2020

Edge Hill University

The Department of Computer Science

Development of systems to sort, recognise and sell trading cards

CIS3140 – Research and Development Project Report

SWAGAT KUMAR

STEVEN RYAN-CARPENTER | 23693703

13/05/2020

This Report is submitted in partial fulfilment of the requirements for the BSc Honours Computing Degree at Edge Hill University.

Abstract

With the rise of the online marketplace, automation has become an increasingly important aspect of running a business. While many large industries have access to these tools, some smaller industries do not. If provided, these tools could provide online retailers the tools to minimise the costs required to list and sell these cards online. One area of note is the trading card game industry. Trading cards have many unique names, sets and values and retailers of this industry have extremely high volumes to sell. Without automation, this increases costs through either the need of extra staff to sort and sell these cards, or spending time that could have been spent performing other tasks. Additionally, individuals themselves have large trading card collections and are interested in the selling and sorting of these cards. As such, this paper will examine and discuss the available tools present in the assistance of trading card selling as well as explain the development of a system used for the automation of trading card sorting and selling. The systems developed during this project allow individuals or businesses to automate the selling of these cards to online retailers.

Acknowledgements

I would like to thank Swagat Kumar for the assistance in the development and design of the project as well as for the advice on my project report. I would also like to thank Ella Pereira for the detailed advice given on this report. Without her help this report would not be written to the standard that it is currently at. Additionally, without the support of Collette Gavan I could not have done as well as I have in my university work and modules.

Table of Contents

Table of Contents

List of Tables and Figures

Chapter 1 – Introduction

1.1 Background

Online marketplaces are an important field for both businesses and domestic individuals. Trading cards have been exceedingly difficult to sell this way due to their vast quantity and variety. This document is a project proposal for the development of systems to physically and digitally sort and enter cards from popular trading card games, primarily Magic: The Gathering (Wizards of the Coast, 2020). It will explain and discuss information regarding the project such as the project background, selected literature review, aims and objectives of the project, methodologies and development methods, resources required and ethical considerations. The project will be created using robotic tools provided within the Tech Hub (Edge Hill University, 2020a) of Edge Hill University (2020b). Cards will be scanned using either a camera integrated into these robots, or an external device. These cards, once scanned, will be entered into a database or spreadsheet, while also being sorted physically based on predefined categories. These cards will then be able to be listed online on marketplaces such as eBay (2020a).

1.2 Project Aim and Objectives

1.2.1 Aim

The project aims to utilise robotics and computer imaging to organise unsorted trading cards into appropriate categories. The scanned cards will be entered into a database, where the user has the option to list these cards on an online marketplace such as eBay. Features include online listing where the system automatically registers the name of the card as well as the price of the card.

1.2.2 Objectives

- To analyse existing systems that serve the same purpose as the project to identify their methods and limitations. This will be done through background analysis and literature review.
- To gather information regarding requirements of the target demographic. This will be done through the use of questionnaires and interviews.
- To design a system appropriate to serve the requirements of the target demographic. This will be done using appropriate modelling methods such as Use-Case diagrams.
- To develop a system according to the design. This will be done using programming languages such as Python (2020) and Java (2020), and robots such as the Dobot M1 (2020a) and Dobot Magician (2020b).
- To test the developed system for issues and errors. This will be done using methods such as white-box testing (Nidhra and Dondeti, 2012) and User Acceptance Testing (Hambling and Van Goethem, 2013).

1.3 Methods and methodology

For the project, methodologies were analysed and compared against each other. When this analysis had concluded, it was originally determined that waterfall methodology was most suitable for this project. Waterfall was originally chosen due to the lack of development in the field of trading cards and trading card sorting, and the static and consistent nature of this field. However, despite this, flexibility issues were found when developing under Waterfall. As such, the Incremental model was chosen instead. The Incremental model allows much more flexibility in responding to information gathered in analysis. This allows the development of the project to continue despite information not yet being gathered, and for the process to respond to difficulties and challenges encountered during development. As both methodologies were at some point intended to be used during development, both are discussed below.

1.3.1 Waterfall methodology

Waterfall methodology is a process in which development is processed through stages. These stages are different with each version of the methodology, which is dependent on the project being produced (Balaji and Murugaiyan, 2012). For the project, as it is a single project that will not receive maintenance or updates, the development will only contain the essentials of the waterfall methodology. For the project, the steps will be Analysis, Specification, Design, Implementation and Testing.

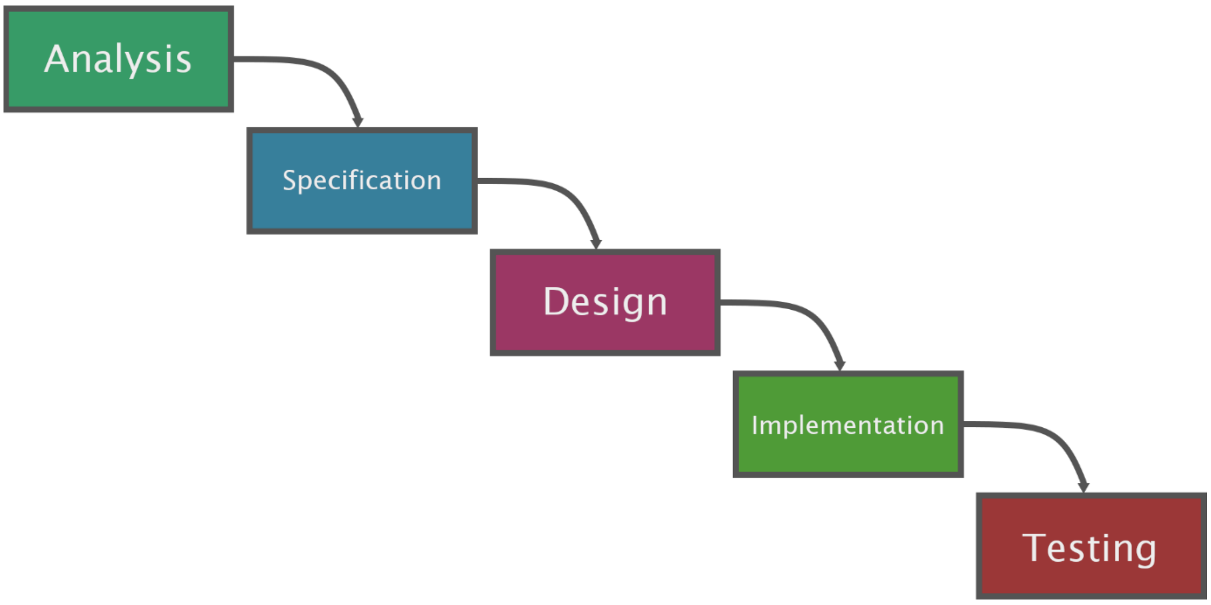


Figure 1 Waterfall Methodology

The analysis stage will consist of a literature review as well as survey methods such as questionnaires for trading card players and interviews for LGS owners and managers. This literature review and these interviews will be used for the purpose of creating the specification. The specification will consist of requirements for the project, consisting of end user requirements necessary for the project to be useful to its target audience. The analysis and specification are important for the design, as without these being gathered the project runs the risk of not being suitable for the intended demographic once fully developed. This would result in wasted development time, and as such should be avoided. Succeeding the analysis and specification is the design. The design phase uses the information gathered in the previous phases and creates a project plan based on this information. It is used to finalise ideas for implementation, as to create a more comprehensive and precise project plan. It eliminates indefinite concepts and allows for the implementation to process smoothly without the worry of features being overlooked.

Implementation will be the bulk of the project. This phase refers to the creation of the project. If the design phase has been successful, each step of the implementation will be completed according to the design plan. However, it is possible that during this phase later information may be acquired and the design will have to change, possibly requiring the return to the design phase. Once the implementation is complete, testing begins. Testing can consist of user trials or in-house testing. User trials allow for the project to be tested in the field it is intended for, and as such are very useful at finding errors made during development and user interface issues. In-house testing refers to a tester, which may be a third party or the developer themselves, using the project with the purpose of identifying possible errors and issues within the project. These steps are vital to ensure that the project is functional. If any errors or issues are found, as is likely, development would usually re-enter the implementation phase and begin the production of fixes and features to solve these problems. However, due to the limited scope of the project, this is unlikely to be possible.

Advantages of waterfall is that the project is the ease of planning and documentation as a developer. As the stages of waterfall succeed one after each other, it is simple to see where progress is by analysing the stage the project is that within the Waterfall. This helps prevent confusion and assists in keeping projects on track and timely.

Disadvantages of waterfall are many. The lack of flexibility does not suit itself well to a changing market, and the plan does not suitably make use of a team's time. Tasks in a modern workplace are often delegated based on individual specialty. In this environment, under waterfall methodology while the analysis is being performed, specification cannot be started. This is true for the implementation stage while the specification is being performed and continues down the stages. As such, it has found to be unsuitable for these modern working environments and has been abandoned in recent years for more suitable approaches such as agile (McConnell, 1999).

1.3.2 Incremental model

The Incremental model is an adaptation of the Waterfall methodology used for projects in which the initial software requirements have been previously established to a reasonable degree, but the development cycle of the project is not suitable for the purely linear process of Waterfall. In an Incremental model, software is completed in stages of development, returning to the beginning of the waterfall as stages are completed. This approach allows development cycles to focus on one feature and allows information to be gathered progressively as development continues. This approach has many benefits, such as the ability to, assuming an increment has been completed, deliver a product by the deadline even when the entire product has not been finished. Additionally, the model allows information gained during the development process to be used. As development may reveal challenges or opportunities, it may be necessary to change the approach of the project dependent on these newly revealed events (Pressman and Maxim, 2014).

It was discovered during development that the Waterfall methodology that was originally used would not be suitable for the project. Due to this discovery, the Iterative development method was chosen instead. This approach was deemed much more useful for the needs of the project, as the Waterfall methodology was too restrictive on development time.

1.3.3 Literature Review

A literature review, as performed for the project, is a method of analysis that gathers information from relevant literature published. It is a comparison of this information and draws conclusions based on agreement or disagreement in this literature. It is a necessary part of any research project as analysis of this secondary information is key to the creation of a successful design.

1.3.4 Surveys

Surveys are an information gathering methodology that gathers data on individuals or groups relevant to the field being researched (Upton and Cook, 2008). Surveys are essential for gathering information on people's opinions or activities. Questionnaires are one form of survey. Questionnaires contain predetermined questions to be given en masse to individuals based on the objectives of the project. Information gathered this way can be analysed both quantitatively and qualitatively. Another form of survey to be performed in the project are interviews. Interviews are an information gathering process that consists of an interviewer asking the interviewee questions related to the project. They are primarily used when discussing complex subjects that a survey would not provide the depth required. They are appropriate for the project as certain detailed information pertinent to the task would be difficult to acquire using a questionnaire. As such, it was chosen to be used in this project.

1.4 Scope

The project consists of the creation of an application to sort trading cards and enter them into a file or database for information gathering. Additionally, the project will create software to take this entered data and sell it on online marketplace websites such as eBay. The project will focus on Magic: The Gathering cards first due to the availability of tools to develop with. If there is sufficient time remaining, additional features will be added. These include compatibility with other trading card games, and the creation of a graphical user interface.

1.5 Report Structure

The structure of this report is that it contains 6 chapters, including this introductory chapter. Then, additional chapters will follow the development of the project as it is completed, beginning with the background and literature review performed for the project, and ending with the conclusory chapter that includes critical evaluation of the process and choices made during development.

As stated, Chapter 2 of the report will detail a project background and give a literature review on works relevant to the project. The project background section details information such as the history of the project, the information publicly available regarding the project, and the path that lead to the decision to create the project. The portion of this chapter that details the literature review will be a detailed description and comparison of information already available regarding the tools and elements relevant to the project. It will critically analyse this information and provide positive and negative conclusions based on this information.

Chapter 3 of the report will analyse the requirements of the project and provide information regarding the design of the project. The chapter will detail primary information gathered during the project and use this information in the creation of the specification of the project.

Chapter 4 details updates on and events encountered during the creation and implementation of the project. It is updated as the project continues, serving as a progress update and documentation for the project's development history.

Chapter 5 is the last chapter regarding the development of the project. It includes information gathered through the testing of the project. This information includes testing tables on specific features, as well as whether those features passed these tests. It also evaluated the project based on its success at performing the tasks at which it was required. Additionally, it interprets data gathered and provides conclusions based on this data.

Finally, Chapter 6 provides a reflection on the project. It is a summary of the project and feedback will be given from project creators based on their feelings of how the project went. It will provide information regarding the positive and negative aspects of the project's development, as well as information on what could be done differently to improve the process of the project's development.

Chapter 2 – Background and Literature Review

2.1 Background

Online marketplaces are a very important area in the modern economic environment. In its last revenue report, eBay released that it made $2.6 Billion in revenue in Quarter 3 of 2019 (eBay, 2019b). This marketplace is being increasingly utilised by Local Game Stores (LGS), a term for shops that sell tabletop games such as trading cards games, miniature games and board games. These local game stores, such as Dice and Decks (2020) and Manaleak (2020) have expanded onto the online marketplace. These stores have thousands of cards listed and keeping a digital inventory of these cards is extremely time consuming. In addition, there are many home users who themselves have thousands of cards from years of play who would like to sort and sell their cards. Due to the extreme time consumption, many people cannot sort and sell these cards themselves. The project aims to solve this issue by use of automatic robotic card scanning, with the addition of functionality to list these cards on online marketplaces such as eBay. This will minimise the effort required for the sorting and selling of cards and as such will allow more shops and individuals to take advantage of the online marketplace. The central technology necessary for this project will be developed primarily through the means of computer imaging, also known as computer vision.

