# Self-Scan Age Verification System

# Introduction

## Purpose of the System

*Joel Guest and Ben Coxford*

In modern day Britain, the law requires all registered retailers and services to refuse the sale of age-restricted products to someone who is younger than the minimum legal age.

“It is an offence to sell products to someone younger than the minimum legal age and for some products it is an offence for somebody below the legal age to try to buy them”

<https://www.cambridgeshire.gov.uk/business/trading-standards/age-restricted-products>

*(Cambridgeshire.gov.uk, 2020)*

Many retailers opt the challenge 25 procedure, so if the customer look under 25, they will be asked for ID. This procedure is in place to prevent the sale of age-restricted items to those who cannot prove they are of the minimum legal age.

This procedure is in place to prevent the increasing dilemma of the underage sales of age restricted items which on one hand can cause harm to the customer if used underage or illegally irresponsible. However, not following challenge 25 protocols or even wrongly selling to a minor as a result of fake ID can lead to large fines or even prosecutions for the company involved. The following document by the home office covers the legalities and guidance on the issue: <https://www.pass-scheme.org.uk/wp-content/uploads/2014/06/Home-Office_false-id-guidance_July-2012.pdf>

*(Pass-scheme.org.uk, 2012)*

In 2018, the discount-store B&M was fined £480,000 because the company had sold knives repeatedly to under-18s and some as young as 14. <https://www.bbc.co.uk/news/uk-england-london-45603915> *(England, 2018)*

Self-scan tills can be difficult when it comes to challenge 25, where customers must wait for approval for age-restricted products. A system is desired to reduce the wait time in a customer’s journey when being challenged to an age-restricted product, whilst maintaining the effectiveness of validating a customer’s age. The overall purpose of this system is to verify the customer's age and validate the ID of the customer in a self-scan scenario.

## Project Description & Deliverables

*Joel Guest and Ben Coxford*

To improve the customer journey and to combat the ever-increasing fraudulent use of fake ID’s we are proposing a program that utilises a range of detection methods which will ultimately authenticate the legitimacy of an EU driving licence. We will test for four main variables:

* **Face Detection and Matching:** We will detect the face of the customer and their ID and determine if the face is a match.
* **Text Detection:** We will extract the text in order to authenticate the date is legal for purchase and whether the expiry date of the card isn’t over.
* **UV Blob Detection:** By shining a UV light on the rear of the driving licence we can see a unique pattern that is one of the most reliable authentication methods for identifying real ID’s. We will integrate a variety of image-processing algorithms and match the unique shape of the watermark to a range of test data.



Face

Name

Address & Signature

Our objective is to deliver a system that can automatically validate a customer’s age and where it cannot, it must produce an intervention where a member of staff then must validate their age. By implementing this service, it should not improve the customers journey but also act as a deterrent in the use of fake ID’s.

Starting Week – 2nd December 2019

End Date – 17th January 2019

# Project Management and Tools

## Agile Development Approach

*Joel Guest*

As a group we aim to implement an agile software development strategy as both a way of managing tasks and progressing with an adaptive approach to programming *(Agile Alliance, 2019*). We aim to implement a scrum framework with an alternating weekly scrum master and a scrum meeting every Monday afternoon between 13:00 and 17:00. The scrum master will add to a weekly Gantt chart with the tasks for everyone for that week.

## Group Work Declarative Statement

*Joel Guest, Ben Coxford and Jack Templeman*

Terms agreed by the group;

* We agree to participate in every aspect in the development and documentation of the system and project.
* We agree to meet every Monday for weekly scrum meetings and if meetings cannot be done in person, they are carried out over Discord.
* We agree to update and declare individual contributions to the documentation and code.
* We agree to log and upload any work, completed or not, to GitLab and GoogleDrive.

Sign below in the given space below.

Joel guest



Ben Coxford



Jack Templeman



## Task Structure

*Joel Guest*

Task structure within software development refers to the degree to which the task is made clear to the employee who must perform it. Task structure includes the extent to which tasks are defined and have detailed job descriptions and procedures. *(Smallbusiness.chron.com, 2019)*

**Our methods of implementing task structure:**

**Gantt chart:** the Gantt chart will document the tasks for everyone for that week with the timescale of the task displayed in the chart. The last used Gantt chart is shown below.

**Gantt Chart Template:** *(Wittwer, 2019)*

**Sprint documents & Backlog:** We will add our sprint tasks into a group Trello form where we can assign tasks and timings for our weekly sprint tasks to ensure they are completed. Google drive will be used to store any other sprint documents. An example of Trello *(Trello.com, 2020)* usage can be seen in Appendix A.

**Scrum master:** The scrum master will be the individual who, on a weekly basis will be in charge of assigning realistic tasks for that specific week and at the end of the week will be in charge of the following weeks sprint tasks, assignment of roles and sprint documentation.

A screenshot of a social media post

Description automatically generated

## Hardware and Development Environment

*Joel Guest and Ben Coxford*

For the coding and software development side of the program we will be using the Python programming language. Python has existing support for using external hardware within a MAC’s Linux based operating system. For the hardware in the programme we will be using both the MAC’s internal webcam and an external webcam to take simultaneous images. Also, for the ID verification we will require a UV light to view the cards unique watermark.

For the management and task structure side of the development we will use word, excel and google drive for our documentation and discord for communication. Within word we will write both this document and our own personal progress logs. Excel and google drive will be used for the gantt chart and sprint documentation respectively. Discord will be used to communicate progress and general questions within the group.

To store the program code, we will use version control Git *(GitLab, 2020)* to manage the versions. Version control allows collaboration when programming a project. Additionally, it stores various versions of the project without the project becoming tedious and difficult to deal with. This way we can manage the code and update it throughout the group. Another benefit is if we are not happy with current progress, we can revert to a previous version without having to backtrack and it acts as a back-up in-case data is lost on our personal computers.

# System Requirements

*Joel Guest*

## Functional Requirements

Live Video:

One of the functional requirements of the system includes a live video of both the users face and ID simultaneously. From this we will be able to compare facial Features in order to recognise if the ID belongs to them. *(Docs.opencv.org, 2020)*

Face recognition:

We will require facial recognition for the above element of live video. In order to compare the users, live face with the face on their ID. This will be the first step in identifying the ID they carry actually belongs to them. *(Docs.opencv.org, 2020)*

Text Recognition:

To test the age of the ID holder we will detect the Birth date to clarify they are over 18. We will also detect the issue date and expiry date to discern whether the ID is valid and in date. *(Rosebrock, 2018)*

Image Processing:

In order to format the image in a way the computer can extract information, image processing must be utilised. in the case of this program Binary Threshold was used in order to extract the text, face and UV watermark form the driving licence. *(Docs.opencv.org, 2020)*

Blob detection and comparison:

The blob detection will ensure the ID is not fake by scanning the unique UV watermark on the back of the card and comparing it to test data of real ID watermarks to detect if the card is real. *(Docs.opencv.org, 2020)*

Hardware:

The required hardware for this system to work will be:

* Camera
* Webcam
* Light Source
* UV Light

Final Solution

*Ben Coxford*

Within the system, the program must take several images as an input and be able to produce a set of results. Following the pipeline, the faces must match, the dates must be valid and in date and the feature matching must produce a reasonable result.

