirydatetextbox

String cvv = encrypt(cvvtextbox)

//Sendcardinfo(Cardnumber,Nameoncard,expirydate,cvv) This part will not be functional to allow the client to connect the payment system to their own.

If sendcardinfo.response = 200 //if cardinfo processed ok

Messagebox.show(paymentsuccessfulmessage)

Else

Messagebox.show(Paymentfailedmessage)

endif

}

<PSEUDOCODE END>

<PSEUDOCODE START>

LoadCustomerInfo

String CustomerNumber = encrypt(CustomerNumberTextbox)

RunSQL( SELECT \* FROM CUSTOMERDATABASE WHERE CUSTOMERNUMBER = CUSTOMERNUMBER)

String expiry = sqlquery.expiry

String cardnumber = sqlquery.cardnumber

String nameoncard = sqlquey.nameoncard

String cvv = encrypt(cvvtextbox) // Should not store cvv for security purposes

//Sendcardinfo(Cardnumber,Nameoncard,expirydate,cvv) This part will not be functional to allow the client to connect the payment system to their own.

If sendcardinfo.response = 200 //if cardinfo processed ok

Messagebox.show(paymentsuccessfulmessage)

Else

Messagebox.show(Paymentfailedmessage)

endif

<PSEUDOCODE END>

These 2 scripts will be included on the payment screen to allow card information to be processed to allow the customer to pay for their goods. They also show a confirmation confirming whether payment was processed successfully. This enables the payment functionality of my project; I will use clear confirmation with large text and representative colours (i.e., green for successful payment) so that a user can clearly see that the program has worked. This should prevent people accidentally walking away without paying and reassure users that the new technology works.

## Creating a Computer Vision Model

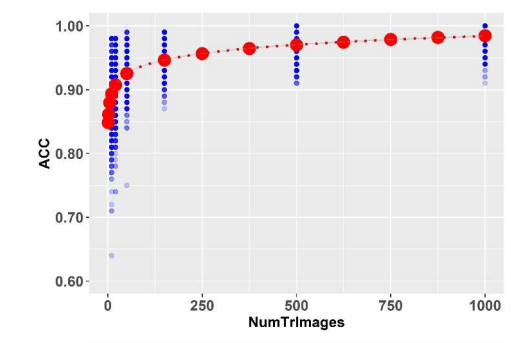
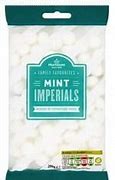
Based on a study by Cornell University ([How many images do I need? Understanding how sample size per class affects deep learning model performance metrics for balanced designs in autonomous wildlife monitoring](https://arxiv.org/abs/2010.08186)) I decided to use 250 images of each object for my demonstration model as this will give me a satisfactory level of accuracy while being realistic for the time, I have available. According to the image below I should expect to see an accuracy of around 96% which for demonstration purposes will be plenty to create an accurate model in a reasonable time frame for my project. For the client I would recommend at least 750 images as they need to maximise accuracy to ensure the solution is as accurate as possible.

Figure 3 from the Cornell study, I can expect an accuracy of around 96% from 250 images

For the photos themselves I chose 5 similar but different objects to present a reasonable challenge to the program while still giving it some factors to differentiate the products. I chose 5 bags of sweets with similar packaging but distinct features and/or colouring (see below). The bags are also flexible and so would be presented to the camera at a variety of angles. This will further challenge the model to demonstrate its accuracy. The images are taken in a variety of lighting poses and angles as specified by the [Microsoft documentation for customvision.ai](https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/getting-started-improving-your-classifier). I will also include negative photos of empty space or unrelated objects such as hands so that in the event of a miss-scan by the solution it can report this and reduce the chance for a false positive occurring.



These sample images show a similar selection of sweets to those that I will use for my project. The varied colour and size of the bags will help to enable the model to differentiate between products while the similar shape and in some cases (marshmallows and mint imperials) pale contents will demonstrate that my solution can handle challenging objects.

## Protecting Customer and Client Information

In order to protect customer information, I will not store a customer’s entire card details in the database. I will require them to enter their CVV every time they pay. This means that even if my database was hacked, the card numbers would be useless. Secondly, if time allows it, I will incorporate encryption for both stored information and manually entered information in my database. This means that the card information is protected when transmitted to process payment and adds a layer of complexity to protect the database. The product database will not be encrypted as it will not contain any confidential information, just the prices and names of each product alongside a unique identifier.

## Technological Requirements for my solution

As my solution is based on a windows form it will have to run on a windows operating system however the standard windows 10 home/pro contains a lot of unnecessary processes and features making it require more powerful hardware and increasing the cost for my solution. Instead, Windows 10 IOT Enterprise would be better suited as a target operating system for my solution as it has much lower hardware requirements (set out below). These requirements are sourced from [Microsoft documentation](https://eastnorfolk-my.sharepoint.com/personal/57073_eastnorfolk_ac_uk/Documents/CS%20Project%2030-6-2021/Windows%2010%20Minimum%20Hardware%20Requirements.pdf). As a result of using a cut down operating system my solution can run on much cheaper hardware, reducing the cost and will also be much faster on hardware as unnecessary processes such as windows analytics will not be using up processor power.

|  |  |
| --- | --- |
| System Requirements for my solution | |
| Processor | 400Mhz or faster, supporting x86/x64 instruction set |
| Memory requirements | 2gb (windows requires 768mb however to ensure no issues streaming from a camera 2gb would be best suited) |
| Storage | 50gb (although windows requires just 2gb, ensuring plenty is free for updates and installation of my solution is important. The price of solid-state storage has also fallen drastically making 50gb very affordable. |
| Display | A minimum 10-inch display is needed to display my program clearly and to enable usability. Touch support is also recommended to allow the user to interact with the program. |
| Output Devices | Speakers- these are required to provide audible feedback in the form of a beep to the user when an item is scanned. |
| Camera | A HD webcam (720p) is required to capture images at a sufficient quality in order to accurately identify the product while also being a small enough file to easily be transmitted to customvision.ai. |

These requirements are approximated based on Windows 10 Enterprise and for the purpose of development my project will be built on windows 10 Home.

# How my Project will be managed.

My project is broken into 3 iterations to make it manageable.