2.2 Literature Review

2.2.1 Pricing Prediction

It is possible but difficult to predict the market value of cards in trading card games. Zhang (2013) found that in the card game Fantasica the price of cards could be determined by the statistics of the cards within the game. This report documents the use of logistic regression to create a model as well as K-means clustering. Although these methods were generally accurate, they sometimes made very inaccurate predictions. The implementation of the separation of categorical and quantitative data allowed the data to become much more accurate. These findings can be extrapolated to other card markets in predicting the prices of cards within those marketplaces. Pawlicki, Polin and Zhang (2014) found in a research on the prediction of Magic the Gathering card prices. In this research they found that, using features and labels, they could successfully predict the prices of cards in the future. Their method of doing so first involved the same logistic regression method that was used in the previous method, but instead additionally supplemented with data from a support vector machine. This support vector machine instead lowered the accuracy of these results, with logistic regression remaining the dominant solution. Additionally, an AI based approach has been developed. Hiroki et al. (2019) performed research into AI based card price prediction methods by focusing not only on basic card information but also additional rules text found on the card. By utilising logistic regression, SVR, RFR and MLPR algorithms, it was determined that logical regression has the highest levels of anomalies in card pricing data. This is likely due to the fluctuations in price when considering cards that have just been released into the standard format. However, when considering SVR, RFR and MLPR algorithms, the have much higher levels of accuracy and as such are better at predicting card pricing data. This work has by far the highest level of accuracy out of research reviewed. This work also aligns with the findings in other papers on the same subject, but due to its higher number of data points analysed, provides a much more accurate analysis and prediction of card prices in the future than other more simplistic means.

2.2.2 Computer Vision

On the topic of computer vision, many relevant works have been written. Umbaugh (2005) states that "the importance of computer imaging is derived from the fact that our primary sense is our visual sense, and the information that can be conveyed in images has been known throughout the centuries to be extraordinary". Hornberg (2017) agrees with this, stating "the advantages of a machine vision system can be multiple, […] the sum of these benefits justifies the expenses for a vision system". Google have utilised a combination of AI based and keypoint image matching algorithms to recreate complex 3D structures. However, they also state that it is difficult to compare image matching algorithms due to the inability to create conclusive benchmarks when considering extremely large data types (2020). A report by Li et al. (2017) states that while feature matching is a very important tool for image matching, the current methods of performing this task have issues that can be solved through the consideration of consideration of photometric and geometric constraints. By utilising these methods, a precision of 98% to 100% could be achieved. This is in comparison to other, more commonly used feature matching algorithms that, in the same data set, could only achieve an accuracy of 76% to 87%. He et al. (2016) state that while neural network image processing has provided breakthroughs in the field of image recognition, they also struggle as the training process is more difficult. As such, this is not a perfect solution to image recognition problems. If more training data can be acquired, accuracy can be increased. But the time required to process this data also increases. Artificial image creation can also be used to create data sets for training purposes.

Computer vision is essential for the project's functionality regarding the analysis of cards from these games. Computer vision solutions have been created specific to Magic: The Gathering cards, such as MTG Card Reader (Fong, 2020). More open solutions which can be adapted for use have also been made, such as ImageAI (Olafenwa, 2020). The first of these solutions has a very high success rate, matching images to specific cards while these images are rotated to any angle or mostly obscured from the bottom side or top, upside down, and is accurate even when only a very small portion of the scanned card is captured (Maddenfong, 2018). Analysis of these methods will be vital in the development of the project, and it is possible that either or both will be used in development once research has been conducted and options have been explored.

2.2.3 Robotics

Robotics are another key part of this project. Robotics have been used in tandem with computer vision for a variety of purposes. Surgery robots have seen great success, with the rate of discharges raising as much as 60% with the introduction of these (Barbash and Giled, 2010). Ocado (2020) have utilised robotic sorting machines for the organisation of groceries, as well as three-dimensional "picking" machines that use image recognition to identify and sort specific items (Vincent, 2018). Lin et al. (2019) found that through the combination of computer vision and robotics, they achieved automatic sorting using 3D visual perception. This technology allows robots to interact with humans and, where errors occur, and guide humans to correct these errors. Siciliano and Khatib (2016) state that while robotics is an important and crucial field both to the industry and humanity, they are also dangerous and should be treated with caution. When improperly controlled, robotics can cause damage to the item they are working on and, in serious cases, the environment around them. If this environment has people within it, potential physical harm could come to them. It is for this reason that robotics must be treated with caution when working with it and when it has been distributed into the environment it is intended. Additionally, a speech recognition system to define rules for the robotics was found to be effective. These factors emphasise the importance of the integration of computer vision into robotic based systems. The combination of the two is necessary for the project.

In terms of previously developed hardware, there are no completed versions on the market to as of the 28th of February 2020. However, there are multiple works in development such as Roca Robotics (2020)'s prototype. This prototype uses a pneumatic suction-cup based design inside of a box and the company reports sorting of up to 500 cards per hour. They claim an accuracy of over 99% and sort cards into a CSV file due to its compatibility with e-commerce platforms. Out of the projects that could be found, this is the most advanced solution available.

Chapter 3 – Requirement Analysis and Design

3.1 Requirement Analysis

Gathering requirements is an important part of the project to ensure that the project fits the needs of the target demographic (Coventry, 2015). Performed correctly, a requirement analysis decreases the development time of the system as well as improving the quality and allowing the scope of the project to be more accurately and accessibly managed. When surveyed, 68% of businesses used poor requirement practices and as a result were completed on budget less than 20% of the time (Coventry, 2015). In the scope of this project, the requirement analysis is twofold: the target audience's needs on how the system should work, and the requirements of the project to function. These two features should not be mutually exclusive but are distinctly different. The target audience's needs involve the intended target demographic's preferences and requirements for the project to be suitable for purpose. The project's requirements involve the features necessary for the project to be finished. As stated, these requirements usually cannot be mutually exclusive as the needs of the user cannot be outweighed by the needs of the project (Watt, 2014). One exception to this is cost; if the cost of the project. If the requirement of the user would not be within the scope or budget of the project, the requirement cannot be reasonably met.

3.1.1 Survey Results

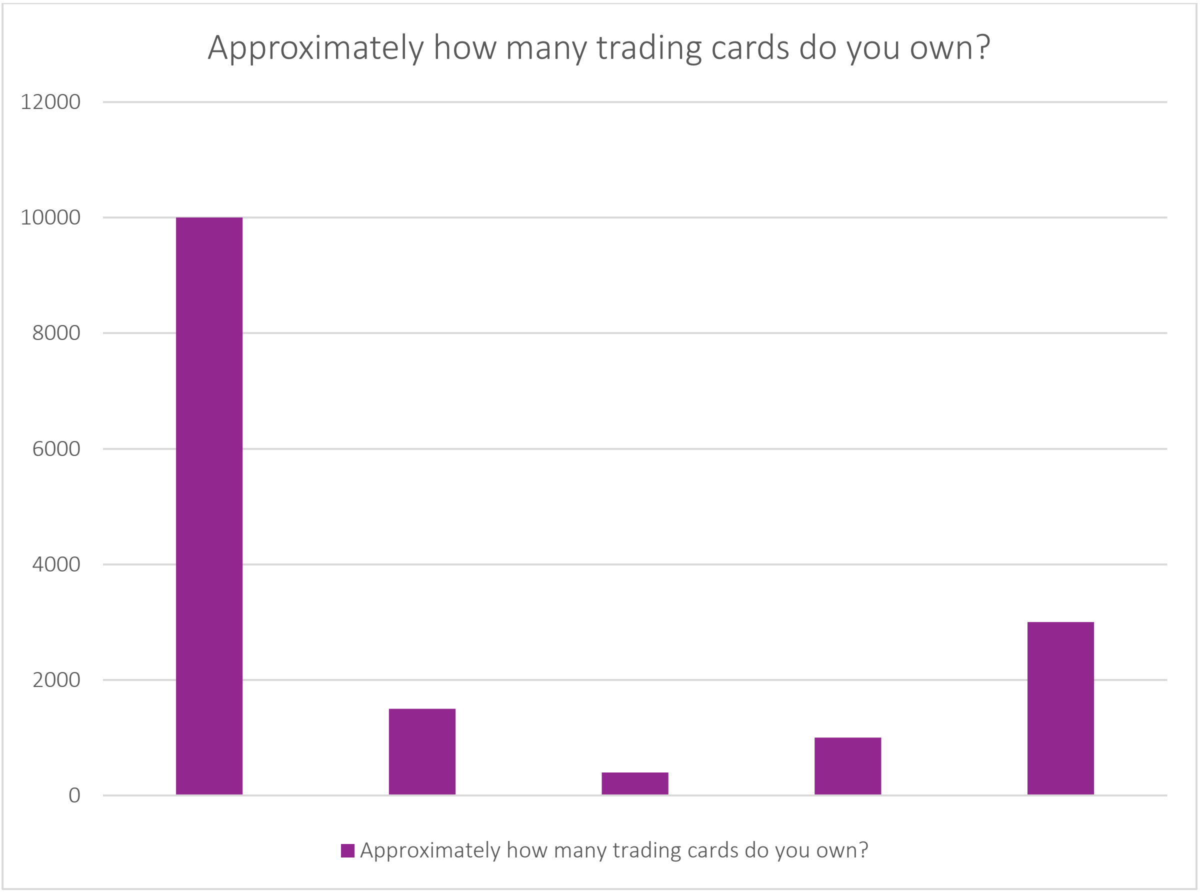


Figure 2

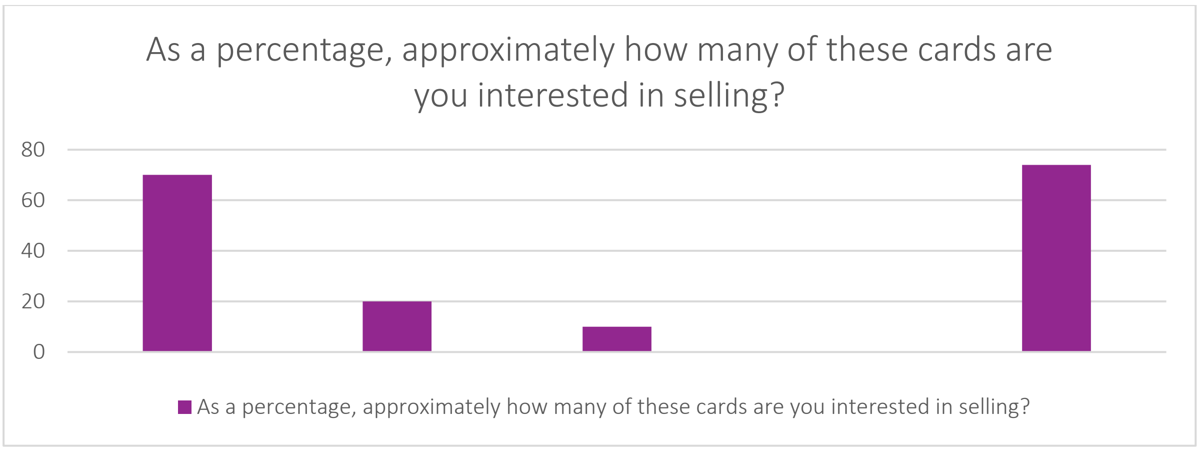


Figure 3

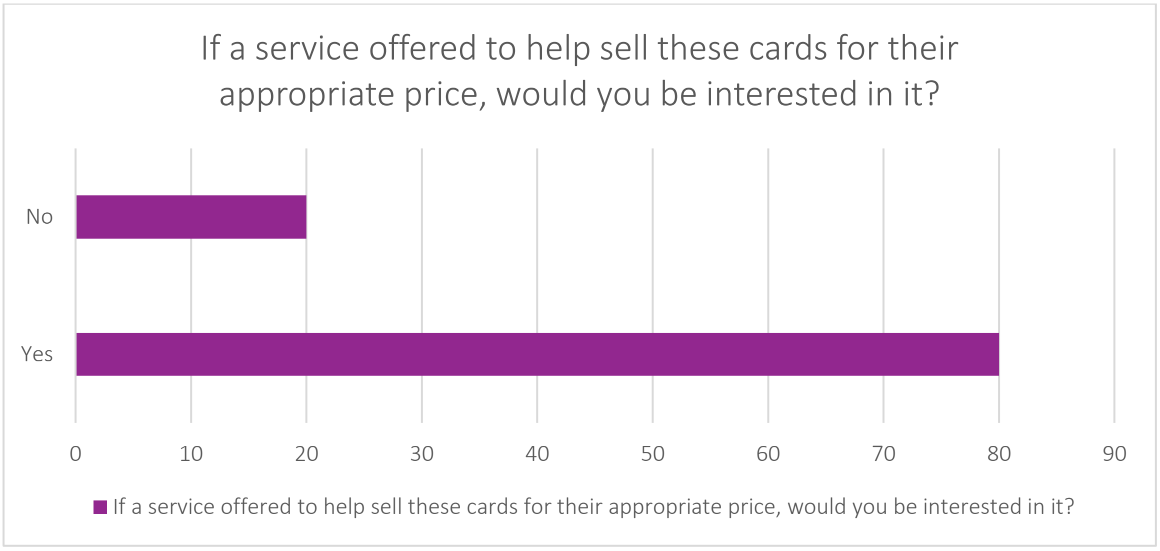
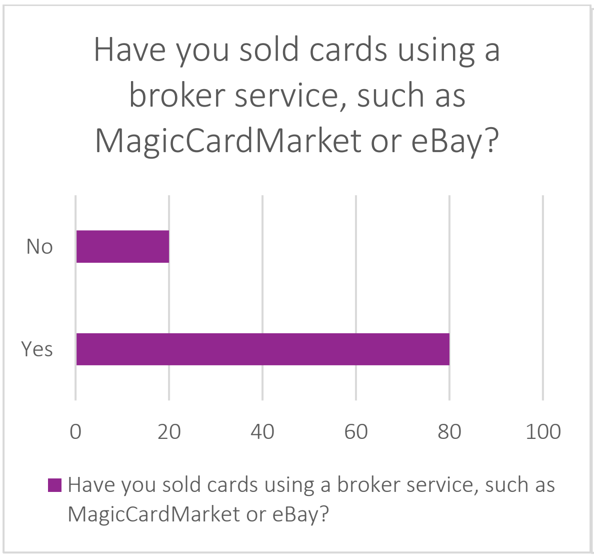