## Non-Functional Requirements

*Jack Templeman*

Reliability:

The reliability of the detection matches is to be near perfect, users must be able to trust the code to absolutely identify any fake or invalid identifications. If any of the tests give a false positive and the ID is allowed to pass, the user will be liable to the offence of selling products to persons of younger age than is required to purchase that item.

Usability:

The end product needs to be extremely easy and efficient to use to be proven as useful. If the checks are extended and long, and the physical camera setup is difficult to use, it will be rendered useless as the user is likely to just manually check the ID instead.

Regulatory:

For legal reasons, the blob and other such comparisons need to remain updated to current standards to ensure all IDs checked are still genuine. This again will stop any guilt being placed on the user, even in the future.

## Domain Requirements

*Ben Coxford*

* Test data for the UV feature matching can be initially created to begin learning.
* The time taken to complete the checks should be under a minute; however, this is not vital for the prototype system.
* Due to data protection laws, no data should be stored on an individual. Only the processed, binary form hologram on the back of the card can be stored.
* Safety must be ensured when using the UV light. Must not look directly into the UV light and eye-wear protection must be used when developing the device.

## Requirements Validation

*Jack Templeman*

Prototype validation will best be able to validate the requirements of the specification. As the program requires hardware such as cameras and UV lights to function, it will best show the product to potential customers. The criteria of the original functional and non-functional requirements will be investigated using an assembled prototype of the project. This will be tested in live use by the clients to gather feedback information about the functionality of all requirements specified.

The focus of these results will be around the reliability and usability of the function requirements, these are by far the most important factors as to whether the project is viable to be used by clients.

The project was presented to 10 potential focus group personal to test the aforementioned aspects as a mark out of 10. The results, averaged, are as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Live Video | Facial Recognition | Text Recognition | Image Processing | Blob detection |
| Reliability | 9.7 | 9.5 | 9.6 | 9.8 | 9.8 |
| Usability | 8.7 | 9.2 | 10.0 | 10 | 8.1 |

General comments were also recorded where marks were significantly lower to identify specifically where the problems lie.

Facial recognition was noted to be inconsistent on people with a large age gap from their original license photograph, as the program largely scans facial structure, if this changes a match will not be made.

The usability of the live video was noted to be too low on framerate, making it difficult and frustrating to use.

Complaints were also registered regarding the usability of the blob detection system implemented as the ID required flipping to access the blob on the back of the card making the system take twice as long as possibly necessary. Some reported difficulty with over exposure of the UV light making the image processing match significantly less than normal.

This prototyping has identified not only the significant areas of problems, but it has also showed that several areas are working to an outstanding level. The reliability and usability of the facial/text recognition and image processing are exactly where customers would like it to be, comments such as “effortless” and “works without issue” were noted with text recognition and image processing scoring perfect scores on usability.

Problems are identified as follows. Increased refresh rate for the webcam feed, possible use of the blob on the front of ID to stop the user needing to flip the card, more stable reliable positioning of the UV light to stop over exposure.

# Software Development

## Design Pattern

## Object Pool

*Joel Guest*

A client with access to an Object pool can avoid creating a new Objects by simply asking the pool for one that has already been instantiated instead. This increases performance as we won’t require to initialise a new class each time. *(Sourcemaking.com, 2020)*

Within the code for this project. Object pooling is used within an image pool file which stores images as objects which can then be used within the project code file for simple machine learning to create a better average when feature matching the UV watermarks.

## Facade Pattern

*Ben Coxford*

The facade design pattern was a clear choice to use because the system has multiple complex features to implement. This allows the system itself to be developed focusing on one feature (subsystem) at a time to produce a complete system.

The design pattern focuses on one wrapper class to encapsulate the subsystem where the client and interface interfaces only with the facade and not directly to a subsystem. In this scenario the facade is also classed as a singleton where only one facade object is created.

This allows each subsystem to be independent from other subsystems and allows the facade to manage them.

## Why use them together?

Facade, coupled with the object pool, allows minimal objects to be instantiated in the system. This is beneficial for the computer system used because the program uses less memory as a result of limiting the number of objects. Object pool enables an object to be recycled and as a result of this the images are stored as of when they are needed and released back to the pool when it is finished. This is beneficial for a system which has to process a vast number of images.

## Architectural Design

*Ben Coxford*

When researching various image processing methods, we came across three ideas to determine the validity and authenticity of a European driver’s license.

1. Date of Birth, Expiry Date and Issue Date Detection and Validation.
2. Face matching the ID cards face to the customers face.
3. Feature matching the id cards mark under UV light against a data set.

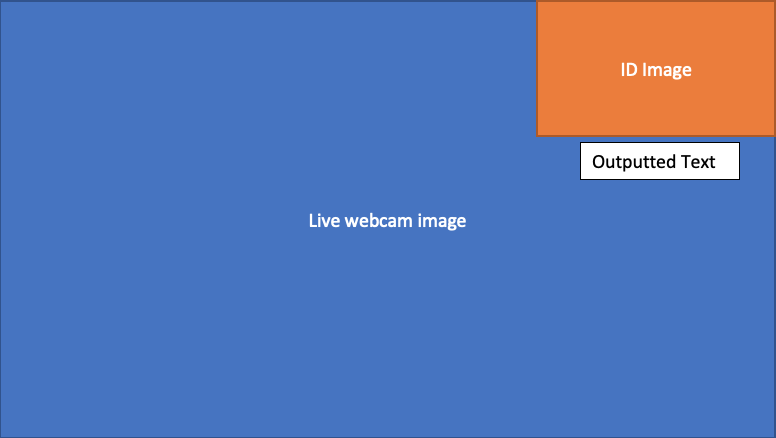
This creates a pipeline of events to take place to produce three possible solutions – it is valid, it is not valid, requires a staff members attention.

Customer Finishes Scanning -> System Starts -> Prompts customer to place card under the reader -> Attempts to match the customers face to the id card and produces outcome **A** -> Attempts to process the dates, and produces outcome **B** -> Prompts customer to turn over the ID card -> Under UV Light Conditions, scans and processes the mark and feature matches it to a data set to produce outcome **C** -> Determines whether accepted or whether the customer requires assistance -> System Ends -> Customer Pays.

## User-Interface

*Ben Coxford*

The user interface consists of a keyboard, a display and two cameras. The user must use the keyboard to interrupt the program to notify that the ID has been turned over. The output is displayed, and an example frame is shown below.



## System Photos

## An open computer sitting on a table Description automatically generated

## A computer sitting on a table Description automatically generated

## A computer sitting on a table Description automatically generated

## An open computer sitting on a table Description automatically generated

## A person using a computer sitting on top of a table Description automatically generated

## A computer sitting on a table Description automatically generated

## A person smiling for the camera Description automatically generated

## Testing

*Joel Guest*

An important step required throughout the coding progress is testing. Where necessary a unit test must be added to the unit testing python file to test that the specific classes within each part of the program were working as expected.