Iteration 1- This iteration focuses on the fundamental UI and basic features such as image capture in my project and handling the responses received from customvision.ai. I will aim to complete this iteration by the end of W/C Oct 18th. This includes 1 week of contingency time if any part of the iteration is particularly challenging and requires more time than expected.

Iteration 2- This iteration adds further relevant features to my project such as the payment handling, staff screens and creating databases to back up my solution. These are necessary to make my solution functional but the standards for the staff screen aren’t as high as it is not a critical feature and so other features can be prioritised if I risk overrunning. I have also again included a week of contingency time to further help me remain within my timeframe. I aim to finish this iteration by the end on W/C Dec 13th.

Iteration 3- This iteration adds the final functionality to my project, connecting relevant databases to the solution through SQL and adding the ability to store transaction history. Both of these features are non-critical so can be reduced down if necessary to make my project more manageable. I aim to finish all this including all the necessary reviews such as the final project review by the end of W/C Feb 7th.

# Time Management

To manage my time, I have created a GANTT Chart and a PERT Chart to allow me to break my tasks down into steps and manage my time. The GANTT chart gives me a rough idea of how far along my project should be by a certain date while the PERT chart helps me make sure I’ve completed all relevant tasks before progressing and means that if I am stuck on a particular task, I can attempt other non-dependent tasks while I try to solve the problem.

[GANTT CHART.xlsx](file:///\\eastnorfolk.ac.uk\Students\Home\start2020\57073\Computer%20Science%20Coursework\GANTT%20CHART.xlsx)

[PERT CHART.vsdx](file:///\\eastnorfolk.ac.uk\Students\Home\start2020\57073\Computer%20Science%20Coursework\PERT%20CHART.vsdx)

By using these planning tools, I can aim to complete my project effectively within the AGILE methodology. Each iteration builds upon the features created within the previous phase so that by completion of my project the solution is as complete as possible.

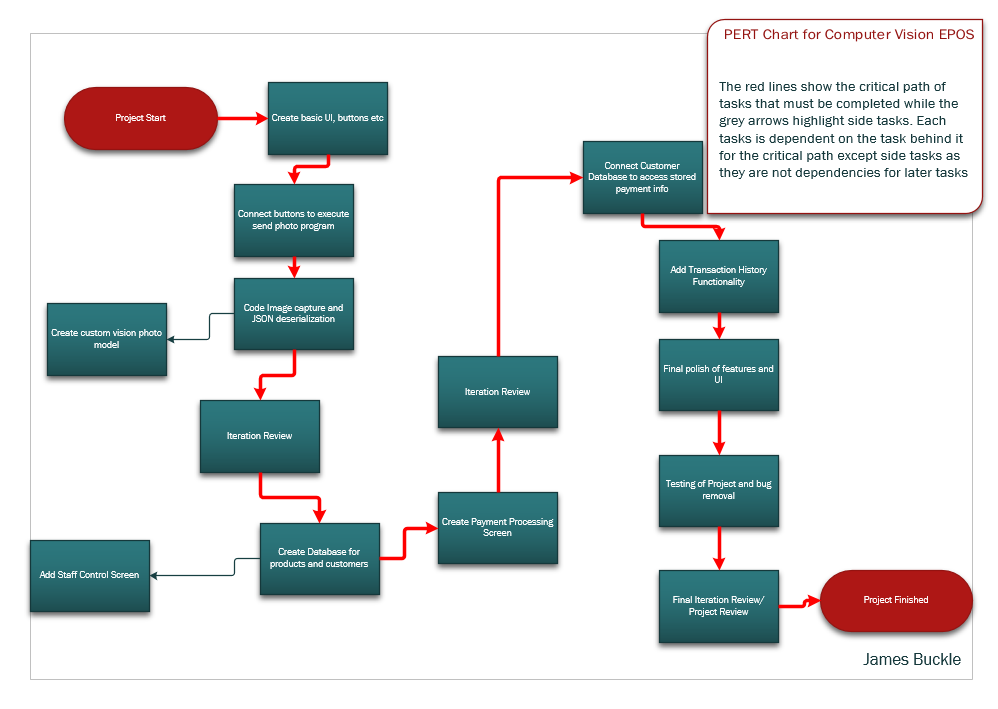


Figure A screenshot from my PERT chart highlighting the critical flow for my project

I have decomposed my project into individual iterations as part of the AGILE methodology

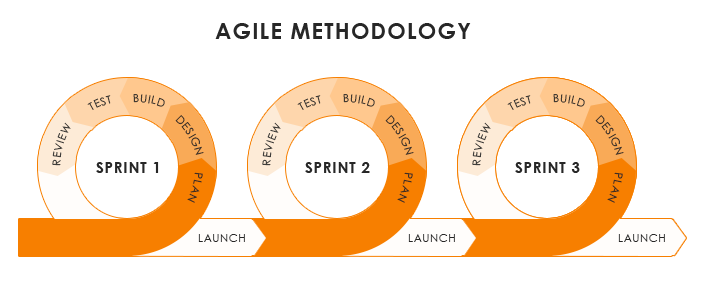


Figure A visualisation of AGILE source:https://www.csolsinc.com/blog/agile-development-in-laboratory-informatics/

Each iteration has a defined set of targets that I will aim to meet in each sprint and then based on the review at the end of the sprint the next sprints target can be adjusted to ensure the solution is built to the best standard in the time frame while also ensuring time is not wasted on unnecessary features. These iterations can be seen on the [GANTT chart](file:///\\eastnorfolk.ac.uk\Students\Home\start2020\57073\Computer%20Science%20Coursework\GANTT%20CHART.xlsx). I will also elaborate on specific targets at the beginning of a sprint as if I achieve a sprint quickly, I will bring forward targets from a later sprint.

# Testing

As my project largely depends on images, I will create test objects to use throughout the project using imitations of common items that might be expected in the final solution. I will test the solution extensively in a range of lights/ angle and other environments to ensure reliability and this will also be reflected by the photos used in training the custom vision model. I will also present the solution with unknown/ unintentional items and in my code, I will attempt to create a fallback that rejects a scan when the probability is too low. If not, I will ensure that it is easy to remove erroneously scanned objects from the solution. I have created tests for each iteration to verify basic features then a more comprehensive set of tests for my final solution to act as integration testing, ensuring all parts of my project work together (see test plan).