Figure 4



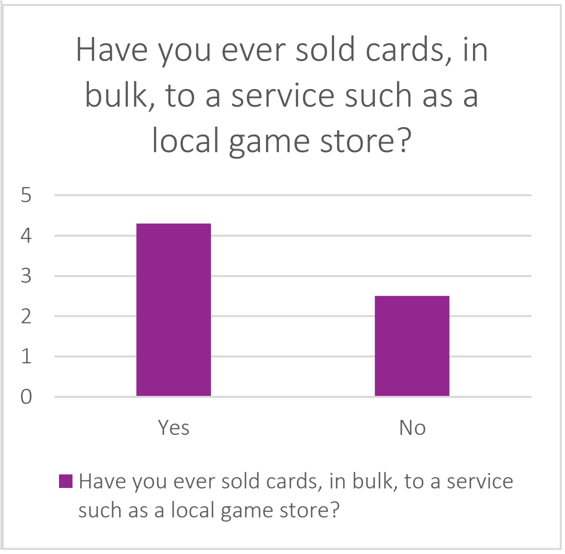


Figure 5

Figure 6

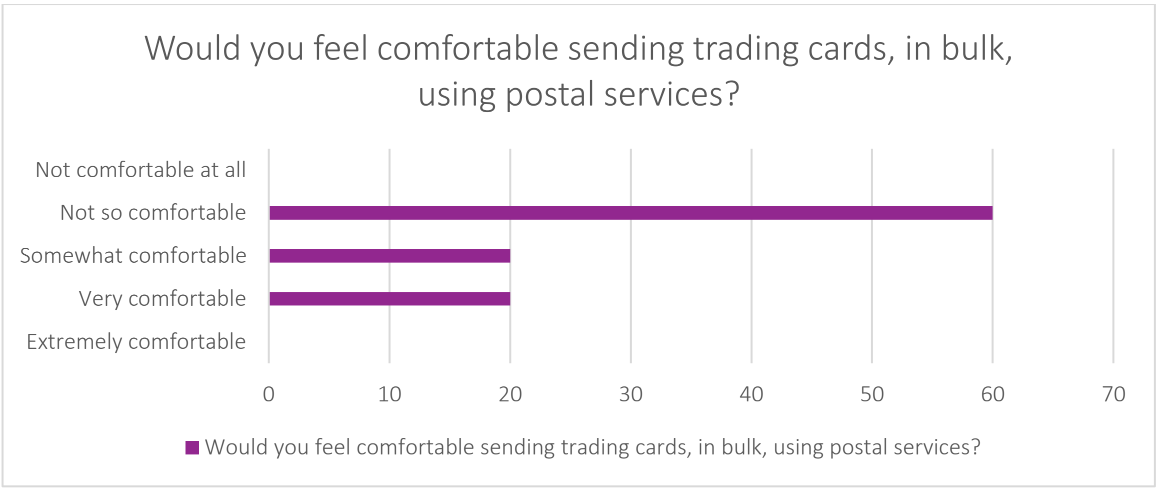


Figure 7

To gather information regarding the needs of the target audience, a survey was made and distributed within online card game groups. While responses to this survey were few, it provided insight in the needs of the participants. The results survey indicated interest in the service, with 80% of participants claiming that they would be interested in a service that offers assistance in the sorting and selling of cards (fig 4). However, participants also stated that they generally would not feel comfortable in sending these cards by post (fig 7). In addition, these users on average approximated that they own 3180 cards (fig 2) and are willing to sell 35% of these (fig 3), leaving an average cards per user to be sold of 1113. Though determining the value of these cards in terms of profit, an approximate average of each card valuing at 1.74USD (1.41GBP as of 07 May 2020) leaves this number at 1936USD. If this statistic translates correctly, then the tool when created and offered as a service into the industry, could be extremely profitable. 80% of participants in the survey expressed that they have used broker services in the past to sell cards (fig 5). They also expressed that the sorting and pricing of cards was an issue they had during the process of using these services. Additionally, only 40% of participants indicated that they have sold cards in bulk to local game stores (fig 6). Of these participants, only one criticism was listed in which the complaint was that they would like a breakdown of individual costs of cards. Key information also discovered in the survey is that individuals do not tend to own only one card game. This indicates that any features created should be easily modifiable as to maximise the number of cards available to be scanned. This information directly translates to a functional requirement of the system.

From these results, a list of key details gathered from the survey can be expressed.

- Participants are interested in the service and are willing to use it
- Participants have approximately 1113 cards that they are willing to sell
- Participants have used online broker services but struggled with sorting and listing cards with the right prices
- Participants do not generally sell cards to local game stores in bulk, however those who have would prefer if they were provided an individual card cost breakdown

These key points are interesting and necessary to note as the project's potential profit is particularly expressed by the number of cards owned by participants' average earnings. Additionally, there are issues and problems expressed both with selling through local game stores and selling to online broker services that can be solved through the project's application. Cards can be more easily sorted and sold to online services with the project's application, and local game stores can provide more thorough card cost breakdowns. This survey provides a stable backbone of information to ensure that the specification of the product does not stray far from the desired audience's requirements.

3.1.2 Functional Requirements

As stated in the beginning of this chapter, the requirements of a project do not purely rest on the requirements of the user. Some user requirements are outside of the scope of the project and some are not applicable to the project. To ensure that the project has a clear indication of the direction that it should progress in, a list of functional requirements is listed below.

- Using sensors, the system should be able to recognise which card is being processed and act accordingly.
  - This sensor is likely to be a camera as the image on the front face of the card is the most defining feature.
- A mechanical component is required of the project to properly sort the cards and automate the scanning process.
  - Due to equipment available and ease of use, this is likely to be a robotic arm.
- Information gathered on cards scanned must be stored in data for later use.
  - This is likely to simply be the name of the card and, if possible, the set the card is from. These two pieces of information are enough to uniquely identify cards.
- Data stored must be able to be used to transfer cards to online broker services such as eBay.
  - As the correct pricing for this card is dependent on market fluctuation, it must be gathered at the time of listing

With these functional requirements specified, the project can enter the design stage with a clear and distinct path to progress on.

3.2 Design

Since the requirements of the project have been gathered and clarified, the design can begin. The design of this project has two aspects to it: software and mechanical. As such, they will be treated as distinct but related to each other, as the software cannot operate without the mechanical and the mechanical cannot operate without the software. As Pressman states, to accomplish a suitable design the problem must first undergo diversification of potential solutions and then conclude with convergence of these solutions into the final solution (2015).

3.2.1 Architectural design

To begin, the project was considered from a generally abstract approach. This process, known as architectural design, aims to provide a general overview of the project to ensure it fulfils the requirements that the project must adhere to (Pressman, 2015). As such, is it an essential part of project design.

The architectural design elements to be adhered to are as follows:

1. A card or cards is given to the system
2. Sensors identify the card
3. The card is sorted mechanically
4. The card is stored digitally for later use

3.2.2 Mechanical Design

The design was then considered from a mechanical perspective. As the code used is dependent on available APIs and architectures available, the hardware is of utmost consideration to assist with the ease of development. To begin with, the Dobot (2020) conveyor belt was tested. Multiple methods for distributing cards were tried, but as it could not be determined that a single card would be distributed in this case, it was quickly determined that the conveyor belt would not be suitable. As such, the Dobot M1 (2020a) was considered. This arm was considered mechanically suitable for the task, and the suction cup attachment could correctly ensure that only one card was lifted at a time. This made it appear to be the suitable choice for development, and as such it was chosen. A camera, the Trust Trino HD Webcam (2020) was also provided by Edge Hill University and as such was chosen for use. A laptop was also provided by the university for development, and all work was to be continued on this laptop. Additionally, a tray was also required for the sorting of cards. This tray would be 3D printed using tools at Edge Hill University with permission from the department.

Below is a table listing all hardware determined to be necessary for the system.

Table of Hardware

Hardware Name	Hardware Purpose	Additional information
Dobot M1 Mechanical Arm	Movement and sorting of cards	
Suction Cup Attachment	Attach cards to arm to allow distribution	
Trust Trino HD Webcam	Scan cards for data entry and sorting	
Laptop	Connect with camera and robotics	
Tray	Sort cards based on category	To be 3D printed

With the hardware to be used determined, hardware could be designed. A sketch of the proposed design is found in fig 8.

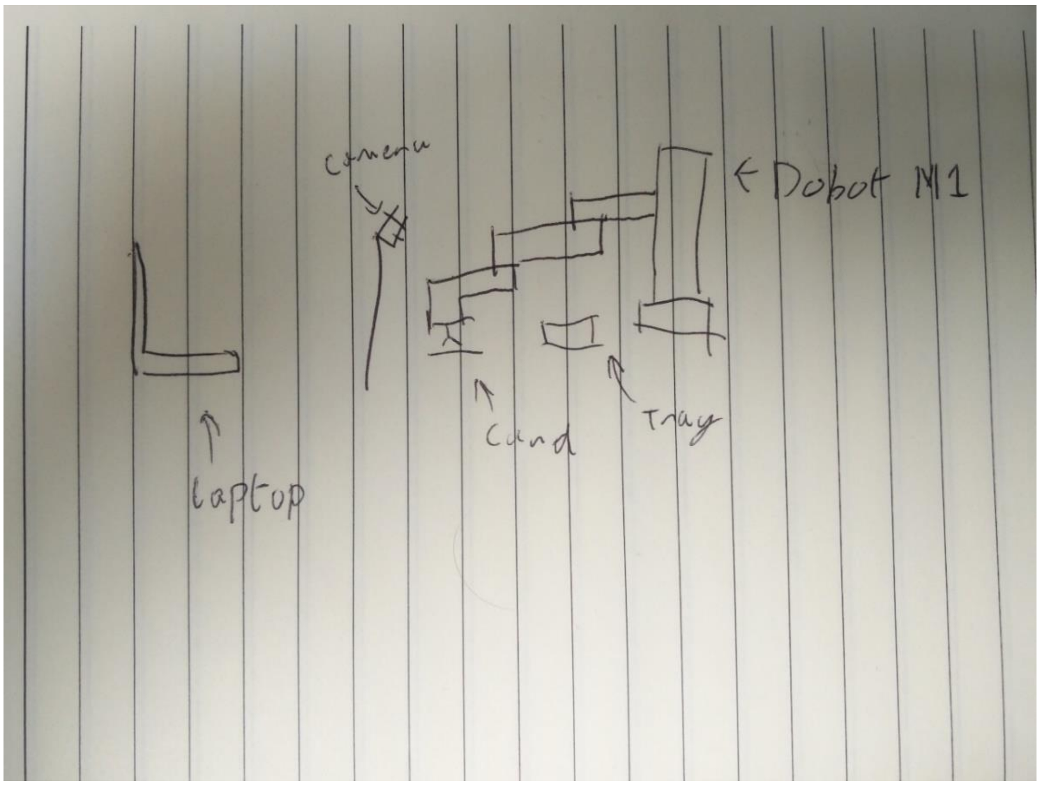


Figure 8 Design of Hardware

As the setup for the hardware has been determined, the design of the software could then be commenced.