The first scrum week which required unit testing was week 4. Within this week unit testing for object pool, camera and face detection subsystem were written. The object pool unit test tests for if the image acquired is the same as the image returned. The camera unit test comprised of two simple unit tests which tested if the image could be read from the camera subsystem as a whole.

In week 5 the unit tests face detection and date subsystem were required. The face detection unit test detected whether two faces had been detected from both the live image of the individuals face and the still image of the face on the ID card. The date unit test firstly tests if 3 dates have been detected, then if these dates are changed to true to verify, they are valid dates.

The unit test was finalised in week 6. In week 6 the unit tests for blob match featuring and training subsystem were. the blob match unit test took the two UV watermarks from the test data to verify if the number of matches in the feature matching was within our given array. Errors within the matching will come from invalid images which won’t have a sufficient number of matches. The training subsystem will simply test for the number of matches and the mean from each matching value.

The testing code is visible in **Appendix B** in the **testing** section.

# Work Breakdown:

*Joel Guest, Ben Coxford, Jack Templeman*

|  |  |  |
| --- | --- | --- |
| TASK | SCRUM Master | Completed By |
| Plan scrum meeting time and contract | Ben | Group |
| Set out scrum master rota | Ben | Group |
| Decide on final project idea | Ben | Group |
| Create work log template | Ben | Group |
| SCRUM Meeting 1 | Jack | Group |
| Setup Google Drive and Trello | Jack | Joel & Jack |
| Set up Gantt Chart | Jack | Ben & Joel |
| Setup hardware and setup prototype | Jack | Joel & Ben |
| Document System Proposal | Jack | Joel |
| Setup git lab repository | Jack | Ben |
| SCRUM Meeting 2 | Joel | Group |
| Research Text detection | Joel | Joel |
| Research Face detection | Joel | Jack |
| Research blob detection and comparison | Joel | Ben |
| Project deliverables & add contract to document | Joel | Joel |
| Add backlog to Trello and update gantt chart | Joel | Joel |
| SCRUM Meeting 3 | Ben | Group |
| Identify design patterns | Ben | Ben & Joel |
| Setup final hardware | Ben | Ben |
| Test UV Light Box and create image processing subsystem | Ben | Ben |
| System Requirements | Ben | Joel |
| Non-Functional Requirements | Ben | Jack |
| Domain Requirements | Ben | Ben |
| Requirements Validation | Ben | Jack |
| Add backlog to Trello and update gantt chart | Ben | Group |
| SCRUM Meeting 4 | Joel | Group |
| Update main system | Joel | Ben |
| Create facade pattern | Joel | Ben |
| Create object (image) pool pattern | Joel | Joel |
| Setup camera sub system | Joel | Ben |
| Set up empty subsystems and layout | Joel | Ben |
| Set up face detection subsystem | Joel | Ben |
| Update main system | Joel | Ben |
| Display customer view of both ID's | Joel | Ben |
| Unit test object pool, camera and face detection subsystem | Joel | Joel |
| Add backlog to Trello and update gantt chart | Joel | Joel |
| SCRUM Meeting 5 | Ben | Group |
| Finish face detection subsystem and match faces | Ben | Ben |
| Complete date and text detection system | Ben | Ben & Joel |
| Unit test face detection and date subsystem | Ben | Joel |
| Start blob match featuring | Ben | Ben |
| Add backlog to Trello and update gantt chart | Joel | Joel |
| SCRUM Meeting 6 | Joel | Group |
| Finish blob match featuring and training subsystem | Joel | Ben & Joel |
| Unit test blob match featuring and training subsystem | Joel | Joel |
| Final unit testing and system testing | Joel | Ben |
| Finalise documentation - maintenance, implementation | Joel | Group |
| Upload zipped documents and files | Joel | Joel |

## 

## Work Breakdown Sign

## Sign below to agree to the contributions of work throughout the entire project. This is outlined by the column ‘Completed By’ in the table above.

Joel guest



Ben Coxford



Jack Templeman



# Conclusions

## Improvements

*Joel Guest, Ben Coxford*

The system itself can be improved drastically in relation to the user’s interface. Ideally, a touch screen would be used along with a clearer output. Additionally, this would be implemented around a self-scan system with staff interrupts and customer help-requests.

To improve the accuracy and reliability of the program further. The program could simply be run more times. This would increase the number of saved images in the matching folder and as this is a learning system, this would increase the potential for successful matches as the mean average for the number of matches would be refined.

In order to improve the effectiveness of the system, more detection features could be added such as checking the UV marks on the opposite side and potentially using CNN networks that can predict the output

## Group overview

Ideally, Jack could have taken the role of SCRUM master a few more times and engaged more with the project and produced some program code. An improvement for Ben would be to focus on additional, more complex features and to look at how more design patterns could benefit the system. Although Joel carried out unit testing and contributed to parts of the code. His contribution to the coding could have been improved further.

As a group communication was difficult to maintain with being apart for long periods over Christmas. This compromised our ability to meet in person and resulted in communication over Discord.

As a group our project management regarding the tools and documents was strong. This allowed the project to be developed without any setbacks and as a result the groups deadline was met on time. The complexity of the project shows how the use of design patterns and pythons’ vast range of libraries can produce a system within a short deadline.

Lastly, our management of time to complete a project of this complexity including a full group documentation and personal logs has been a very positive attribute.