Other test data will be false/ erroneous card or user information, I will include [input masks](https://docs.microsoft.com/en-us/office/vba/api/access.textbox.inputmask) to prevent a number in the wrong format from being entered and in the SQL queries I will create a system to show an error if the customer number entered is invalid.

My main tests are detailed here in [my test plan](file:///\\eastnorfolk.ac.uk\Students\Home\start2020\57073\Computer%20Science%20Coursework\Project%20Testing%20Plans.xlsx). These are specific tests that I will perform to ensure that key functionality set out in my GANTT chart is contained within my solution.

# Key Variables

|  |  |  |
| --- | --- | --- |
| Variable | Type | Explanation |
| Imagefile | String | This will be the location of the image after it has been captured and saved. It will be passed into the program which sends the photo to customvision.ai so that it can be identified. |
| Ismotion | Boolean | This will be toggled when motion is detected and then used to trigger other scripts such as capturing an image.  (this is simulated with a button). |
| Objecttag | String | This will be returned by customvision.ai as the identified object. An SQL query will reference this with a database to identify what the object is and its price. |
| Probability | Double | This is how confident the program is in the guess it has made. This will be used to detect errors |
| cameraconnected | Boolean | This may be used as a self-test to ensure that a camera is connected to the program for it to work. |
| Totalprice | Real (2 dp) | This price will be the total of all items scanned. |
| Paid | Boolean | When set to true this will reset the program so that another user can use it. |

# Iteration Review 1

The first Iteration for my project was designed to build the fundamental code and layout for my project so that further Iterations can build on this to add additional features. While a large amount of this iteration was achieved, some areas were not fully met and 1 week of contingency time had to be utilised. At first I got distracted in the initial task of my project and created a debug menu to handle the camera connection

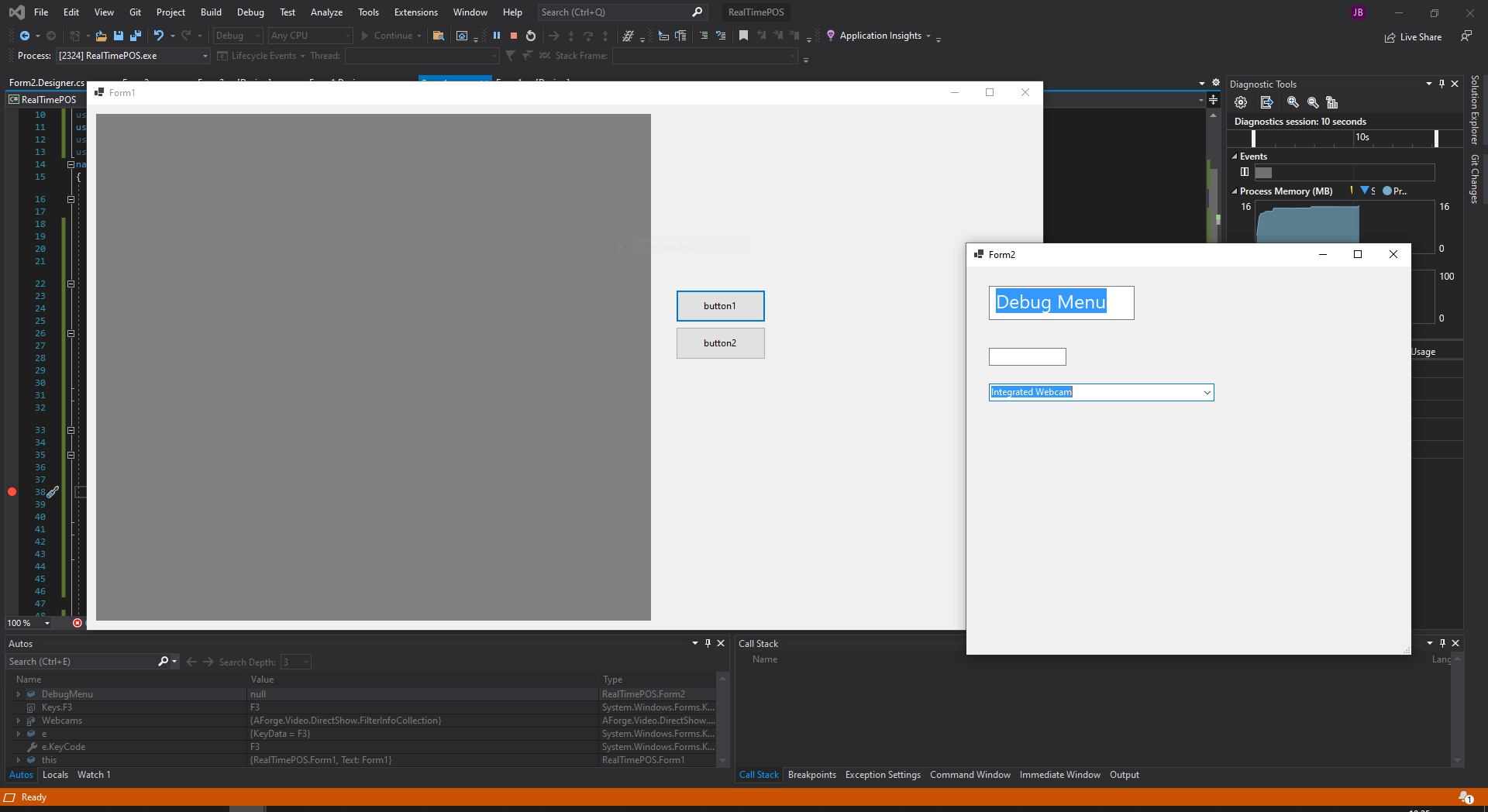


Figure Debug menu for connecting to camera

After getting stuck on creating this menu I then got myself back on task and following the agile methodology I got the first phase to a “good enough” state quickly to get myself back onto task. I utilised a separate program called SendCustomVisionRequest that I created from the Microsoft documentation to send the image captured by my program. I had to heavily modify this to work with my program and this was one of the biggest challenges presented by this phase. Passing the response from the program back into my code was not achieved easily and I resorted to storing the results in a text file and then reading this text file in the other program. This allowed me to get the Custom Vision client to identify my objects.