3.2.3 Software Design

The software necessary to be used was determined by first researching the parts determined be used in the hardware design. Upon research, certain key pieces of software were noted. To begin, OpenCV (2020), a computer vision library with interfaces with C++, Python and Matlab. Since this is a key piece of software, Python was chosen from these three due to familiarity. OpenCV has multiple options for image recognition and based extensive review of their capibilities, it was determined that SURF and SIFT feature matching algorithms would be ideal during the development process. Since the other piece of hardware to be interfaced with is the robotic arm, research was performed on this field. This discovered information on a tool used widely in fields across the industry and academia, ROS or Robot Operating System (2020). This tool was determined to be suitable so was marked for use during development. These are the two key elements required for the purpose of use during development, however other tools are needed during development. Firstly, an IDE (Integrated Development Environment) was needed to assist in development. Due to tools available on the laptop, this was quickly chosen as Spyder (2020). Spyder is an IDE for working with Python and provides project level views as well as code inspection. These features help with code organisation and implementation, and as such are useful during the development process. The next piece of software chosen was for the use of the 3D modelling of the tray to be used in the mechanical aspect of the project. Due to availability and ease of use, Blender (2020) was chosen. Though experience with this software was extremely limited at the time, it was determined that there were plenty of resources to learn from and improve ability to level of being able to model what was needed. Once this software was determined, a full list of software requirements could be determined for development.

Table of Software

Software Name	Hardware Purpose	Additional Information
Python	Programming Language	
OpenCV	Computer Vision Library	
Robot Operating System	Interface with robotics	
Spyder	Programming IDE	
Blender	3D modelling software	

As both the software and hardware had been determined, the convergence of these features could begin.

3.2.4 User Interface Design

One element that is not primarily focused on either hardware or software, but is equally important, is User Interface Design. Put simply, User Interface Design is the process of presenting information in an easy to understand and accessible way that naturally aligns with human intuition and conventions (Galitz, 2007). As such, when designing a User Interface, careful consideration must be taken for the size and position of objects and their relation to other objects, alike objects should be grouped and objects without relevance to one another be separated. With this in mind, a concept of the UI applying these elements was created.

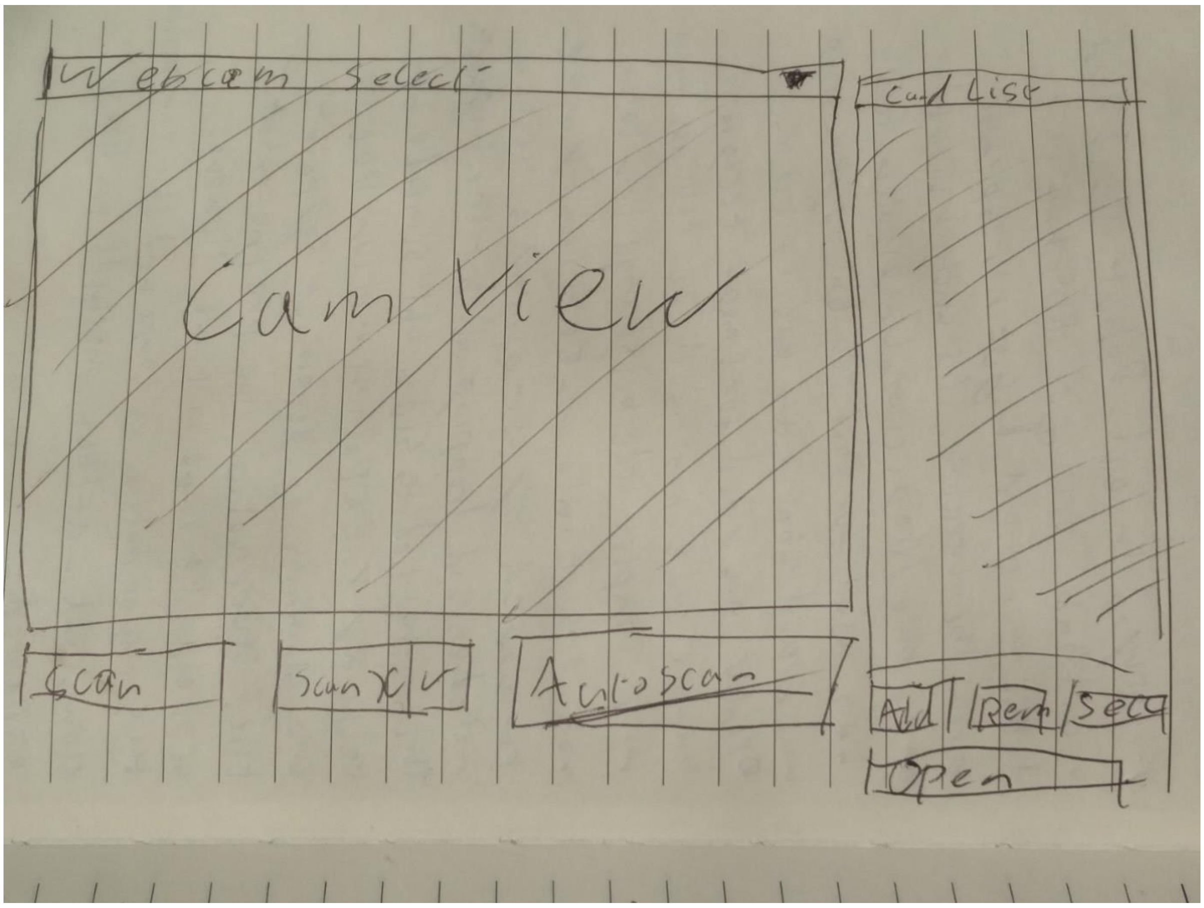


Figure 9 User Interface Concept Art

This design adheres to these basic concepts and features all elements required for the application to run as designed. As such, this design will be used during implementation.

3.2.5 Design Convergence

Now that the software and hardware requirements have been defined, the complete pipeline integrated them both can be created. This pipeline was thoroughly discussed but kept abstract. A sketch was created to demonstrate how the process will perform, shown in fig 9.

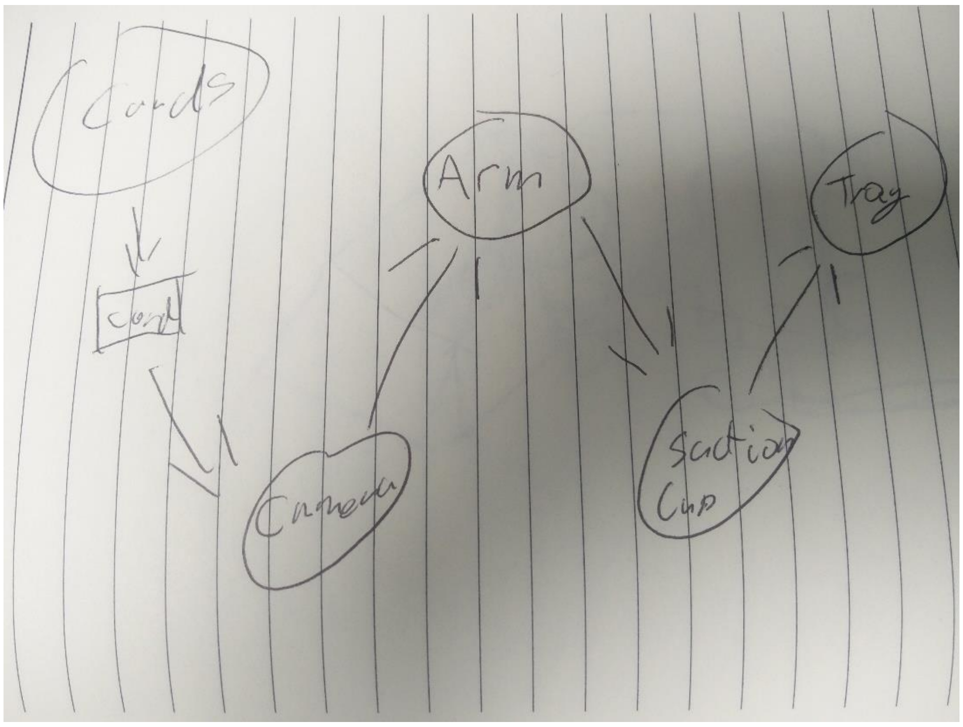


Figure 10

This design being completed as well as the parts being gathered meant that the implementation of the project could begin and the project progress.

Chapter 4 – Implementation

4.1 Implementation

Implementation began on the Dobot M1 as planned, and ROS was examined. Through reading the documentation, a greater understanding of the ROS framework was understood, however it was then realised that the Dobot M1 does not have support for the ROS. It was then that a GitHub repository, "Dobot M1 ROS Control" (2019). While this repository looked promising, the many trials and efforts put into getting it so failed and interest began to become focused on other available alternatives. It was then that, noting the significance of it being a newer model, the Dobot Magician (2020b) was researched as an alternative. The discovery of a Python library for interfacing with the Magician, "pydobot" (2020) heavily influenced the decision to switch. After discussion, the higher range of motion, ease of development, and more up to date software and firmware concluded the decision that development would switch from the Dobot M1 and to the Magician. As such, the tables for hardware and software required must change.

Table of Hardware

Hardware Name	Hardware Purpose	Additional information
Dobot Magician Mechanical Arm	Movement and sorting of cards	Decision to switch made during implementation
Suction Cup Attachment	Attach cards to arm to allow distribution	
Trust Trino HD Webcam	Scan cards for data entry and sorting	
Laptop	Connect with camera and robotics	
Tray	Sort cards based on category	To be 3D printed

Table of Software

Software Name	Hardware Purpose	Additional Information
Python	Programming Language	
OpenCV	Computer Vision Library	
Robot Operating System	Interface with robotics	
Spyder	Programming IDE	
Blender	3D modelling software	
Pydobot	Library for interfacing with Dobot Magician	Added after development switched to Dobot Magician

With these new changes in place and the new hardware and software confirmed working, the computer vision components of the project were then researched and developed.

4.2 Pre-COVID-19

Due to the impact of the COVID-19 pandemic, it is important to split this project into two halves. Due to significant and substantial adjustments made to the project in order to adapt to the COVID-19 pandemic, the project can be treated to two halves to one whole. The COVID-19 pandemic made access to robotics required for the project unavailable and, as such, no elements requiring these components could be included in the final version of the software. This caused the project to be fractured into two separate parts that each fill certain requirements of the criteria. However, any features or changes made in the post-COVID-19 portion of the implementation have been given strong consideration so that they would require minimal effort to integrate the tools developed in the pre-COVID-19 portion of the report.

Firstly, the parts developed before the COVID-19 outbreak will be discussed. This will then follow with a more detailed discussion on decisions made to adapt to the COVID-19 situation and conclude with the report on the implementation process and decisions during the second half of this development.

4.2.1 Computer Vision

As per the design, the OpenCV Python library was used for the image recognition of the cards. The SURF and SIFT algorithms that were decided upon during the design of the system were researched and found that they were no longer supported due to a proprietary license conflicting with the open source nature of the library. Although no official statement from the OpenCV team could be found, the license on SURF and SIFT restrict the use of the algorithm for non-academic works and, as such, is at conflict with the OpenCV's free nature. Because of this, OpenCV version 4.2 was used during development as this was the latest version found without the removal of the SURF and SIFT algorithms.

Tutorials and guides on how to use the OpenCV libraries were easy to come by and development quickly begun working at full speed. Documentation found on the OpenCV website was followed and by the end of the week, a brute force image matching algorithm system had been developed.

```
# Code based off of code found in https://docs.opencv.org/master/dc/dc3/tutorial_py_matcher.html

import numpy as np
import cv2
from matplotlib import pyplot as plt

img1 = cv2.imread('avacyn.jpg',0)          # queryImage
death = cv2.imread('death.jpg',0)
img2 = cv2.imread('camdeath.jpeg',0) # trainImage

# Initiate SIFT detector
sift = cv2.xfeatures2d.SIFT_create()

# find the keypoints and descriptors with SIFT
kp1, des1 = sift.detectAndCompute(img1,None)
kspDeath, desDeath = sift.detectAndCompute(death, None)
kp2, des2 = sift.detectAndCompute(img2,None)

# FLANN parameters
FLANN_INDEX_KDTREE = 0
index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
search_params = dict(checks=50)   # or pass empty dictionary

flann = cv2.FlannBasedMatcher(index_params,search_params)



matches = flann.knnMatch(des1,des2,k=2)
matchesDeath = flann.knnMatch(desDeath, des2,k=2)

print(len(matches))
print(len(matchesDeath))


# Need to draw only good matches, so create a mask
matchesMask = [[0,0] for i in range(len(matches))]
matchesMaskDeath = [[0,0] for i in range(len(matchesDeath))]

numMatches = 0
numMatchesDeath = 0

# ratio test as per Lowe's paper
for i,(m,n) in enumerate(matches):
    if m.distance < 0.7*n.distance:
        matchesMask[i]=[1,0]
        numMatches += 1

for i, (m,n) in enumerate(matchesDeath):
    if m.distance < 0.7*n.distance:
        matchesMaskDeath[i]=[1,0]
        numMatchesDeath += 1

if (numMatchesDeath > numMatches):
    matches = matchesDeath
    matchesMask = matchesMaskDeath
    img1 = death

draw_params = dict(matchColor = (0,255,0),
                   singlePointColor = (255,0,0),
                   matchesMask = matchesMask,
                   flags = 0)

img3 = cv2.drawMatchesKnn(img1,kp1,img2,kp2,matches,None,**draw_params)

cv2.imwrite('image.png', img3)
cv2.imshow('image', img3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Figure 11 Version 1 of Image Matching Algorithm

In this prototype version of the code, two images were simply being compared to one base image. If one of the images to be compared had more matches than the other, that would be the image displayed once the software has finished running. This version is not scalable and does not use camera images, however the code proved that the image recognition and matching technique worked. It was then obvious that the algorithm, to serve its purpose, would need to be functionally scalable to work with sets of cards, that may have many various data set sizes. For this functionality, the card set Shadows Over Innistrad (2016) from the Magic: The Gathering card game was chosen to be analysed. An image set was downloaded, and images within it matched based on their resemblance to an image chosen from the set.