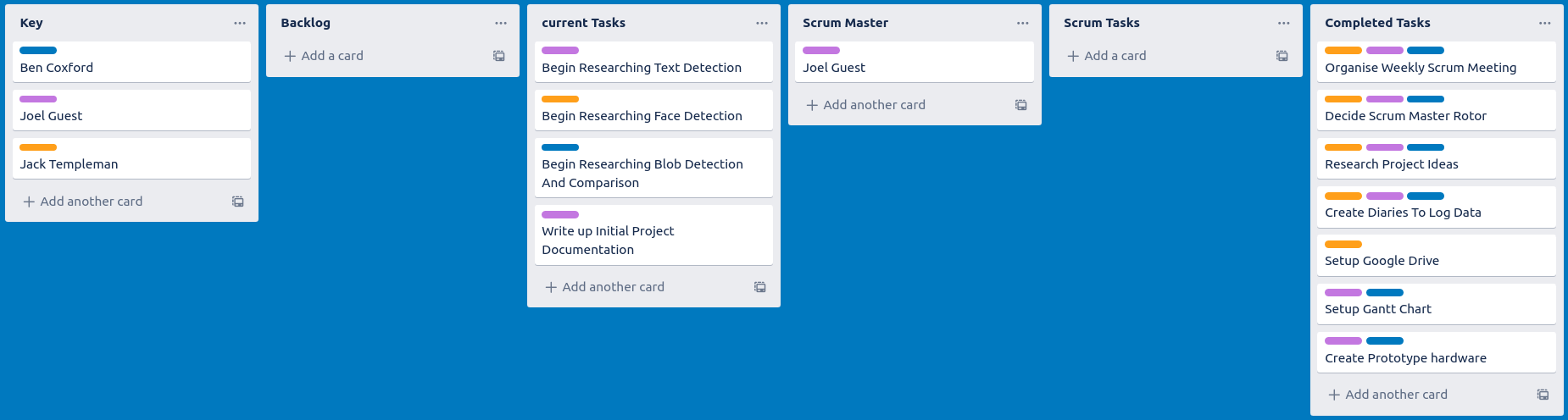
# Appendices

*Joel Guest*

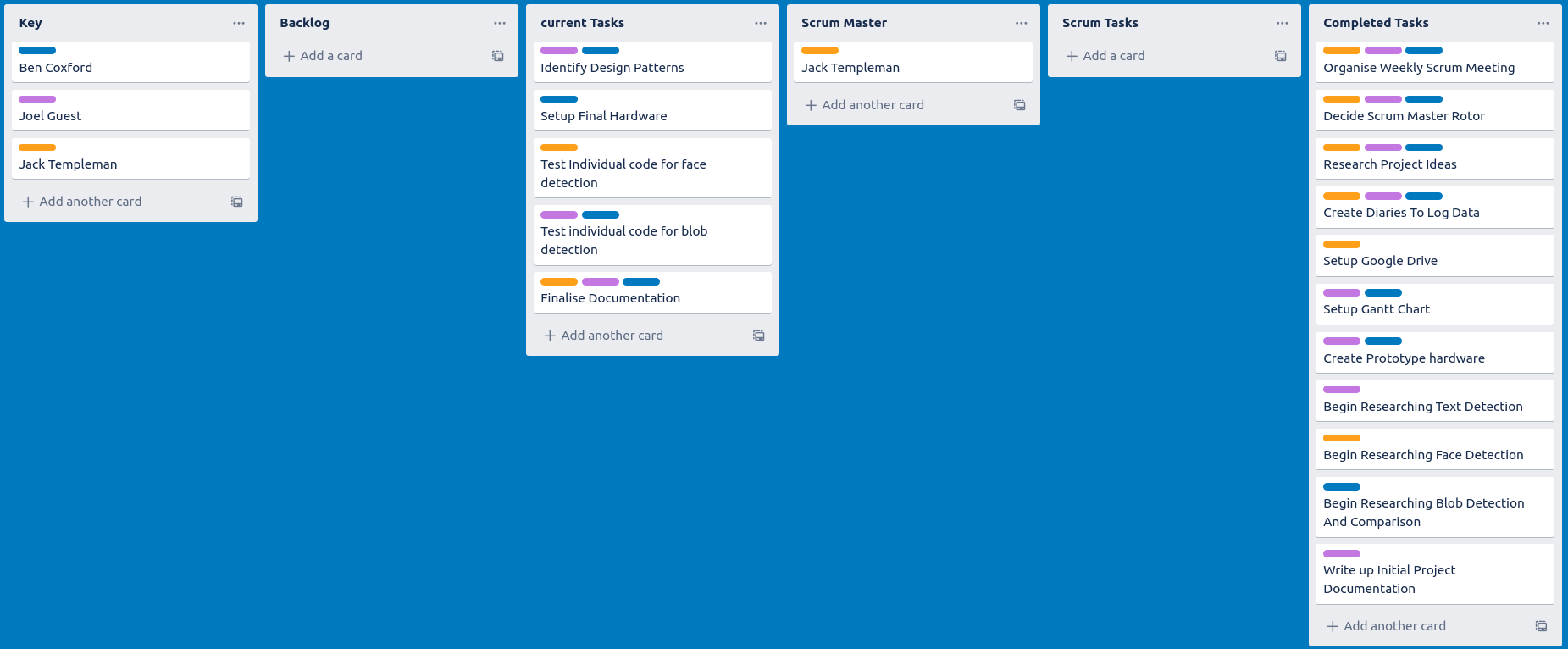