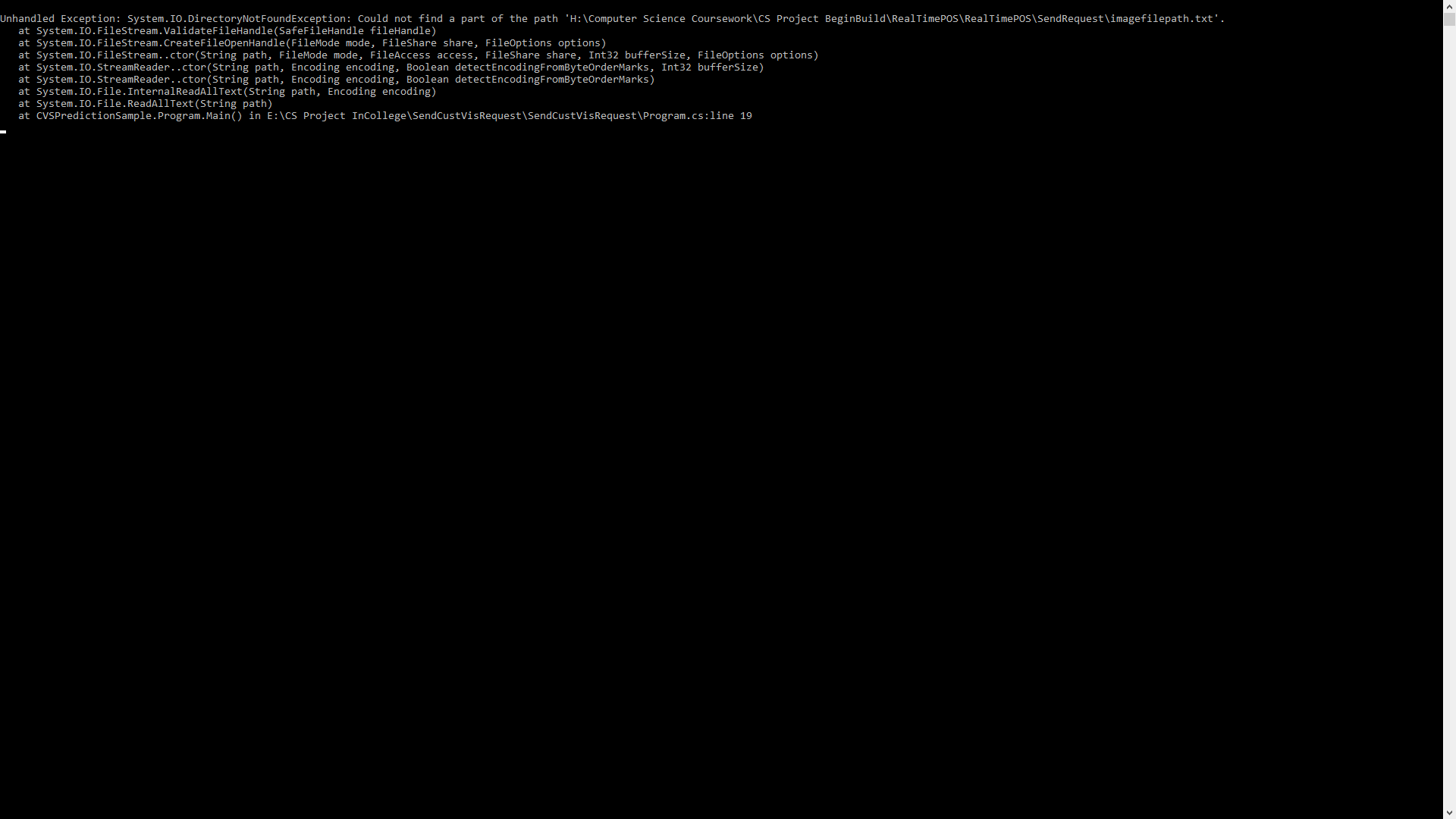


Figure Unable to find text file for response (cropped to fit page)

This method had its challenges, with having to create file paths that would work with machine independence so that my solution can be delivered on different pc’s without significant modifications to the program.

Another issue I came across in my project was passing the camera from the Debug Menu to the main screen. I wasn’t sure which data type the camera had to be passed as due to it being from a 3rd party library (Aforge3).