```
# Code based off of code found in https://docs.opencv.org/master/dc/dc3/tutorial_py_matcher.html

import cv2 as cv2
import numpy as np
import os

class ImageData:
    image = None
    keypoints = None
    descriptors = None
    good_matches = None


images = []
for index, file in enumerate(os.listdir(".\SOI")):
    images.append(ImageData())
    images[index].image = cv2.imread(".\SOI\\" + file)



cameraImage = cv2.imread("omnathcam.png")

matcher = cv2.DescriptorMatcher_create(cv2.DescriptorMatcher_FLANNBASED)

minHessian = 400
detector = cv2.xfeatures2d_SURF.create(hessianThreshold = minHessian)

keypointsCam, descriptorsCam = detector.detectAndCompute(cameraImage, None)

for index, image in enumerate(images):
##    images[index] = [image, detector.detectAndCompute(image, None)]
    image.keypoints, image.descriptors = detector.detectAndCompute(image.image, None)
    knn_matches = matcher.knnMatch(image.descriptors, descriptorsCam, 2)
    ratio_thresh = 0.7
    image.good_matches = []
    for m,n in knn_matches:
        if m.distance < ratio_thresh * n.distance:
            image.good_matches.append(m)

bestMatch = images[0]
for image in images:
    if (len(image.good_matches) > len(bestMatch.good_matches)):
        bestMatch = image

img_matches = np.empty((max(bestMatch.image.shape[0], bestMatch.image.shape[0]),
bestMatch.image.shape[1]+cameraImage.shape[1], 3), dtype=np.uint8)
img3 = cv2.drawMatches(bestMatch.image, bestMatch.keypoints, cameraImage, keypointsCam,
bestMatch.good_matches, img_matches, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)


cv2.imwrite("image2.png", img3)
cv2.imshow('Good Matches', img3)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Figure 12 Version 2 of Image Matching Algorithm

This version of the code revealed a key optimisation that could be made. The images, when being matched, were being processed and keypoints calculated from them. This was not necessary to be performed each time the algorithm was ran, as these keypoints could be calculated once and used for all future runs. For this process, a separate Python file was created. This file, when run, performs the keypoint calculation and saves extracted data as arrays containing objects within a file. These files can then be opened later and objects within them used to minimise processing time. This optimisation halved the running time of the program, from approximately 1 minute 30 seconds per run to 45 seconds.

```python
import cv2 as cv2
import numpy as np
import os
import pickle
import copyreg
import requests
import json


class ImageData:
    name = None
    image = None
    keypoints = None
    descriptors = None
    good_matches = None
    cardId = None

class SetImages:
    images = []

def _pickle_keypoints(point):
    return cv2.KeyPoint, (*point.pt, point.size, point.angle,
                          point.response, point.octave, point.class_id)

copyreg.pickle(cv2.KeyPoint().__class__, _pickle_keypoints)


setImages = SetImages()
minHessian = 400
detector = cv2.xfeatures2d_SURF.create(hessianThreshold = minHessian)

setFileNum = 0
images = []
for index, file in enumerate(os.listdir(".\SOI")):
    print(index)
    print(file)
    workingImage = ImageData()
    workingImage.image = cv2.imread(".\SOI\\" + file)
    workingImage.keypoints, workingImage.descriptors = detector.detectAndCompute(workingImage.image, None)
    workingImage.name = os.path.splitext(file)[0]
    restCardName = workingImage.name.replace(" ", "+")
    resp = requests.get('https://api.scryfall.com/cards/named?exact=' + restCardName + '&set=soi')
    workingImage.cardId = resp.json()['multiverse_ids'][0]
    images.append(workingImage)
    if(index != 0 and (index + 1) % 15 == 0):
        setImages.images = images
        with open('.\setFiles\SOI\setFile' + str(setFileNum), 'wb') as file:
            pickle.dump(setImages, file, -1)
        setFileNum += 1
        setImages.images = []
        images = []
```
Figure 13 Pre-processing Algorithm

```
import cv2 as cv2
import numpy as np
import os
import pickle
import time


class ImageData:
    image = None
    keypoints = None
    descriptors = None
    good_matches = None

class SetImages:
    images = []

images = []
for index, setFile in enumerate(os.listdir(".\setFiles\SOI")):
    setFile = open(".\setFiles\SOI\\" + setFile, 'rb')
    setImages = pickle.load(setFile)
    setFile.close()
    for image in setImages.images:
        images.append(image)




cameraImage = cv2.imread("porttowncam.png")

matcher = cv2.DescriptorMatcher_create(cv2.DescriptorMatcher_FLANNBASED)

minHessian = 400
detector = cv2.xfeatures2d_SURF.create(hessianThreshold = minHessian)

keypointsCam, descriptorsCam = detector.detectAndCompute(cameraImage, None)

for index, image in enumerate(images):
##    images[index] = [image, detector.detectAndCompute(image, None)]
    timeStart = time.time()
    knn_matches = matcher.knnMatch(image.descriptors, descriptorsCam, 2)

    ratio_thresh = 0.7
    image.good_matches = []
    for m,n in knn_matches:
        if m.distance < ratio_thresh * n.distance:
            image.good_matches.append(m)
    print(time.time() - timeStart)


bestMatch = images[0]
for image in images:
    if (len(image.good_matches) > len(bestMatch.good_matches)):
        bestMatch = image

img_matches = np.empty((max(bestMatch.image.shape[0], bestMatch.image.shape[0]),
bestMatch.image.shape[1]+cameraImage.shape[1], 3), dtype=np.uint8)
img3 = cv2.drawMatches(bestMatch.image, bestMatch.keypoints, cameraImage, keypointsCam,
bestMatch.good_matches, img_matches, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)


cv2.imwrite("image2.png", img3)
cv2.imshow('Good Matches', img3)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Figure 14 Version 3 of Image Matching Algorithm

With this optimisation being a huge success, focus could then be placed on making sure the image matching algorithm worked with real camera-taken images. This stage was the most simple and smooth to perform, as the SIFT algorithm used was robust enough to perform the task without fail. One final version of the algorithm was created before moving on, this being the final version until the robotics were required.

```
import cv2 as cv2
import numpy as np
import os
import pickle
import time
from pydobot import Dobot
from serial.tools import list_ports
from PyQt5.QtWidgets import QApplication, QLabel

cap = cv2.VideoCapture(1)

class ImageData:
    name = None
    image = None
    keypoints = None
    descriptors = None
    good_matches = None
    cardId = None

class SetImages:
    images = []


def loadFiles():
    images = []
    for index, setFile in enumerate(os.listdir(".\setFiles\SOI")):
        setFile = open(".\setFiles\SOI\\" + setFile, 'rb')
        setImages = pickle.load(setFile)
        setFile.close()
        for image in setImages.images:
            images.append(image)
    return images



images = None
images = loadFiles()

cameraImage = cap.read()[1]

matcher = cv2.DescriptorMatcher_create(cv2.DescriptorMatcher_FLANNBASED)

minHessian = 400
detector = cv2.xfeatures2d_SURF.create(hessianThreshold = minHessian)

keypointsCam, descriptorsCam = detector.detectAndCompute(cameraImage, None)

for index, image in enumerate(images):
##    images[index] = [image, detector.detectAndCompute(image, None)]
    print(index)
    timeStart = time.time()
    knn_matches = matcher.knnMatch(image.descriptors, descriptorsCam, 2)

    ratio_thresh = 0.7
    image.good_matches = []
    for m,n in knn_matches:
        if m.distance < ratio_thresh * n.distance:
            image.good_matches.append(m)
#    print(time.time() - timeStart)


bestMatch = images[0]
for image in images:
    if (len(image.good_matches) > len(bestMatch.good_matches)):
        bestMatch = image

img_matches = np.empty((max(bestMatch.image.shape[0], bestMatch.image.shape[0]),
bestMatch.image.shape[1]+cameraImage.shape[1], 3), dtype=np.uint8)
img3 = cv2.drawMatches(bestMatch.image, bestMatch.keypoints, cameraImage, keypointsCam,
bestMatch.good_matches, img_matches, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)



cv2.imwrite("image2.png", img3)
listFile = open("cardList.txt", 'a')
listFile.write(bestMatch.name + ", ")
listFile.close()
cv2.imshow('Good Matches', img3)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Figure 15 Version 4 of the Image Matching Algorithm

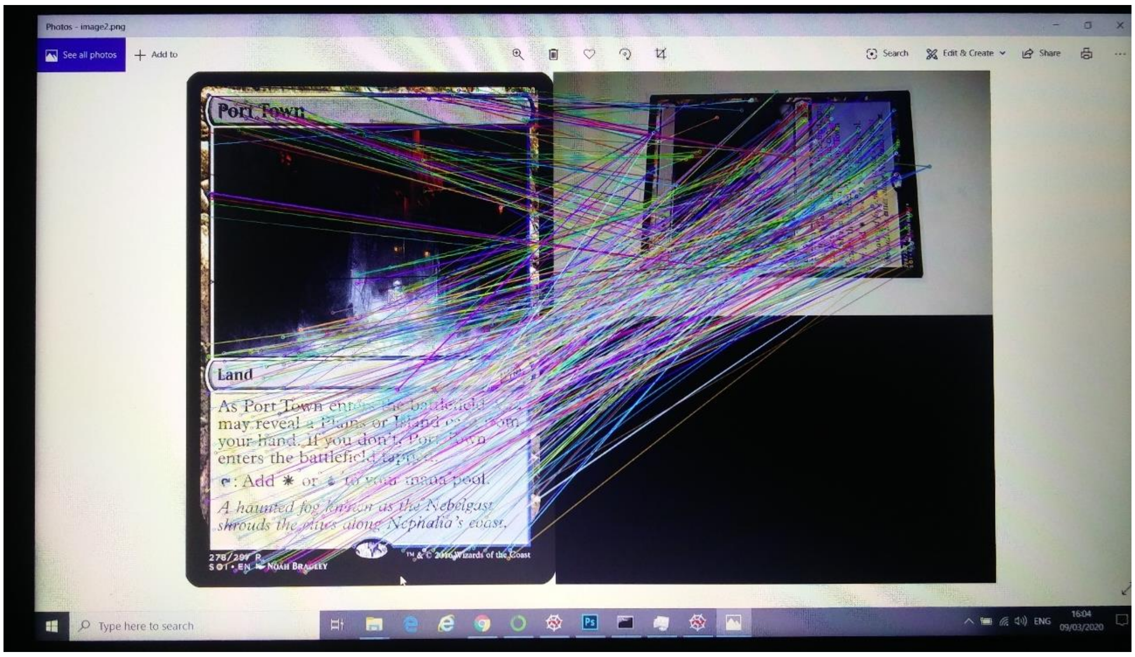


Figure 16 Image output of card scanned by camera

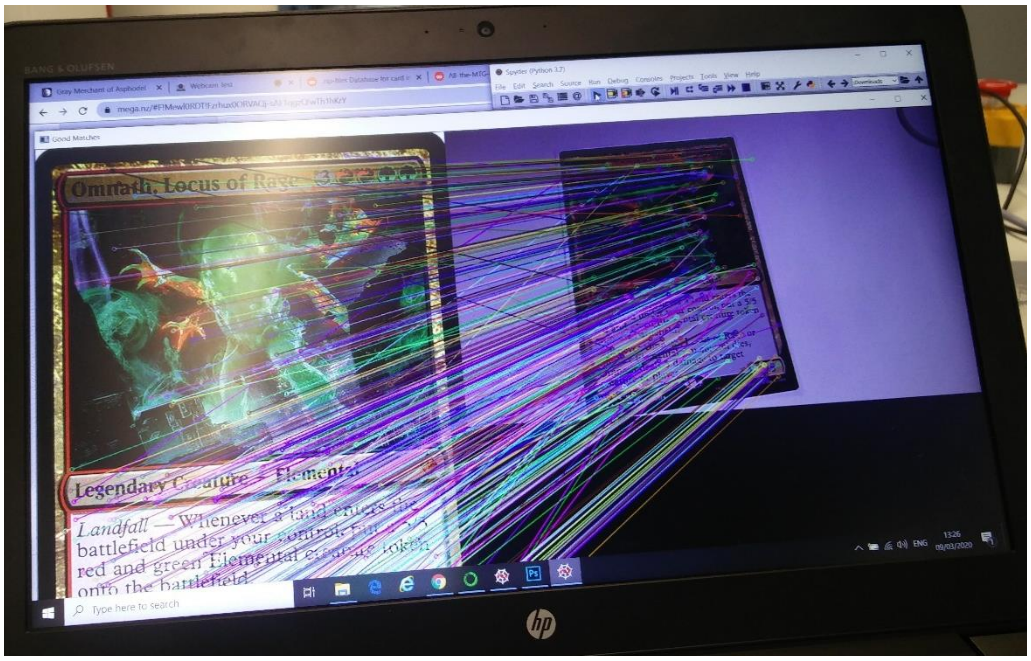


Figure 17 Output of card scanned by camera

It is important to note that while the method of image matching used in this software uses a brute force comparison algorithm, there are much more efficient ways of comparing and matching images. One such method to note is a k-Nearest Neighbor, or k-NN algorithm. This algorithm, when properly implemented, provides much greater levels of efficiency and speeds up the comparison of image data drastically. However, due to the difficulty of implementation and the suitability of the brute force matching algorithm for the demonstration of the process used to accomplish the aims and objectives for the project, it was decided that this comparison method would not be implemented.