# Appendix A

# Week 1:

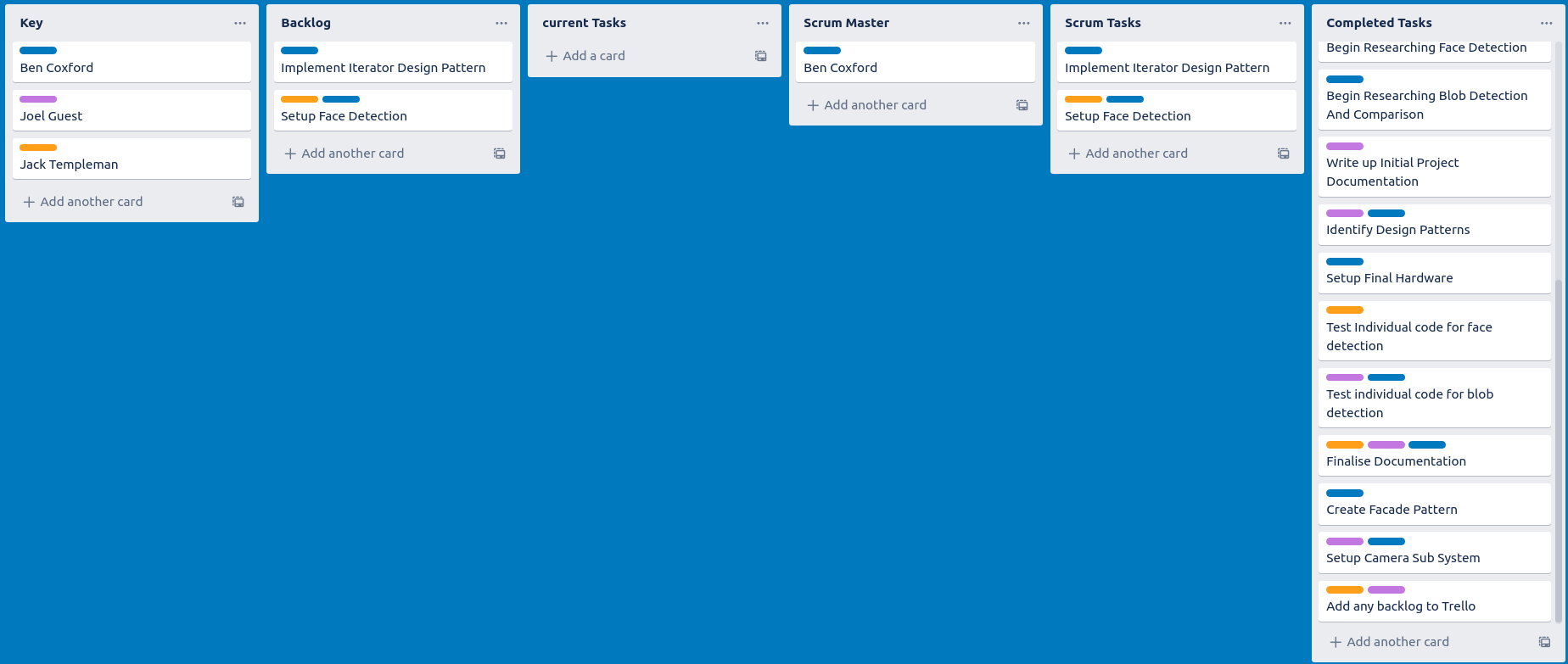
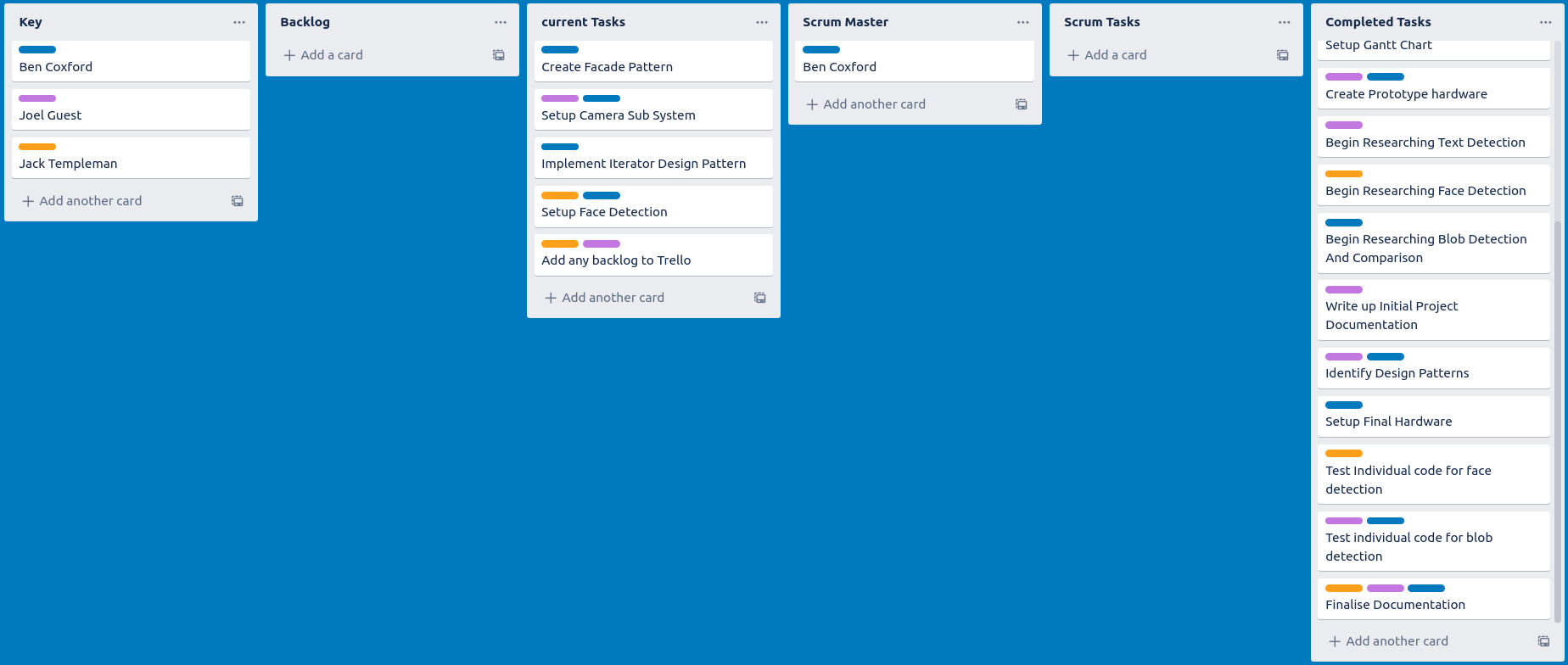
# Week 2:



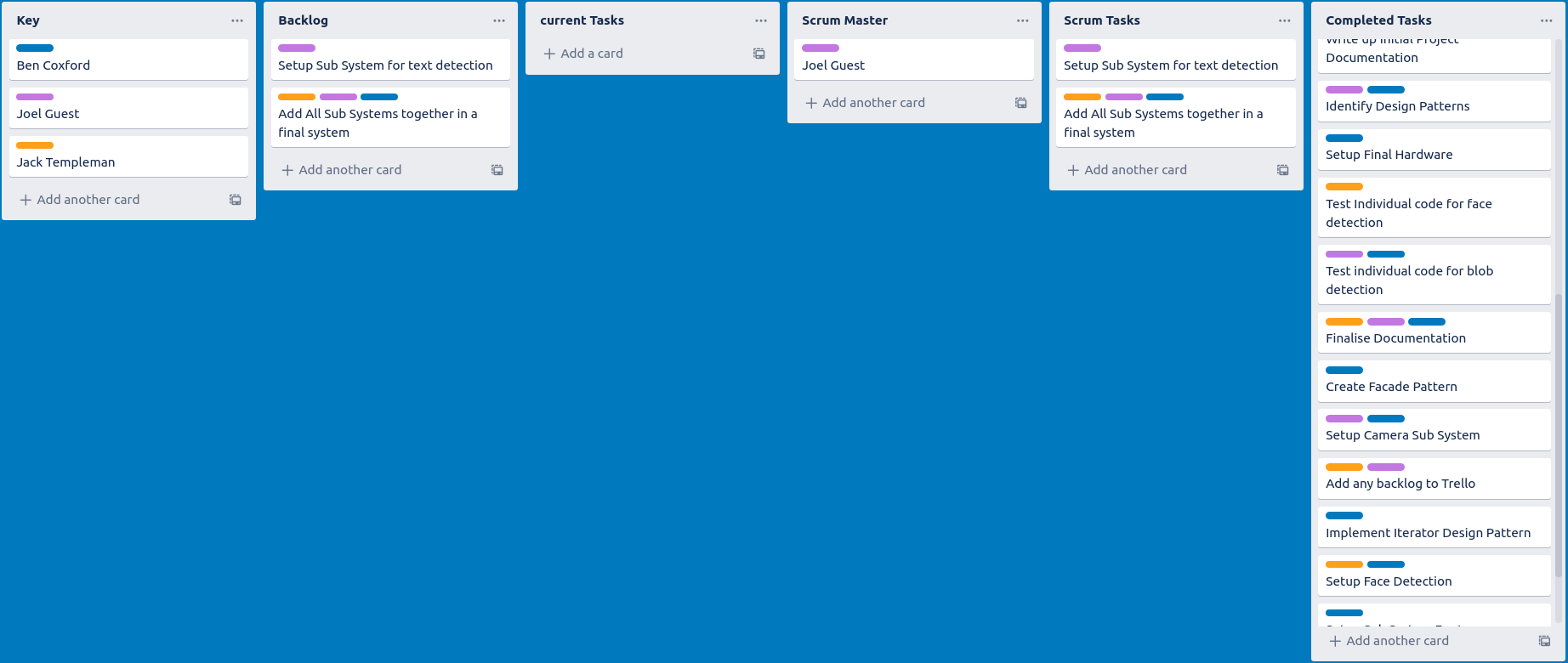
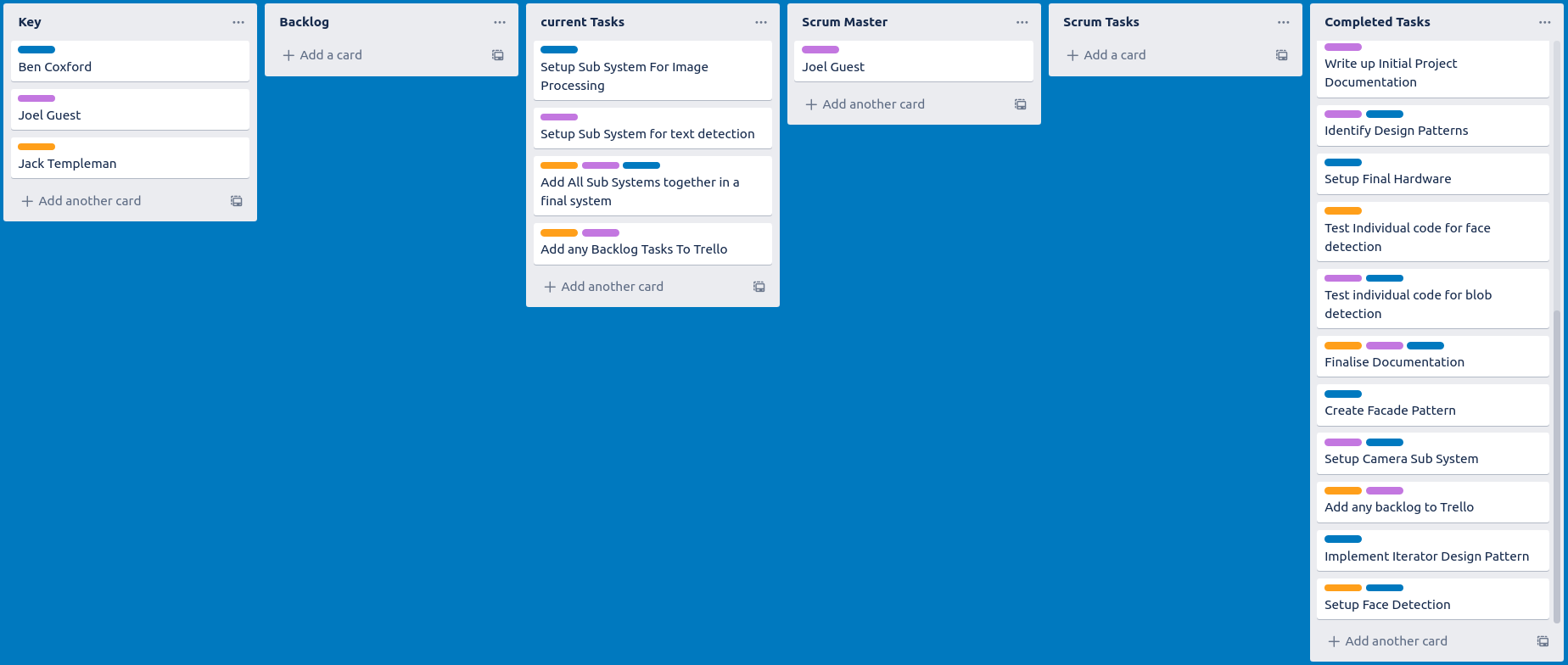
# Week 3:



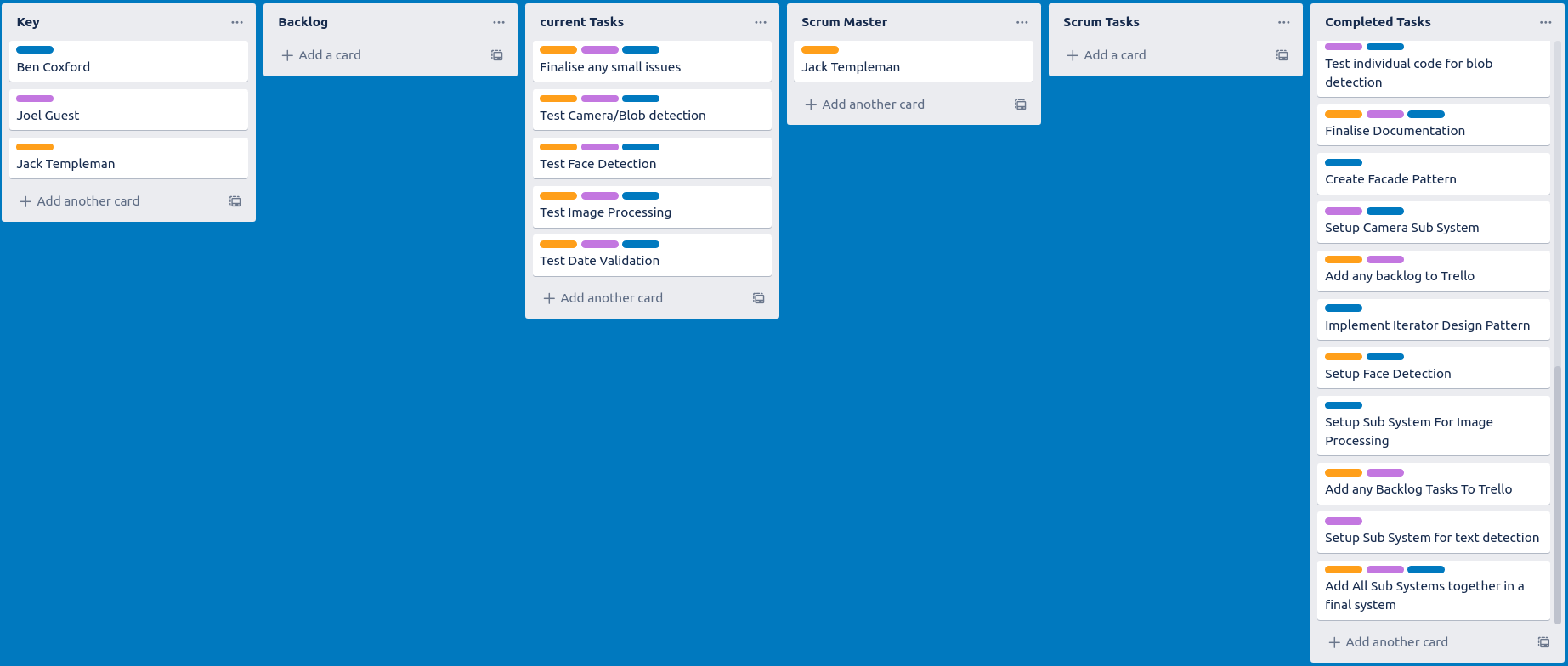
# Week 4:



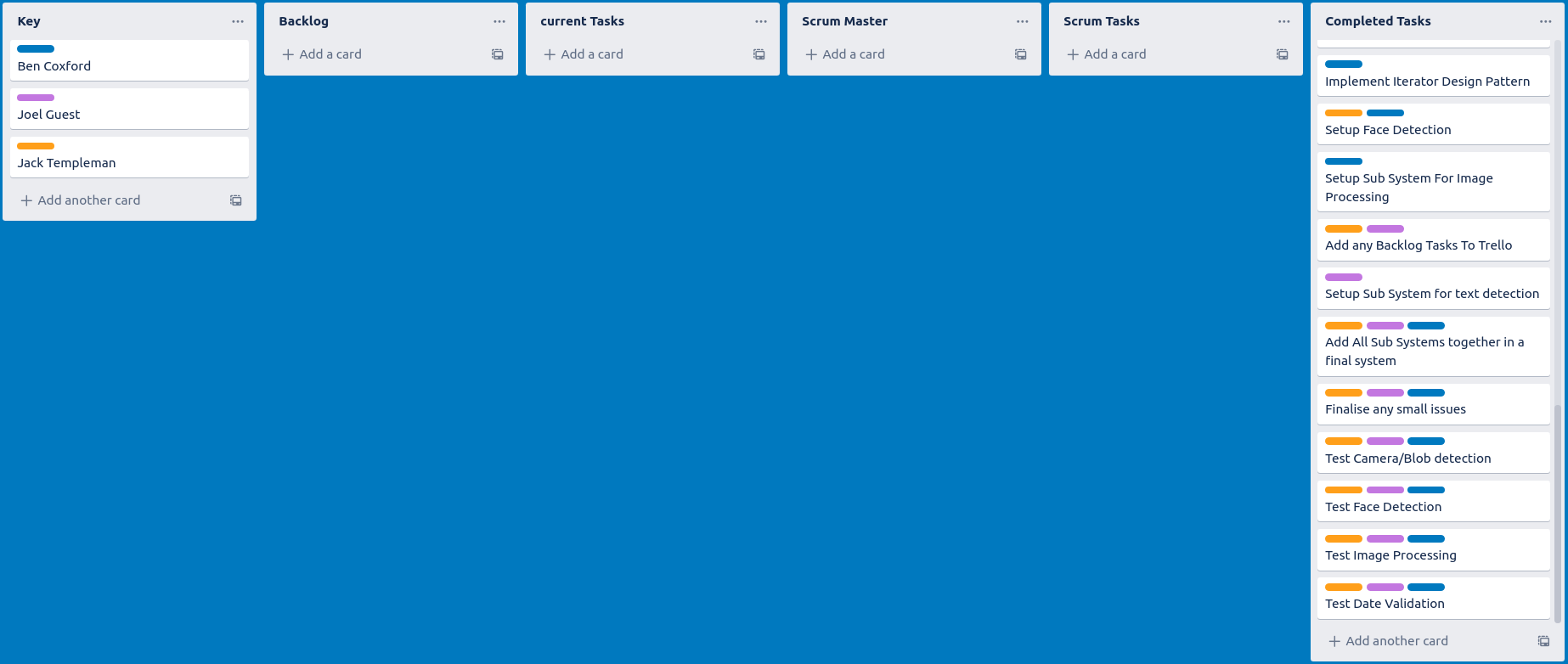
# Week 5:



# Week 6:



# Final:



## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

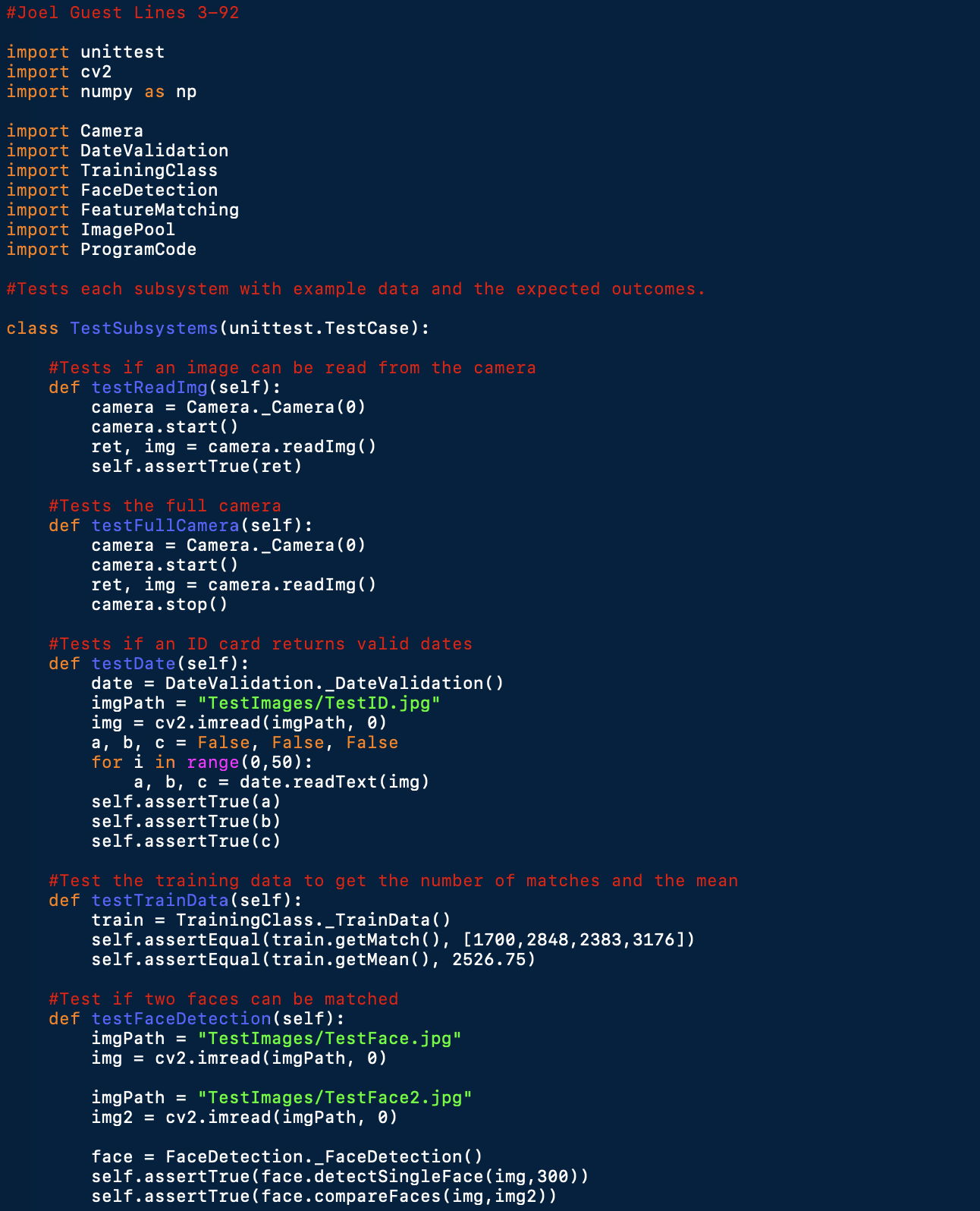
## 

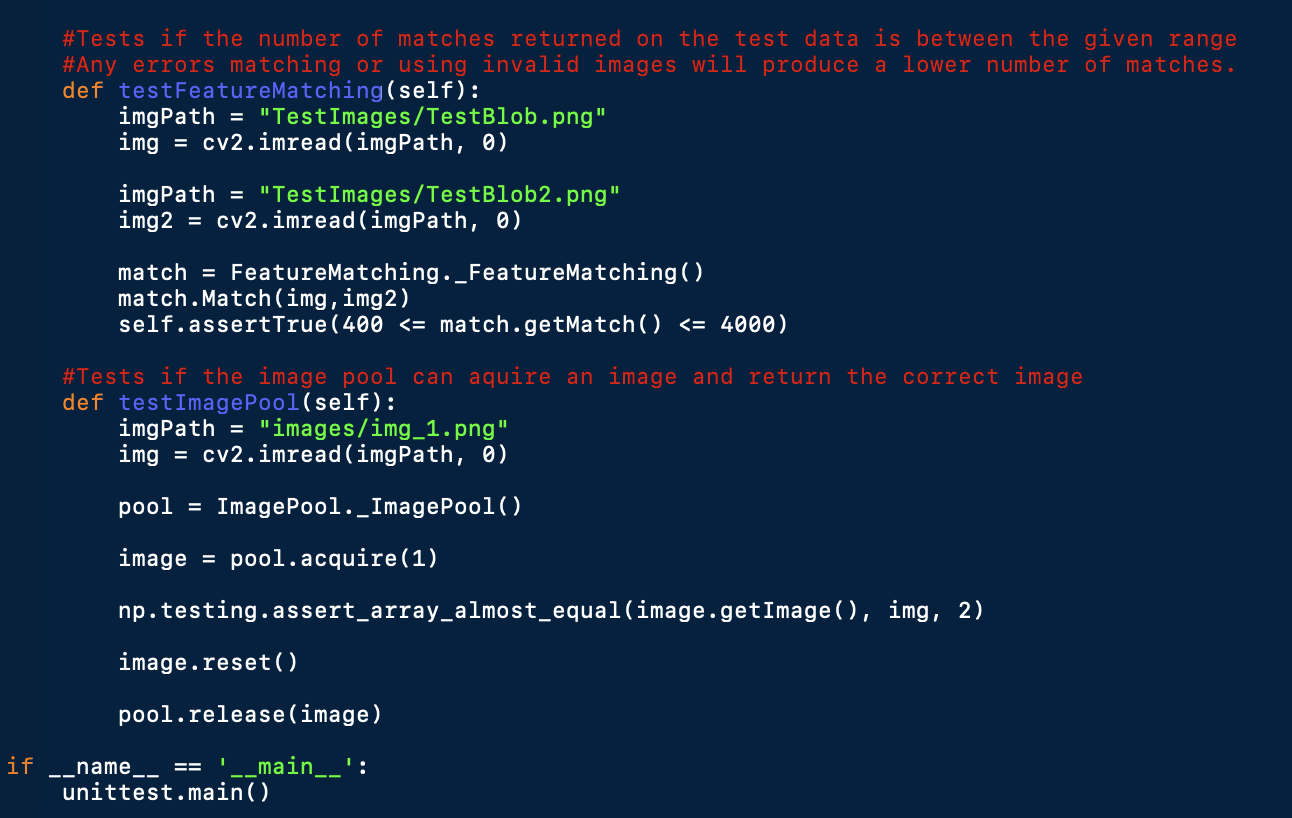
## 

## 

## Appendix B

### Testing:





# Bibliography

**Agile alliance website:** *Agile Alliance. (2019). What is Agile Software Development?. [online] Available at: https://www.agilealliance.org/agile101/ [Accessed 10 Dec. 2019].*

**BBC Reference:** *Bbc.co.uk. (2005). BBC Inside Out Extra - Fake ID cards. [online] Available at: http://www.bbc.co.uk/insideout/extra/series2/teenage-drinking.shtml [Accessed 26 Nov. 2019].*

**Binary Threshold:** *Docs.opencv.org. (2020). OpenCV: Image Thresholding. [online] Available at: https://docs.opencv.org/3.4/d7/d4d/tutorial\_py\_thresholding.html [Accessed 13 Jan. 2020].*

**Cambridgeshire Council:** *Cambridgeshire.gov.uk. (2020). Age restricted products - Cambridgeshire County Council. [online] Available at: https://www.cambridgeshire.gov.uk/business/trading-standards/age-restricted-products [Accessed 13 Jan. 2020]*

***Facade:*** *Sourcemaking.com. (2020). Façade Design Pattern. [online] Available at: https://sourcemaking.com/design\_patterns/facade [Accessed 13 Jan. 2020].*

***Face Detection:*** *github.com. (2019). Face Recognition. [online] Available at: https://github.com/ageitgey/face\_recognition [Accessed 17 Dec. 2019].*

**Gantt Chart Template:** *Wittwer, J. (2019). Gantt Chart Template for Excel. [online] Vertex42.com. Available at: https://www.vertex42.com/ExcelTemplates/excel-gantt-chart.html [Accessed 13 Jan. 2020]*

**Home Office Document:** *Pass-scheme.org.uk. (2012). [online] Available at: https://www.pass-scheme.org.uk/wp-content/uploads/2014/06/Home-Office\_false-id-guidance\_July-2012.pdf [Accessed 26 Nov. 2019].*

***Object Pool:*** *Sourcemaking.com. (2020). Design Patterns and Refactoring. [online] Available at: https://sourcemaking.com/design\_patterns/object\_pool [Accessed 13 Jan. 2020].*

**OpenCV Feature Matching:** *Docs.opencv.org. (2020). OpenCV: Feature Matching. [online] Available at: https://docs.opencv.org/master/dc/dc3/tutorial\_py\_matcher.html [Accessed 13 Jan. 2020].*

**Task structure website:** *Smallbusiness.chron.com. (2019). What Is a Task Structure in the Path-Goal Leadership Theory?. [online] Available at: https://smallbusiness.chron.com/task-structure-pathgoal-leadership-theory-65396.html [Accessed 17 Dec. 2019].*

**Text Detection:** *Rosebrock, A. (2018). OpenCV OCR and text recognition with Tesseract - PyImageSearch. [online] PyImageSearch. Available at: https://www.pyimagesearch.com/2018/09/17/opencv-ocr-and-text-recognition-with-tesseract/ [Accessed 13 Jan. 2020].*

**Trello:** *Trello.com. (2020). Trello. [online] Available at: https://trello.com/en [Accessed 13 Jan. 2020].*