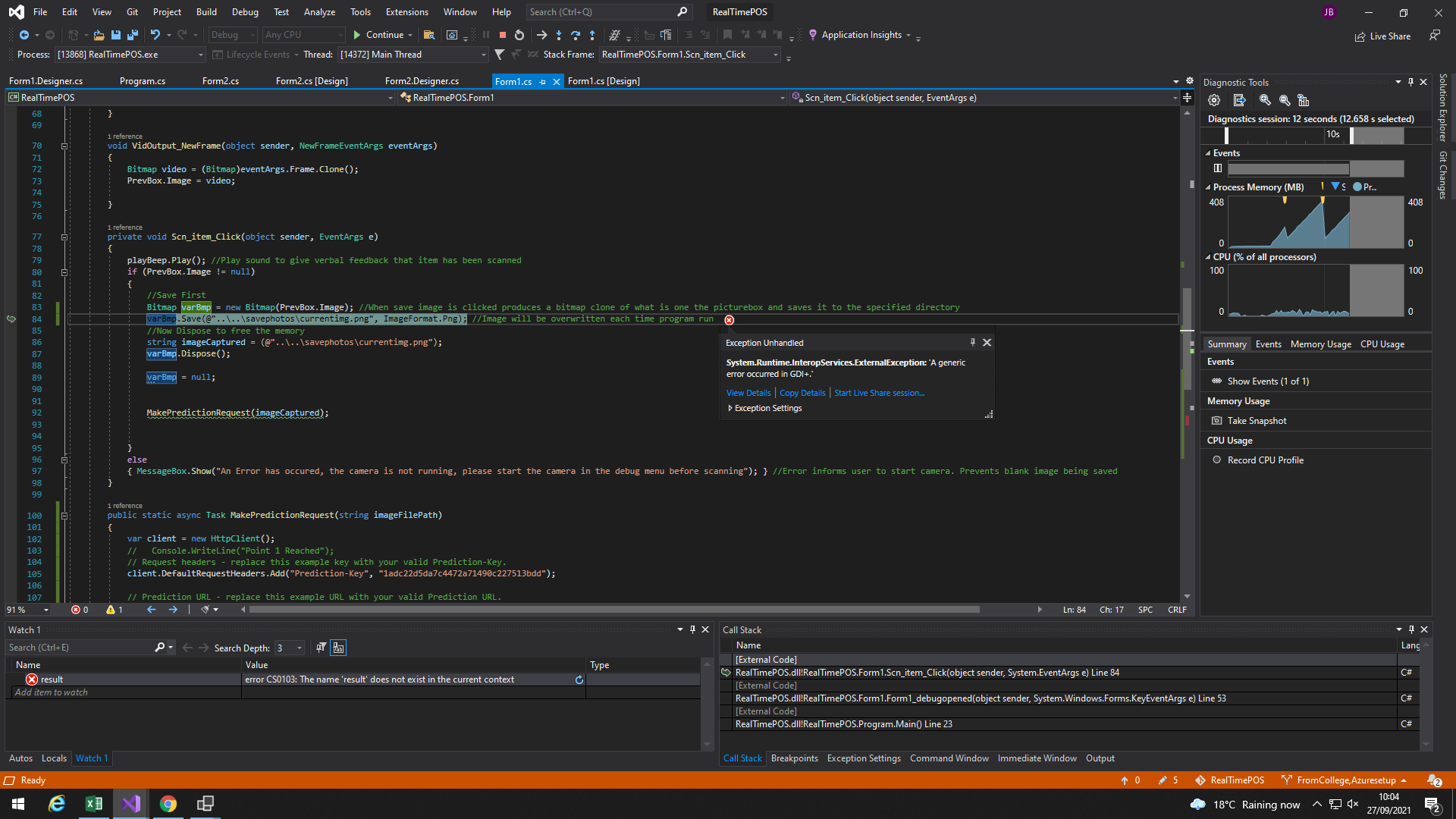


Figure A Generic error thrown when trying to use the camera

Some errors were very unclear and made it harder for me to debug my solution, spurious errors would send me off in the wrong direction, somewhat delaying my project

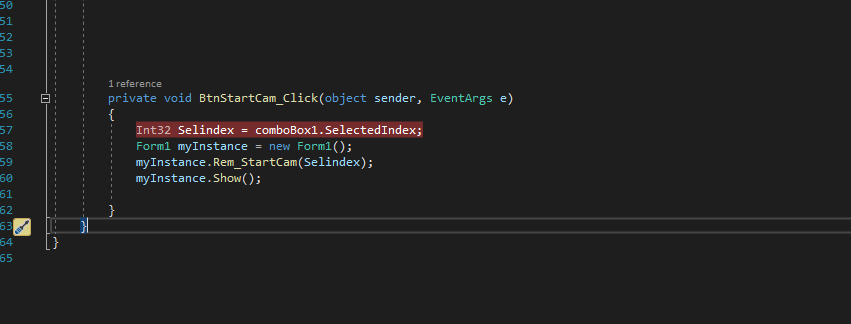


Figure Passing the camera as in INT32 data type

In the end I found that I needed to use the INT32 data type for the selected camera in order to get it to work and once I connected this to the picture box in my script the camera worked as it was expected to.