4.1.2 Robotics

With the software completed, the hardware could then in turn begin formation. The first stage was to create a rig in which a camera can view and scan cards while staying out of the way of the robotic arm. For this purpose, the Trust Trino Webcam was zip-tied to a metallic frame. This frame could then be adjusted to ensure that the cards to be captured would fully be within frame when they are needed to be scanned. A photograph of this completed rig and setup can be found in fig 16.

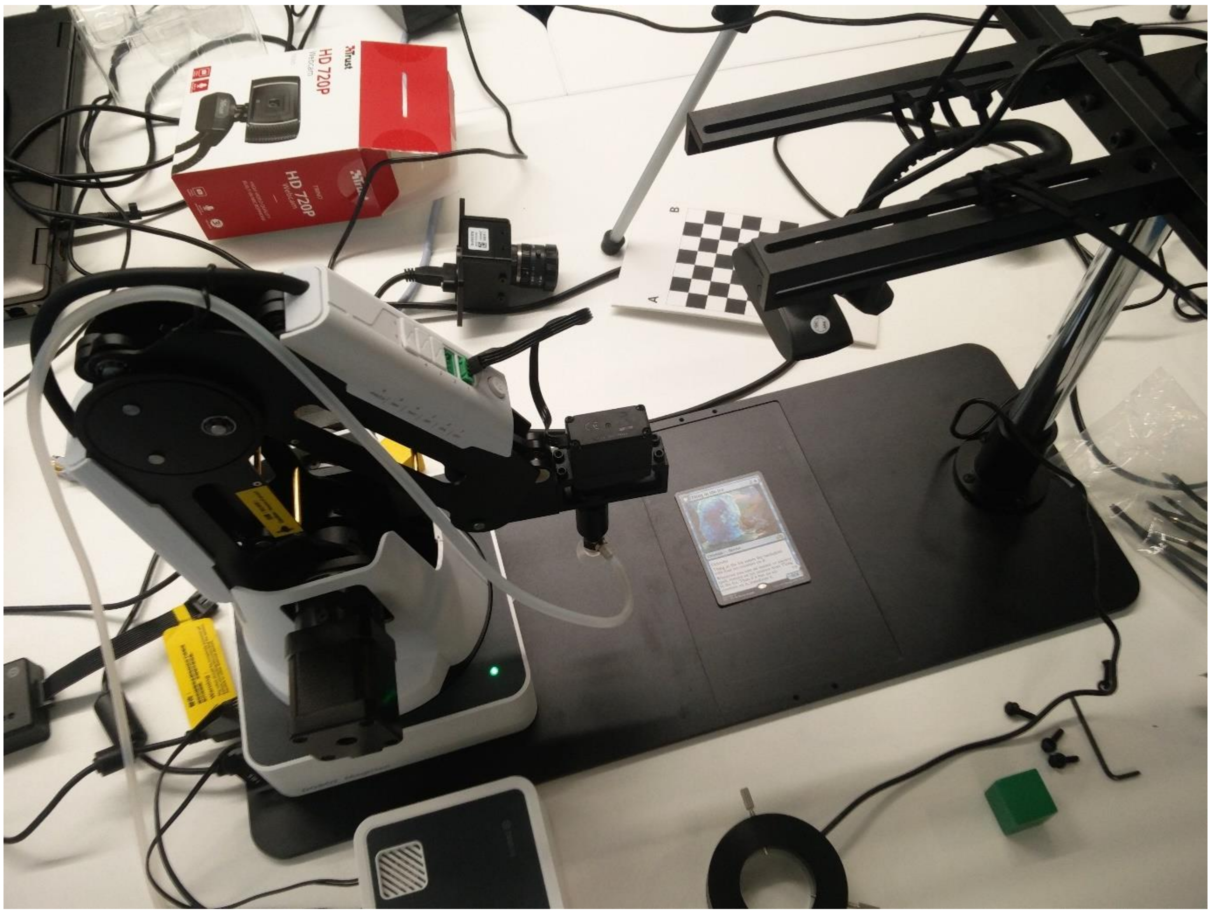


Figure 18 Completed Rig

With the use of the Python libraries found, developing software to control became a lot easier than expected. The Python libraries facilitate the ability for the robot to be programmed without the requirement of interfacing directly with Dobot APIs. This greatly increases ease of use and reduced development time. What may have taken days to learn took only the time needed to read a few lines of documentation.

```
from pydobot import Dobot
from serial.tools import list_ports

port = 'COM4'
device = Dobot(port=port)

(x, y, z, r, j1, j2, j3, j4) = device.pose()
print(f'x:{x} y:{y} z:{z} j1:{j1} j2:{j2} j3:{j3} j4:{j4}')
device.move_to(x + 90, y, z-60, r, 0x00)
device.suck(True)
device.move_to(x-150, y+200, z, r, 0x00)
device.suck(False)
device.move_to(x, y, z, r, 0x01)

device.close()
```

Figure 19 Dobot Control Code

Fig 19 shows lines of code extracted from the final version of the combined robot and computer vision code. This code, when ran with the rest of the code detailing the computer vision process, will first scan a card, then move it to an area to be stored, and then move back to its original position to scan the next card.

4.1.3 3D Printed Tray

As the robotics were now complete, the next step was to 3D print trays for the cards to enter. The designs for these trays were made in Blender, a 3D modelling program. Once created, the design was transferred to a 3D printer where the item was printed.

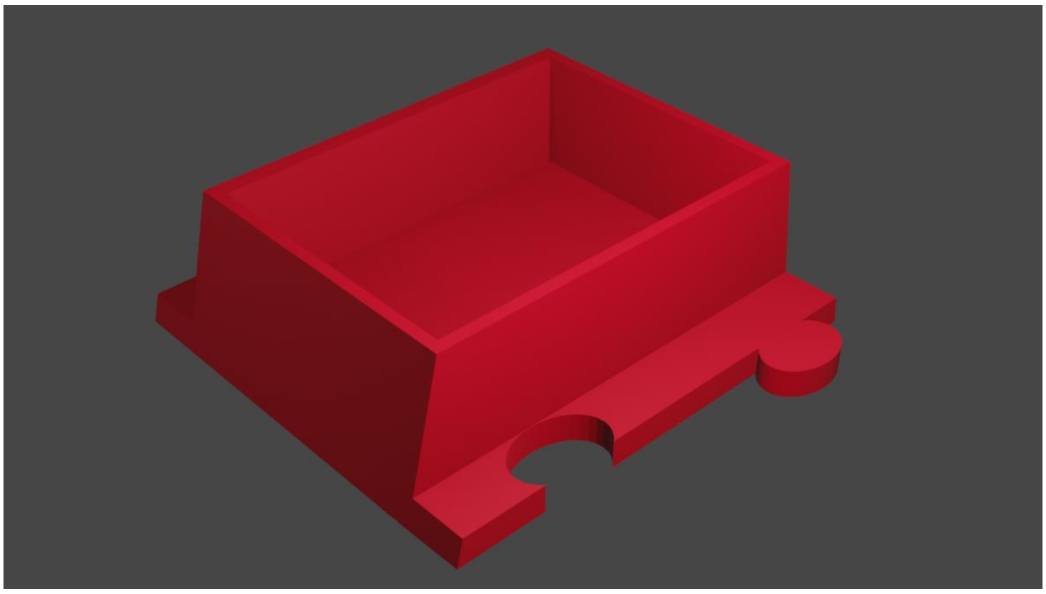


Figure 20 3D Model Design In Blender

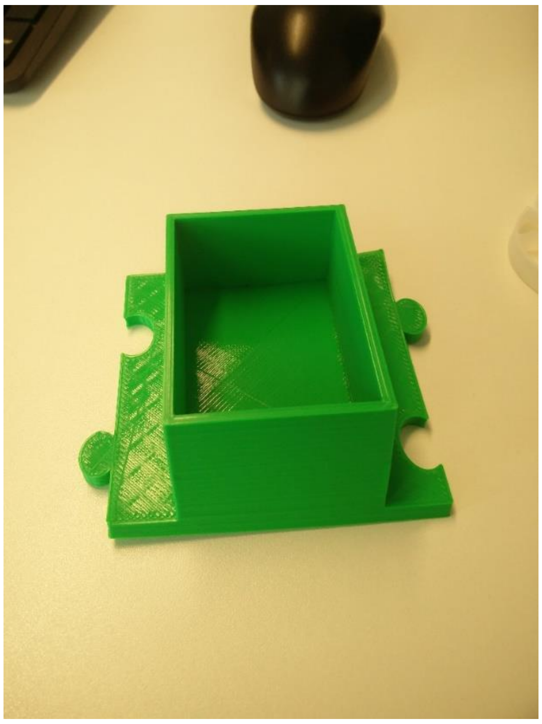


Figure 21 3D Printed Tray

4.3 Post-COVID-19

Upon the impact of the safety requirements of the COVID-19 pandemic, access to the robotics was lost. All hardware required except for the robotics, however, was already in possession personally and as such, development did not halt. Unfortunately, the robotics-oriented side of the project needed to be dropped due to this inaccessibility. As such, development following this point stripped robotics functionality and focused on what could be done regarding the development of the computer vision-based systems and user interface. Additionally, while the original scope of the project did not state how many cards would be used as data, it was at this point that the decision was made to limit the card set used to the Shadows Over Innistrad expansion.

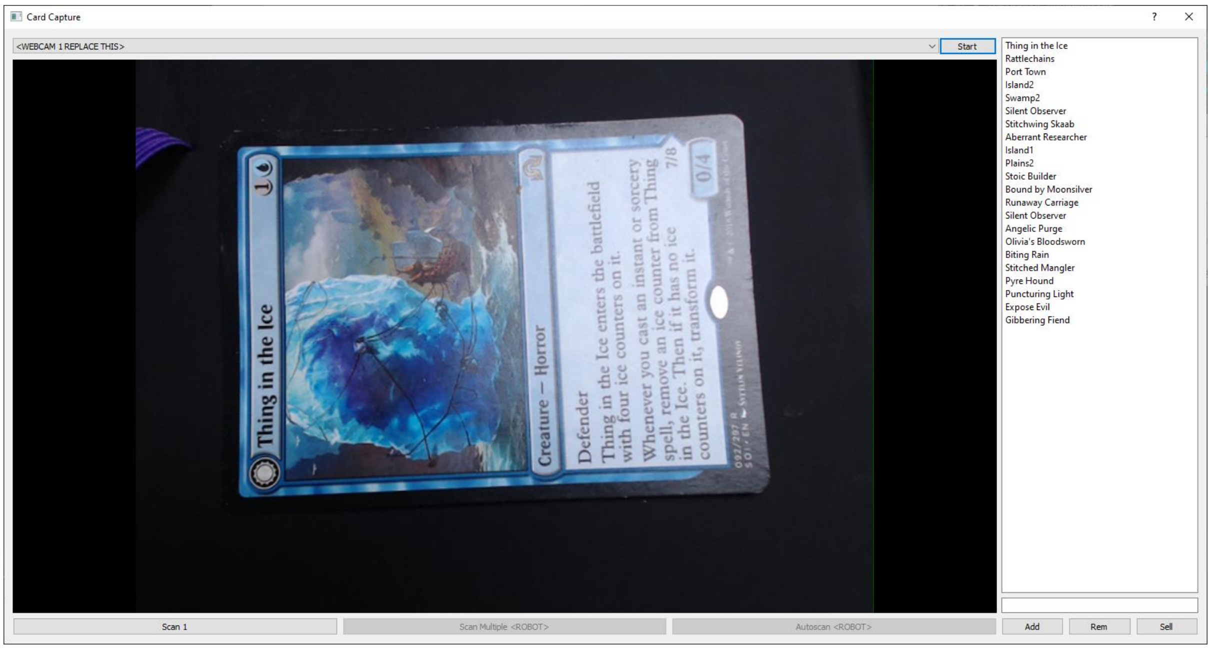
4.3.1 Computer Vision

To adapt to COVID-19, the code from this point was completed remotely and developed in absence of the robotics. With this brought about the task of setting up IDEs again. The IDE was switched to PyCharm (2020). This IDE was found to be easier to use. When setting up OpenCV on this machine, there were issues installing the OpenCV packages that are compatible with SIFT and SURF. It was because of these issues, that the decision was made to move to ORB based algorithms. ORB (2011) is an open source alternative to SURF and SIFT and provides a greater level of control, allowing accuracy to be traded for efficiency and vice versa. Switching to this provided many benefits and ensures that the software will work much further into the future than if SURF and SIFT were utilised. Once this was completed, the User Interface was to be implemented.