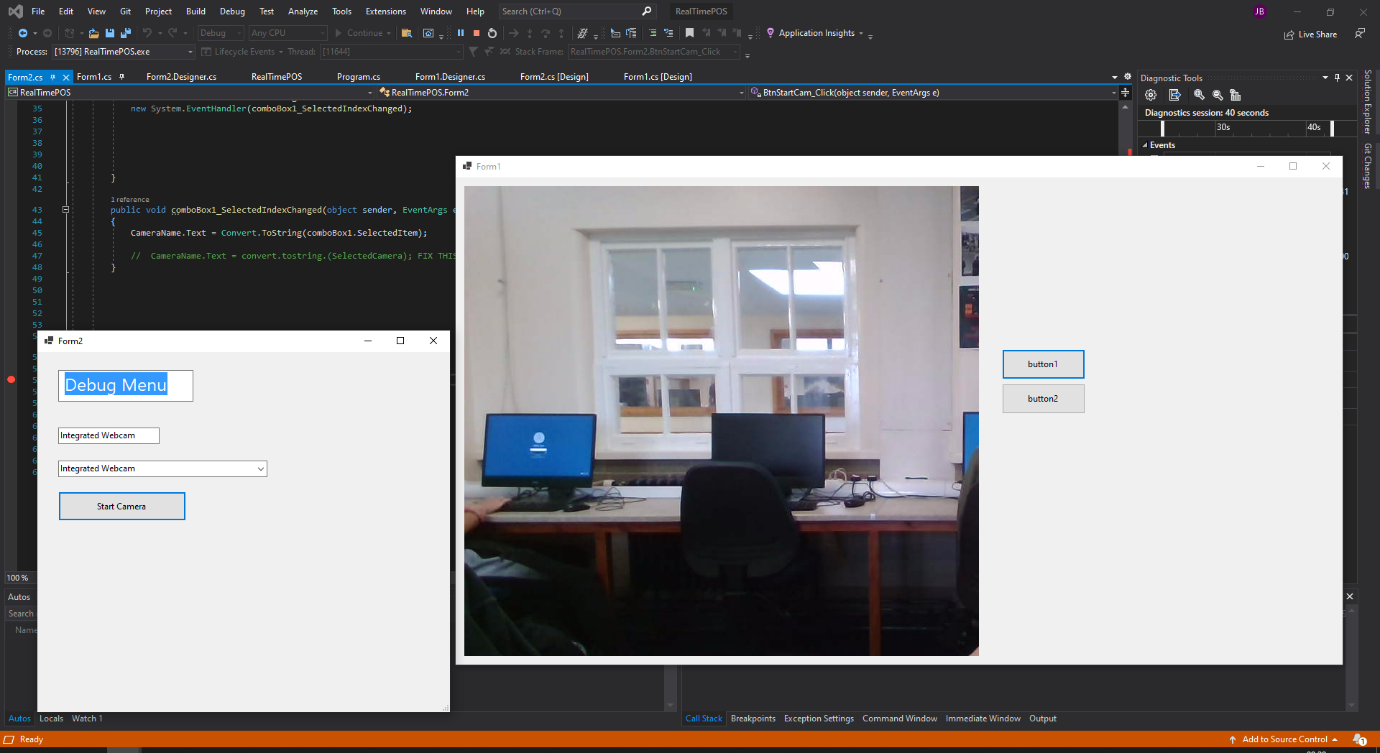


Figure Camera working as expected

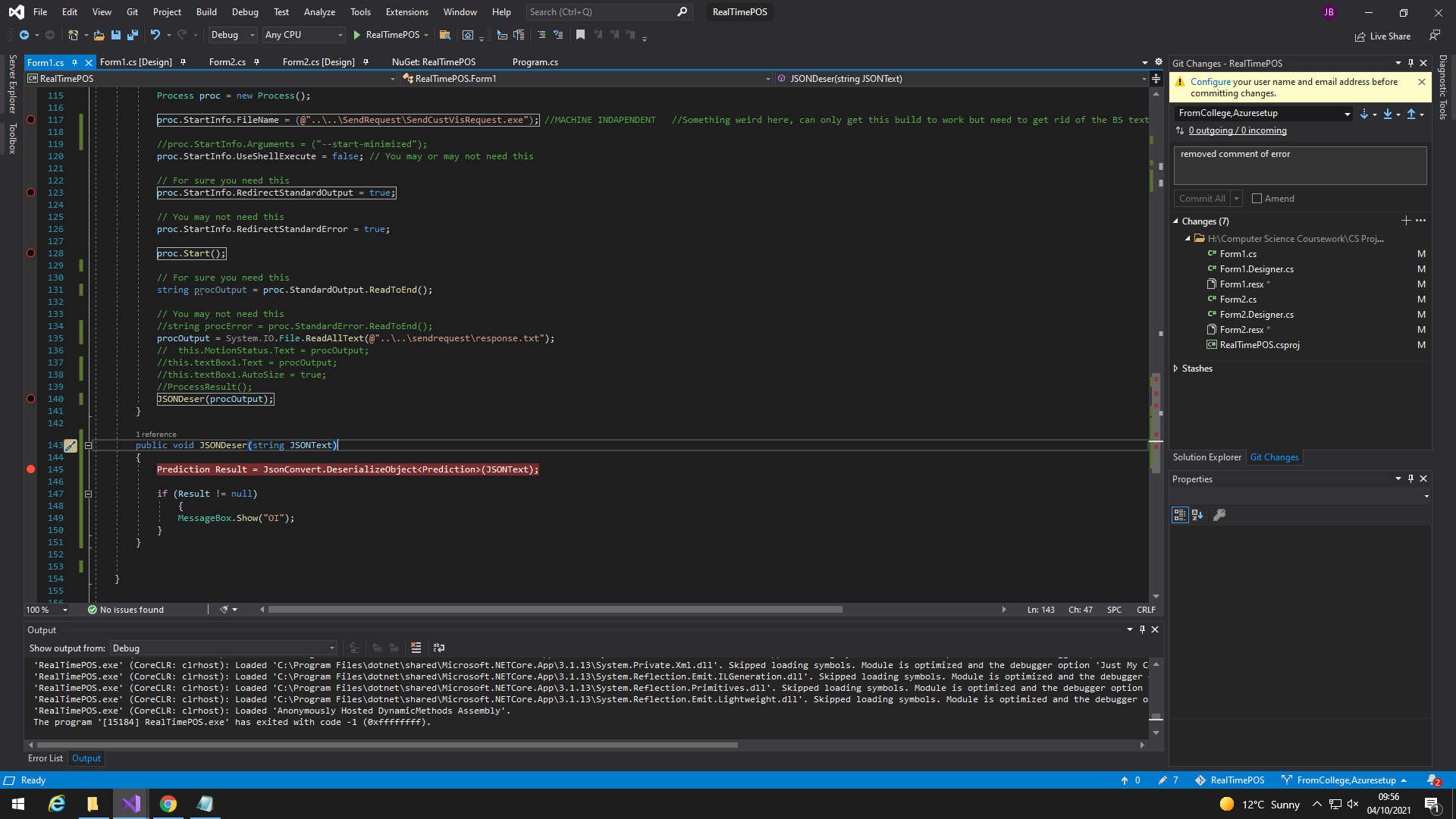
By this phase I was 1 week behind schedule and now had to attempt the most daunting task of this phase. The responses from the REST API Custom Vision Client were sent as JSON text including a JArray. I had no experience with JSON before and its arrangement made it very daunting.

Figure First attempt using JSONConvert.DeserializeObkect

In my first attempts I was not aware of how to properly extract the JArray from within the JSON. In the end I read the position of the start and end characters of the array ( [ , ] ) and subtracted any text either side of them. This is likely not the most effective method for reading the JArray but it was functional and allowed me to access the array.

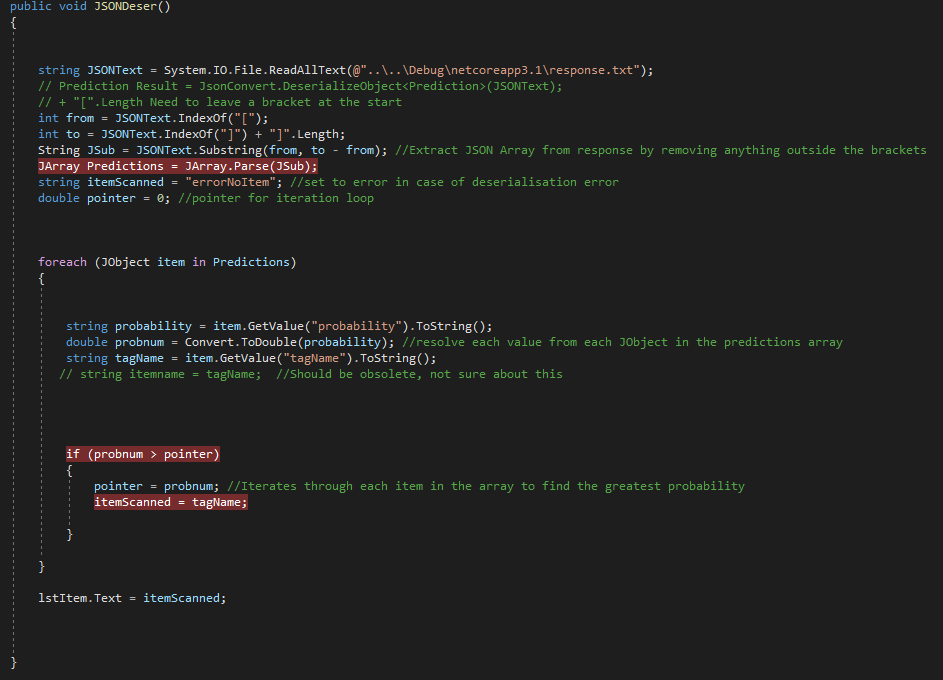


Figure The JSONDeser method used to access the array

I used a foreach loop with an if argument to iterate through the array and identify the tagName with the greatest probability from the array.

## Testing Iteration 1

For testing Iteration 1 my focus was to ensure there was basic error catching so that during further development of my project I do not encounter errors from errors in the fundamental code.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | | Test Type | Test Details | Expected Outcome |
| Iteration 1 Testing | | Normal | | Attempt to capture an image and correctly identify it (tests buttons and custom vision model) | Image should in most cases be correctly identified |
| Extreme | | Send 5 photos in quick succession. | Program may slow but should remain functional and recover quickly |
| Erroneous | | Capture a photo of nothing in the field | Negative tag should be identified, and nothing should be added to the basket |
| Erroneous | | Attempt to capture a photo without a camera connected to the computer | An error message should show, requesting a staff member and displaying a relevant error |

I conducted these tests and for this phase and all of them bar one was passed successfully. I have detailed the results below

### Normal Function Test

For this test I verified the basic parts of my project were working. Using my iteration loop for JSON, camera setup and basic UI. This test was passed successfully. When the camera was presented with a bag of mint imperials it was able to identify it correctly with over 90% certainty without any issues

### Extreme Excess Items Test

This test was an issue for my project. Due to the dependency on the separate SendCustVisRequest program the project cannot send photos in an extremely fast rate however, at the fastest rate possible the program was able to identify each image correctly in a quick manner whilst remaining responsive. Therefore, although not achieved in an ideal manner this test was overall passable.

### Camera Error Testing

These tests were designed to ensure the camera was connected reliably and that proper errors were shown. When attempting to capture an image with know active camera a message box informs the user of the error as expected and with a negative image, i.e. a person’s hand or a background image the project returns the tagName HandsandNegative as expected

# Iteration Review 2

This iteration was the most technical part of my project, I had to build the database elements and connect them to my program, creating complex SQL queries as well as develop my abilities in passing variable across forms.

The biggest issue I faced in this phase were the underlying issues I couldn’t resolve in iteration 1. Particularly, the use of an external program to identify the image. This was very unreliable as sometimes it would read the text file before the image had been identified, causing it to read the same response twice.

FIGURE REMOVED TO ANONYMISE INFO

Figure showing the identification from customvision.ai executing in the main program using asynchronous execution.

To solve this, I read into using asynchronous execution, this enabled me to identify the image in the main script as the rest of the program would await the response being generated. Although this should have been attempted in iteration 1, achieving this was significant progress for my project as it enabled it to run much faster and result in a much more elegant solution.

As mentioned, most of this iteration’s difficulty came from the databases required for the project. I developed the actual databases with relative ease as they were just created as 3 flat file databases, one for customer history, one for product information and one for the customer database used to process saved card information.

Text

Description automatically generated

Figure OleDB not registered on the local machine

The first issues presented when attempting to connect my program to the database was the fact that the OleDB package was not installed on my virtual machine. Although this is typically included in windows distributions, my installation did not have it. Fortunately as I had moved my project to a virtual machine I was able to install the package from the Microsoft website and correct this error. Developing in a virtual machine was a significant help to me when creating my project as I had administrator permissions without any group policies that would be applied on my centres network. As such I could install custom programs etc such as OleDB without issue.

Text

Description automatically generatedThe first query I created was one which would return the items name and price from the database. By reading the variable item from the JSON response it then gets the items price and FriendlyName which is the user friendly name (featuring spaces mainly). The program then adds the price of the item to the total price and calls the DisplayScan method to display the price and name of the individual item.

Figure SQL query to get item price from the database

Once I had figured out this query, the queried for the customer info database was relatively easy and allowed customer payment information to be retrieved from the database.

Text

Description automatically generated

Figure Insert into price history DB

The final command I created was one to insert the transaction history into a database. This was somewhat more challenging as I had not created Insert queries in OleDB before however with some experimentation this was completed. My biggest mistake when creating this was putting the variables in the wrong order causing the name to be inserted into the email column and vis versa however this was easily corrected.

A screenshot of a computer

Description automatically generated with medium confidence

Figure rudimentary email validation

Using a guide from Stack Overflow I implement email validation onto my payment processing screen. It attempts to put the text entered into a system.net.mail.MailAddress data type using a try, if this fails it indicates the email is likely the email is invalid and tells the user to request a member of staff

## Testing Iteration 2

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration 2 Testing | Normal | Enter payment details within required format | Payment details should be captured correctly |
| Normal | Use Staff screen to modify database (add demo product) | Product should be added correctly (will not work with custom vision model unless photos added) |
| Erroneous | Enter incorrect payment details (incorrect format) | Error message should be shown to alert that details are not of required format |
| Erroneous | Attempt to create items in the database with incomplete and false information | Formatting should prevent entry and instead specify expected format |

The testing for this phase was designed to test the quality of the databases I had created within this iteration. For the 1st test I wanted to ensure my payment details processing was functional and this test was overall successful, all details were captured in the correct format and input into variables which could be processed if the solution were connected to a payment processor.

My second test was not possible, modifying the database from the staff screen was not in the scope for this phase and as such I had not implemented it, therefore this test was failed as the required function was not implemented.

For my 3rd test, I entered data of the wrong type, format and length into the text boxes, due to my email validation method and input masks created within visual studio this test was successful, in nearly all cases the incorrect details could not be entered due to formatting errors. The only exception was for emails which would show a warning message when incorrect data was entered.

Finally, my 4th test was again not fully possible. Inputting items into the database was not implemented into the staff screen. However for items entered into the database from the table view with incomplete data the “required” property of all columns made this impossible so this test was partially successful as some features were present.