4.3.2 User Interface

Although the User Interface was not explicitly within the project scope, it was determined that, due to the change in routine in response to COVID-19, there would be enough time to program this element. As such, the User Interface was started immediately to ensure that minimal modification of existing code was necessary for its deployment.

Due to the nature of this feature being additional to the project rather than required, minimal research had been done into the software and methods behind programming user interfaces within Python. However, it did not take long to discover that the PyQt (2019) libraries are likely the most suited for this task. Upon following tutorials found on the website, iterative versions of the software were completed. This iterative process assisted with the modular nature of the PyQt libraries and user interface programming process.



Thing in the Ice
Rattlechains
Port Town
Island2
Swamp2
Silent Observer
Stitchwing Skaab
Aberrant Researcher
Island1
Plains2
Stoic Builder
Bound by Moonsilver
Runaway Carriage
Silent Observer
Angelic Purge
Olivia's Bloodsworn
Biting Rain
Stitched Mangler
Pyre Hound
Puncturing Light
Expose Evil
Gibbering Fiend

Figure 22 Completed User Interface

The design of the user interface made before implementation began was followed closely and resulted in a smooth and simple process allowing development to progress rapidly. However, no functionality was available in these earlier versions and as such that was the next stage. The first piece of functionality to be added was the camera. The camera feed was chosen to run from the same OpenCV camera functions that are used in the computer vision portions of the code. This solution provides two advantages: the first is that the feed from the camera can be directly fed to the computer vision components of the software when necessary, and secondly this reuse of code provides greater readability and reduces bloat of possible unnecessary libraries.

The second feature to be implemented is the sidebar card list. The sidebar card list is a way to view scanned cards after they have been recognised by the computer vision algorithm. The primary objective was for the card list to be concise and easy to use. It is for this reason that adding a card and removing a card are two simple processes. To add a card, you type the name of the card in the text box below the card list and press the "Add" button. To remove a card, you select the card you wish to remove and press the "Rem" button. This process is intuitive, simple to do and extremely readable. As such, it complies to the user interface design paradigms expressed in the design section of this report. The storage of the cards was done using a CSV file. Using an easily editable and readable format such as CSV provides multiple advantages. Firstly, CSV is a low-level file format without great levels of abstraction. This means that, if necessary, the data within the CSV can be edited at will with rudimentary tools. Secondly, CSV files are a commonly used file format and as such data can be imported and export more easily by using this method. The greater ease of use and functionality when compared to an approach, such as object serialisation, are much more beneficial to the user and as such were used in the final project.

To implement existing features within the user interface, substantial but non-difficult changes to the code were necessary. The code found within the computer vision sections was needed to be added to classes and functions to allow it to be ran only when necessarily called by user interface calls. As such, this method required multiple iterative runs to ensure that the functionality of the program was as desired. The card scanner was adapted by placing the code within a class, in this instance AppOrb, and creating a function, scan_card, in which an image can be provided to perform the image comparison process upon. This method allows the card scanning to be started from within the user interface and as such provides the functionality required to make the existing computer vision code compatible with the user interface.

4.3.3 eBay functionality

The last main feature from the project scope to be added is the functionality to add items to the eBay marketplace. This functionality requires knowledge of the eBay API, authentication, verification, and image uploading methods. Additionally, card information must also be gathered. Due to the nature of the marketplace, card pricing data is prone to market fluctuation and as such must be gathered upon the listing of the card on the website. As such, additional APIs are required to gather this information.

To begin, the decision was made to ensure that the software was object oriented from the beginning. This object-oriented approach provided that the software be compatible with the user interface when it was necessary to implement it as such. Due to the unpredictable nature of ever-changing online APIs, error handling was also to be added to key components to ensure that if something is to fail, the software informs the user of what happened so that the problem can be fixed.

Firstly, the multiverse ID of the card needed to be gathered. This multiverse ID is a unique identifier for cards within the Magic the Gathering card game. Once this multiverse ID has been gathered, additional information to be used during the upload process can be gathered and used in the marketplace listing. This includes the name, the price, and the set the card is from.

Once this information has been gathered, additional preparation begins. Since the more accurate currency is in United States Dollars, the card price is multiplied by 0.8 as an approximate conversion of USD to GBP. Then, as a measure to comply with eBay's minimum pricing policy, if the price of the card is less than £1 it is set to £1. All decimals except the last two are stripped to also comply with eBay's listing APIs. Then, to ensure the image to upload is up to date, the image of the card is extracted from the Gatherer website by Wizards of the Coast. This website is an official repository of information pertaining to cards in the game and provides images of each card. Extracting from this site ensures that the image is suitable. Then, the image is resized to comply with eBay's minimum image size policy. This resized image is then uploaded to another website, PictShare (2020). Uploading to another website first is, again, to comply with eBay's image upload policy. It is possible to upload an image without using this method, however in the time of development a solution to this problem could not be discovered. However, due to an error, possibly with PictShare's API, it was required to add a slash after the end of the domain name extension. This was done using regex and was, in the case of the issue being fixed by the developers of PictShare, provided compatibility code to ensure that the software works as intended if the problem ceases to exist.

It is at this point that the marketplace uploading process itself begins. However, this itself is not without additional details to discuss. For items to be uploaded to the eBay marketplace, verification must first begin. This can be done through multiple methods, however for the sake of the project and security, the method that ensures the least amount of personal data processing was chosen. To ensure this, authorisation information must first be provided in the eBay.yaml file found in the root directory of the software. There are two methods of user authorisation that were available for the project. The first of which is to provide the user with a login page, in which the user's information would then be gathered automatically upon authorisation. The second of these is to ask the user themselves to provide the user's information. The first of these solutions can be very complicated to ensure that user's information is not vulnerable during the process (Farrel, 2009). The second of these requires more steps from the user, however it does not include the security concerns that completing this process through software faces. The second of these two choices was chosen as, while complicated, it provides the user functionality of the software while having the least potential impact on the user’s data security. Additionally, the lack of ethical concerns involving the use of data within the project allows the development to move smoothly forward without worry that rights are being violated. This information found within the eBay.yaml file must be filled in by following the steps found within the README.md file. Additional information such as the postcode and currency of the listing can be found in the details.yaml file.

With all this information gathered, the item can then be listed to eBay. Once this is done, the program will prompt to user to confirm that the item has been listed correctly. If a failure message is provided by eBay, it will prompt the user telling them that the upload is failed. However, warning messages from eBay will in certain occasions not list the item and in other occasions list the item without fail. As such it has not yet been determined what the method for ensuring the item has been listed and the user must confirm this.

With this completed, the eBay upload process is complete and the software returns to the user interface, ready for the next task to be provided.

Chapter 5 – Testing

5.1 Testing

Testing is a required part of any project, as without it, problems that the project may be facing or will face go unknown until they occur. Additionally, it allows the judging of products to judge and verify that the quality of the project is suitable for the purpose is has been created (Jorgensten, 2014). There are many testing methodologies available. One testing methodology that was used in this project is white box testing. White box testing is a type of software testing where the user is aware of the code structure of the project and how the project works. As such, it is most commonly done by developers of the software who already have extensive knowledge of the project (Nidhra and Dondeti, 2012). Although it was originally planned that User Acceptance Testing (Hambling and Van Goethem, 2013) would be used, the COVID-19 pandemic made this unreasonable to pursue. Additionally, for this reason the robotics-focused half of the project cannot be tested.

5.1.1 White Box Testing

To perform white box testing, a list of features to test and the possible types of failure that the software can perform was created. Then, the software was tested against these elements and the expected response detailed. The findings can be found below.

Test	Expected Response	Response	Additional Notes
Press the "Start" button without camera	Error message	Error message	
Press the "Start" button with camera	Camera feed displays	Camera feed displays	
Press the scan button without camera	Error message	Error message	
Press the scan button with camera	Card is scanned	Card is scanned	
Press add button with no text	Blank entry added	Blank entry added	
Press add button with text	Card added to list	Card added to list	
Press remove button with no item selected	Nothing	Nothing	
Press remove button with item selected	Item removed	Item removed	
Press sell button with no item selected	Error message	Error message	
Press sell button with item selected	Card listed to market	Card listed to market	
Press sell button with invalid item selected	Error message	Error message	

Chapter 6 – Conclusion

6.1 How objectives have been met

All objectives laid out earlier in this report have all been met. Existing systems have been analysed in the literature review with their methods and limitations explained and discussed. Information has been gathered on the requirements of the target demographic and a design created appropriate to these needs. A system has been developed according to this design and then tested using white box testing. As such, no more requirements are necessary for the project to be deemed completed.

6.2 Critical Evaluation

The project overall ran smoothly, but there were some issues that required decision making to solve them. Firstly, the decision to change from the Dobot M1 to Magician was made. Factors leading to this decision were the lack of development tools available for easily coding for the M1 in comparison to the Magician. This was caused by lack of research prior to development into the M1. However, the change lead into smoother implementation as the Magician had many more resources available. In the future greater planning should be performed before the design and implementation phases occur. An additional issue within development was the timing alongside the COVID-19 pandemic. Although this could not be predicted, there were still steps that could have been made before it happened and decisions to make after the impact of the virus. While the development was well paced for the time allotted, if development had been more rapid in the beginning then the work developed would have been more suited for an unpredictable crisis such as this. When the pandemic occurred, two key decisions were made. The first of the key decisions was to ensure that project development within the university ceased and additional content was created remotely. This decision minimised the risk and danger to the public vulnerable to the virus. Although the COVID-19 pandemic is an unprecedented event, if an event like this is to occur, the same decision should be made. The second key decision made was to remove the robotics from the project. This decision improved the development process as attempting to develop for the Magician would have proved difficult or impossible. Development of these features could not continue and had to be dropped. This was the most suitable decision in the current situation and if a similar event were to occur the same decision should be made.

6.3 Future Work

The most important feature to be implemented is the completion of the features that were not possible to be completed due to remote development. This includes the merging of the robotics and image recognition within a user interface. As both aspects have been demonstrated to work, it is also true that a joining of these tools would also work. More Magic the Gathering sets should be implemented, as well as cards from other trading card games. Other features that could be included are the integration of other card games as well as additional websites such as Magic Card Market. Additional research should be performed into more efficient mechanical designs and the creation of more efficient image comparison algorithms.

6.4 Summary

This report has discussed the project’s background, the aims and objectives of the project and the methodologies used within the project. Additionally, a requirement analysis has been detailed and design explained. The implementation has been documented and testing performed. Finally, a critical evaluation of the project has been performed and additional areas to perform research have been recommended.

Reference List

ALTMAN, N.S., 1991. *An Introduction to Kernel and Nearest Neighbor Nonparametric Regression* [online]. Available from: https://ecommons.cornell.edu/bitstream/handle/1813/31637/BU-1065-MA.pdf [Accessed 10 March 2020].

BALAJI, S. and MURUGAIYAN, M., 2012. Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management* [online]. 2(1), pp.26-30. Available from: http://www.jitbm.com/Volume2No1/waterfall.pdf [Accessed 12 December 2019].

BARBASH, G., GILED, S., 2010. *New technology and health care costs--the case of robot-assisted surgery* [online]. Available from: https://www.semanticscholar.org/paper/New-technology-and-health-care-costs--the-case-of-Barbash-Glied/465fd6e50dc5c3c2583bc1d7a7b3b424e0101b9f [Accessed 21 February 2020].

BLENDER, 2020. *Blender* [online]. Available from: https://www.blender.org/ [Accessed 08 March 2020].

COVENTRY, T., 2015. *Requirement management – planning for success!* [online]. Available from: https://www.pmi.org/learning/library/requirements-management-planning-for-success-9669 [Accessed 07 March 2020].

DICE AND DECKS, 2020. *Dice and Decks* [online]. Available from: https://www.diceanddecks.com/ [Accessed 21 February 2020].

DOBOT, 2020a. *Dobot M1* [online]. Available from: https://www.dobot.cc/dobot-m1/product-overview.html [Accessed 21 February 2020].

DOBOT, 2020b. *Dobot Magician* [online]. Available from: https://www.dobot.cc/dobot-magician/product-overview.html [Accessed 21 February 2020].

EBAY, 2019b. *eBay Inc. Reports Third Quarter 2019 Results* [online]. Available from: https://www.ebayinc.com/stories/news/ebay-q3-2019-results/ [Accessed 21 February 2020].

EBAY, 2020a. *eBay* [online]. Available from: https://www.ebay.co.uk/ [Accessed 21 February 2020].

EDGE HILL UNIVERSITY, 2020a. *Tech Hub* [online]. Available from: https://www.edgehill.ac.uk/computerscience/facilities/techhub/ [Accessed 21 February 2020].

EDGE HILL UNIVERSITY, 2020b. *Edge Hill University* [online]. Available from: https://www.edgehill.ac.uk/ [Accessed 21 February 2020].

FARREL, S., 2009. API Keys to the Kingdom. *IEEE Internet Computing* [online]. 13 (5), pp. 91-93. Available from: https://ieeexplore.ieee.org/abstract/document/5233617 [Accessed 11 March 2020].