# Iteration Review 3/4

Due to my success in completing the tasks specified in iteration 3 I decided to incorporate the scope of iteration 4 of the project within this iteration in order to maximise my productivity and ensure the project is developed on time. The only tasks scheduled for iteration 3 were to connect the customer database to the payment details screen, allowing for users to pay using their customer pin instead of having to manually enter their information every time they wish to pay, and to add the ability to store past transactions in the transaction history which was done using a simple insert SQL query. As I already had experience using SQL in this project for getting an items price I reused large elements of this code with small adaptations in order to complete the necessary tasks

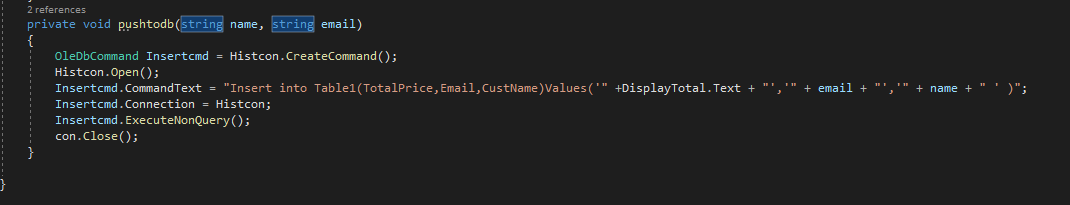


Figure my function to insert the payment details into the price history database HistDB

The main challenge I faced was in redesigning the unattractive default look of winforms as set out in iteration 4 of my project. The classic look of winforms is very dated and unpleasant. It does not give my project the modern and fresh look I wanted to create.

Graphical user interface, application

Description automatically generated

Figure A default windows form

Unfortunately, there are limited options to modernise a windows form as most styling resources are based on Windows Presentation Foundation and I did not have time to merge my project over to this. Instead I decided to modernise my project by restyling within the designer in visual studio, I utilised a new [colour palette](RealTimePOS%20Colour%20Palette.pdf) to give a theme for the project. This was a drastic improvement to the basic layout and although it did not give the refined feel I wanted to give my project I am happy within the outcome given the tools I had available.

Graphical user interface

Description automatically generated

Figure My main form after being recoloured

My project looks suitably well finished and I am overall happy with the outcome of these phases.

## Testing Iteration 3

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration 3 Testing | Normal | Complete a transaction using a stored customer number instead of payment details | Should query database for card info and combine with entered CVV |
| Erroneous | Input an invalid customer number | Error message that the number is invalid. |
| Extreme | Enter an invalid customer number 5 times consecutively | Program may slow but should not crash and display a relevant error message |
| Normal | Ensure transactions are correctly stored in the database and can be recalled by the staff screen. | Transaction history should be queriable and stored in an access database. |

For my 1st test I wanted to ensure that querying the database for payment info worked as expected and this test was successful. My solution did not require the CVV to be entered separately but is instead stored in the database. Despite this minor deviation from the expected outcome this test was overall successful.

For my second test I wanted to ensure that only valid customer numbers can be entered in my solution and that a clear error message is shown when an invalid number is entered. This test failed as I have not at time of writing implemented this function into my solution. I will aim to implement this feature within some of the contingency time I have available to ensure that this test is passed before my final project testing.

Test number 3 was not needed as it was dependent on test 2 passing, this test will also be repeated after the feature is implemented.

Test number 4 was a partial success. I did not implement the ability to access the database from the staff screen as I believe it will cause issues with document locking in access and present great challenges. Instead the staff can access the transaction history by opening the database and manually reading it.

Overall the testing for iteration 3 has failed and I will now take steps to correct the failures within this iteration.

## Correcting Failures in Iteration 3 Testing

Text

Description automatically generated

In order to mitigate the failure of test 3 I implemented an if loop. By default the variable CustName is set to “NULL” and so by checking if after the query it is still null it ensures payment is only completed if a valid customer number has returned a valid customer name.

For test 4 I quickly clicked the button in rapid succession however due to the presence of the messagebox.show argument it prevents the button from being pressed excessively so overall this testing phase following this correction can be considered successful.

Following these corrections, my project now meets the requirements I set out with iteration 3

## Testing Iteration 4

For Iteration 4 I had no formal testing plans as the goal of this iteration was to polish the UI of my project and fix any lingering errors. Therefore this phase can be regarded as a success as I recoloured and adjusted the UI to run in full screen.

# Final project evaluation

Overall I have thoroughly enjoyed the development process for my project. I have faced many challenges due to the various functions I aimed to include such as the database, camera script and communication over the REST API however the program developed meets the brief that I specified and complies with the tests I set out for myself to a good standard. My code is robust and uses logical names for functions as well as comments to clarify anything that might be unclear. When my program is compiled to an executable it is portable to any x86 windows machine from windows XP onwards. This means that the program can be easily installed for the end user without too much difficulty. I am very pleased with the quality of my code and the configuration of the various functions I included. Variables are passed elegantly and with concise names. By incorporating the custom vision script into a C# Task it also makes the project much faster than I initially expected.

My object recognition model developed in customvision.ai achieved a much higher degree of accuracy than I expected to achieve given my limited sample size of just around 1000 images.

Graphical user interface, application

Description automatically generatedThe challenge with my custom vision model was creating a vast selection of negative images to be used when a hand or other non-item is passed in front of the scanner. This was somewhat reliably achieved as when a hand was passed in front of the scanner it would return the negative tag and display an error message however in a limited range of instances it would instead falsely identify an item. To avoid this, I would ensure the camera is placed in fix position where it is unlikely to scan any unexpected option and I have also configured a feature to reset the program in the event of an error.

For the end user (supermarkets and retailers) my solution has a significant cost saving, making till systems much faster to scan. It would also be cheap for the end user to purchase as it is an off the shelf solution which means the cost can be spread across the many customers who purchase it rather than having an expensive bespoke solution developed.

To achieve my development, I stuck well to the AGILE methodology. I consistently focused on the end goal for each of my iterations to ensure I did not get distracted on unnecessary features or functions. My iterations allowed me to refer to my charts and pseudocode in order to evaluate how my project was doing and if I needed I utilised contingency time periods ton correct any mistakes or omissions.

## Overview of usability

In order to test the usability of my solution I had it peer reviewed by a volunteer (A). My volunteer has some technical knowledge but no previous knowledge into how to use my solution. This gives me insight into how usable my product is. I will at first have him attempt to use the product with no explanation and then I will provide a small amount of guidance if necessary to evaluate how effective my solution is as it should be readily usable as a self-checkout to anyone without specific technical ability.