FONG, D., 2020. *MTG-Card-Reader* [online]. Available from: https://github.com/TrifectaIII/MTG-Card-Reader [Accessed 21 February 2020].

GALITZ, W., 2007. *The Essential Guide to User Interface Design* [eBook] 3rd ed. Wiley. Available from: https://www.dawsonera.com/abstract/9780470146224 [Accessed 09 March 2020].

HAMBLING, B., VAN GOETHEM, P., 2013. *User Acceptance Testing*. Swindon: BCS.

HASCHEK, C., 2020. *PictShare* [online]. Available from: https://pictshare.net/ [Accessed 10 March 2020].

HE, M., ZHANG, X., REN, S., SUN, J., 2016. *Deep Residual Learning for Image Recognition* [online]. Available from: http://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf [Accessed 13 March 2020].

HORNBERG, A., 2017. *Handbook of Machine and Computer Vision: The Guide for Developers and Users* [eBook]. John Wiley & Sons, Incorporated. Available from: https://ebookcentral.proquest.com/lib/edgehill/reader.action?docID=4821050 [Accessed 21 February 2020].

JAVA, 2020. *Java* [online]. Available from: https://www.java.com/ [Accessed 21 February 2020].

JEONG, H., 2014. *Architectures for Computer Vision: From Algorithm to Chip with Verilog* [eBook]. John Wiley & Sons, Incorporated. Available from: https://ebookcentral.proquest.com/lib/edgehill/reader.action?docID=1767023 [Accessed 21 February 2020].

JETBRAINS, 2020. *IntelliJ IDEA* [online]. Available from: https://www.jetbrains.com/idea/ [Accessed 21 February 2020].

JETBRAINS, 2020. *PyCharm* [online]. Available from: https://www.jetbrains.com/pycharm/ [Accessed 09 March 2020].

JORGENSTEN, P, C., 2014. *Software Testing: A Craftsman's Approach* [online]. Available from: http://mumbaibscitstudy.hostfree.pw/files/ebooks_files/bscit/sem-v/sem-v-ebooks_files/st/software-testing-a-craftmans-approach-4-edition.pdf?i=1 [Accessed 11 Match 2020].

LI, J., HU, Q., AI, M. and ZHONG., R., 2017. Robust feature matching via support-line voting and affine-invariant ratios. *ISPRS Journal of Photogrammetry and Remote Sensing* [online]. 132, pp. 61-76. Available from: https://www.sciencedirect.com/science/article/abs/pii/S0924271617300102 [Accessed 12 March 2020].

LIN, Y., ZHOU., H., CHEN., M. and MIN, H., 2019. Automatic sorting system for industrial robot with 3D visual perception and natural language interaction. *Measurement and Control -London- Institute of Measurement and Control* [online]. 52 (28), pp. 1-16. Available from: https://www.researchgate.net/publication/330435765_Automatic_sorting_system_for_industrial_robot_with_3D_visual_perception_and_natural_language_interaction [Accessed 12 March 2020].

LINUX KERNAL ORGANIZATION, 2020. *The Linux Kernal Archives* [online]. Available from: https://www.kernel.org/ [Accessed 21 February 2020].

LOGITECH, 2020. *Logitech C920 HD Pro Webcam* [online]. Available from: https://www.logitech.com/en-gb/product/hd-pro-webcam-c920 [Accessed 21 February 2020].

LOWE, D., 2004. *Distinctive Image Features from Scale-Invariant Keypoints* [online]. Available from: https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf [Accessed 12 March 2020].

MADDENFONG, 2018. *MTG Card Reader v1.1 Thesis Demo Backup* [online video]. Available from: https://www.youtube.com/watch?v=KvsBkOgKNgQ [Accessed 21 February 2020].

MANALEAK, 2020. *Manaleak* [online]. Available from: http://www.manaleak.com/ [Accessed 21 February 2020].

MCCONNEL, S., 1999. *Rapid Development: Taming Wild Software Schedules*. Microsoft Press: Redmond.

MESAS, L., 2020. *Python Library for Dobot Magician* [online]. Available from: https://pypi.org/project/pydobot/ [Accessed 09 March 2020].

MICROSOFT, 2020. *Windows* [online]. Available from: https://www.microsoft.com/en-gb/windows [Accessed 21 February 2020].

NIDHRA, S. and DONDETI, J., 2012. Black box and white box testing techniques-a literature review. *International Journal of Embedded Systems and Applications (IJESA)* [online]. 2 (2), pp. 29-50. Available from: https://s3.amazonaws.com/academia.edu.documents/38077013/Black_Box_and_White_Box_Testing_Techniques_-_A_Literature_Review.pdf [Accessed 21 February 2020].

OCADO, 2020. *Ocado* [online]. Available from: https://www.ocado.com/ [Accessed 21 February 2020].

OLAFENWA, M., 2020. *ImageAI* [online]. Available from: https://github.com/OlafenwaMoses/ImageAI [Accessed 21 February 2020].

OPEN ROBOTICS, 2020. *ROS.org | Powering the World's Robots* [online]. Available from: https://www.ros.org/ [Accessed 08 March 2020].

OPENCV TEAM, 2020. *OpenCV* [online]. Available from: https://opencv.org/ [Accessed 08 March 2020].

PAWLICKI, M., POLIN, J., ZHANG., J., 2014. *Prediction of Price Increase for Magic: The Gathering Cards* [online]. Available from: http://cs229.stanford.edu/proj2014/Matt%20Pawlicki,%20Joe%20Polin,%20Jesse%20Zhang,%20Prediction%20of%20Price%20Increase%20for%20MTG%20Cards.pdf [Accessed 13 March 2020].

PRESSMAN, R.S. & MAXIM, B.R. 2015. *Software engineering: a practitioner's approach*. 8th ed. McGraw-Hill Education: New York.

PYTHON, 2019. *PyQt* [online]. Available from: https://wiki.python.org/moin/PyQt/ [Accessed 10 March 2020].

PYTHON, 2020. *Python* [online]. Available from: https://www.python.org/ [Accessed 21 February 2020].

ROCA ROBOTICS, 2020. *Roca Robotics – Roca Sorter automatic Magic the Gathering card sorter* [online]. Available from: https://www.rocarobotics.com/ [Accessed 02 March 2020].

RUBLEE, E., RABAUD, V., KONOLIGE, K., BRADSKI, G., 2011. Orb: An efficient alternative to SIFT or SURF. *IEEE International Conference on Computer Vision (ICCV)* [online]. Available from: https://ieeexplore.ieee.org/document/6126544 [Accessed 10 March 2020].

SAKAJI, H., KOBAYASHI, A., KOHANA, M., TAKANO, Y., IZUMI, K., 2019. Card Price Prediction of Trading Cards Using Machine Learning Methods. *Advances in Networked-based Information Systems* [online]. 1036, pp. 705-714. Available from: https://link.springer.com/chapter/10.1007/978-3-030-29029-0_70 [Accessed 13 March 2020].

SICILIANO, B., KHATIB, O., 2016. *Springer Handbook of Robotics*. 2nd ed. Springer.

SMHALLER, 2019. *Dobot M1 ROS Control* [online]. Available from: https://github.com/smhaller/dobot-m1 [Accessed 09 March 2020].

SPYDER, 2020. *Spyder* [online]. Available from: https://www.spyder-ide.org/ [Accessed 08 March 2020].

TRULLS, E., *Announcing the 2020 Image Matching Benchmark and Challenge* [online]. Available from: https://ai.googleblog.com/2020/04/announcing-2020-image-matching.html [Accessed 12 March 2020].

TRUST, 2020. *Trino HD Video Webcam* [online]. Available from: https://www.trust.com/en/product/18679-trino-hd-video-webcam [Accessed 08 March 2020].

UMBAUGH, S., 2005. *Computer Imaging: Digital Image Analysis and Processing*. London: CRC.

UPTON, G., COOK, I., 2008. *A Dictionary of Statistics* [online]. Oxford University Press. Available from: https://www.oxfordreference.com/view/10.1093/acref/9780199541454.001.0001/acref-9780199541454 [Accessed 21 February 2020].

VINCENT, J., 2018. *Welcome to the automated warehouse of the future* [online]. Available from: https://www.theverge.com/2018/5/8/17331250/automated-warehouses-jobs-ocado-andover-amazon [Accessed 21 February 2020].

WATT, A., 2014. *Project Management*. 2nd ed. [online]. Available from: https://opentextbc.ca/projectmanagement/ [Accessed 07 March 2020].

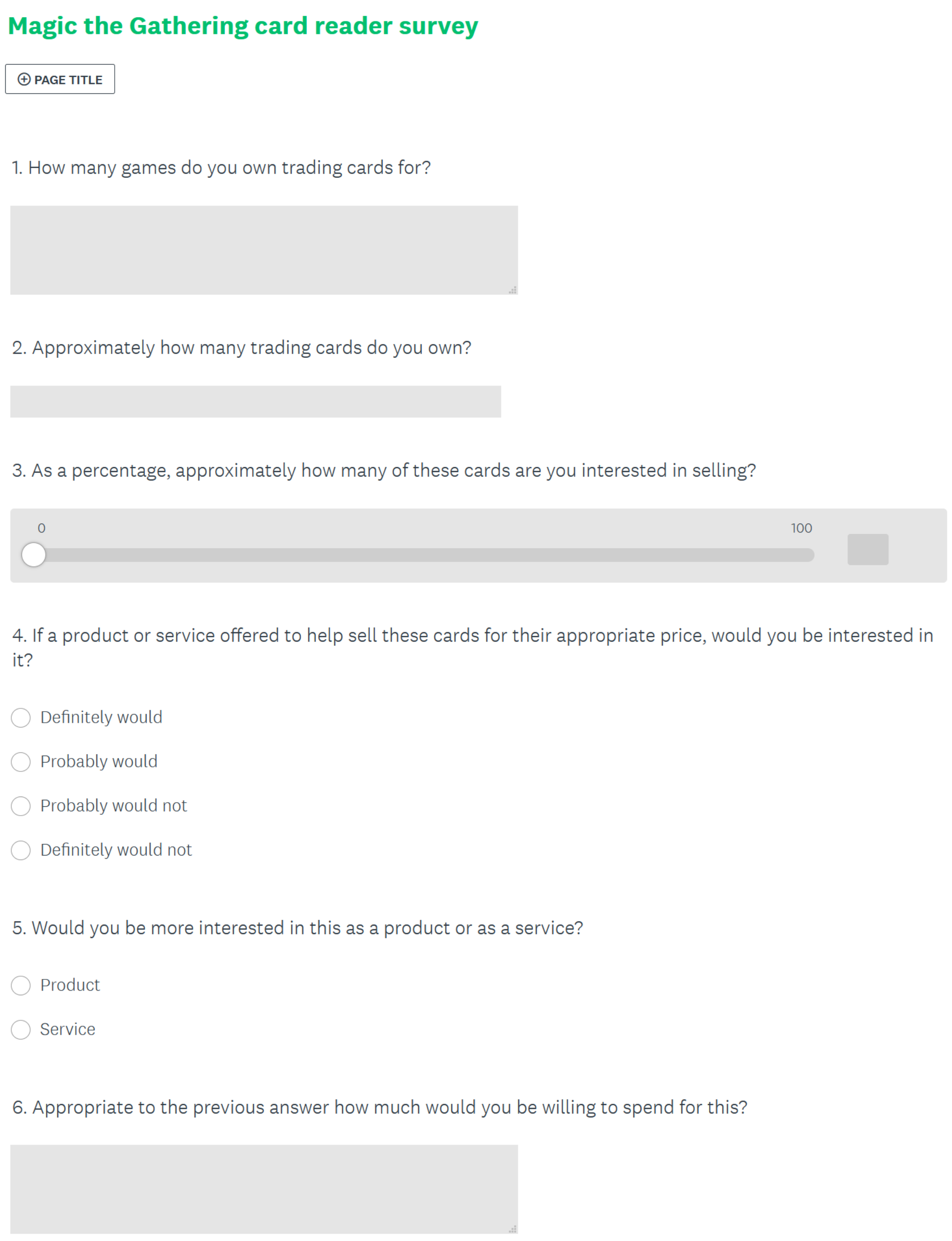
WIZARDS OF THE COAST, 2016. *Shadows Over Innistrad* [online]. Available from: https://magic.wizards.com/en/content/shadows-over-innistrad-cards [Accessed 09 March 2020].

WIZARDS OF THE COAST, 2020. *Magic: The Gathering* [online]. Available from: https://magic.wizards.com/ [Accessed 21 February 2020].

ZHANG, L., (2013) *Predicting Virtual Markets* [online] Available from: http://cs229.stanford.edu/proj2013/zhang-PredictingVirtualMarkets.pdf [Accessed 12 March 2020].

Appendix

i.



Magic the Gathering card reader survey

1. How many games do you own trading cards for?

2. Approximately how many trading cards do you own?

3. As a percentage, approximately how many of these cards are you interested in selling?

4. If a product or service offered to help sell these cards for their appropriate price, would you be interested in it?

- ○ Definitely would
- ○ Probably would
- ○ Probably would not
- ○ Definitely would not

5. Would you be more interested in this as a product or as a service?

- ○ Product
- ○ Service

6. Appropriate to the previous answer how much would you be willing to spend for this?