# MapMyNotes

Final Report for CS39440 Major Project

*Author:* Ryan Gouldsmith (ryg1@aber.ac.uk)
*Supervisor:* Dr. Hannah Dee (hmd1@aber.ac.uk)

4th March 2016
Version: 1.0 (Draft)

This report was submitted as partial fulfilment of a BSc degree in Computer Science (G401)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

## Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name ..........................................................

Date ..........................................................

## Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name ..........................................................

Date ..........................................................

## Acknowledgements

I am grateful to...

I'd like to thank...

# Abstract

Include an abstract for your project. This should be no more than 300 words.

# CONTENTS

# LIST OF FIGURES

## LIST OF TABLES

# Chapter 1

# Background & Objectives

## 1.1 Background

Handwriting notes is still considered to be an important aspect of note taking. Smoker et al [14] conducted a study comparing handwritten text against digital text for memory retention and out of 61 adults 72.1% preferred to take notes using pen and paper, rather than on a computer. Smoker et al concluded that recollection rates for handwritten text was greater than that of typed text proving that handwritten notes are better for a user's memory retention.

Technology has advanced and people are being more connected with software in the cloud as well as tracking things in their life digitally; Google Calendar is an example of this. Therefore, there's a need to ensure that memory retention with handwritten notes is carried forward into the digital age.

### 1.1.1 Taxonomy of notes

When notes are made they will often be very different from any other note. Some are semi-structured and some are back of the envelope kind of notes. When thinking about an application to analyse notes, first there has to be consideration for what a note will consist of. A taxonomy, by definition, is a biological term for a classification of similar sections, showing how things are linked together [18].

Notes can be thought of as a collection of similar classifications, whether this is the pure textual descriptions of a note or whether this is purely pictorial form or a mixture of both. However, the notes are normally split into three distinct categories:

1. Textual descriptions
2. Diagrams
3. Graphs

In figure 1.1, it shows a taxonomy of the different aspects which may form a part of a note. Textural descriptions form the core content of a note, this is essentially the important aspect that a note-taker is trying to remember and write down. Different note-takers form their notes in different

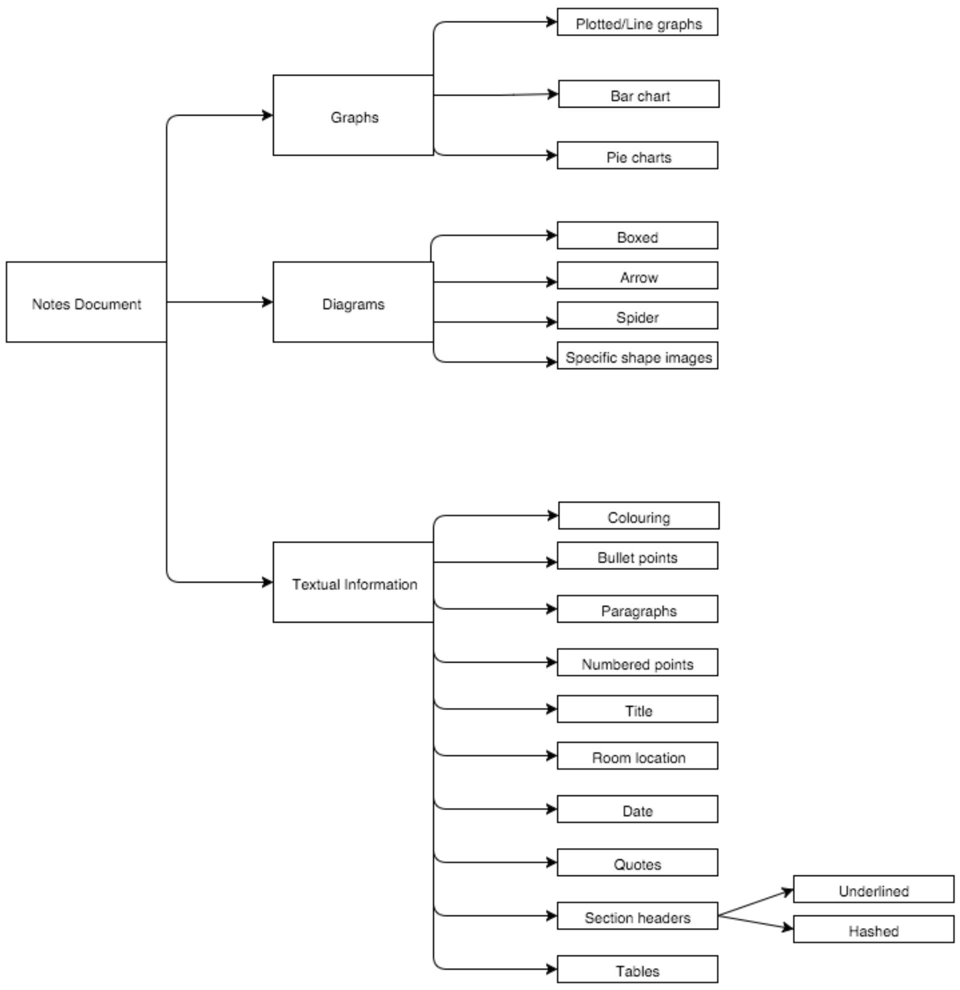

Figure 1.1: A taxonomy showing the structure and classification of different types of notes and what is contained in a note.

ways, for example the headings may be underlined or hashed - if they were adopting a mark-down style approach. These sections helps to show that there's a break in the content, and should be sub-sectioned. Textural points that are short, but important, are often characterised by a colon or a bullet point; these are the most common form of concise note building, in the classification.

Coloured text is often used for a variety of reasons: it stands out on the page and improves memory retention of that text [3]. Both congruent and incongruent coloured text helped to increase memory retention of post-graduate learners [11]. With congruent text Olurinola et al, on 20 students a 10.3 mean retention rate was achieved and 8.1 mean retention rate was achieved. The studies conducted show that coloured items would improve a user's retention rate in a lecture.

Finally, tables help to represent textual content in tabular form - this is often good in notes for comparisons.

Graphs are great visual tools for users to help to convey important textural information easily. Naturally, they have their limitations such as they come in different shapes and sizes, such as a line-graph, pie chart or a bar chart. Coupled with graphs, notes often consist of diagram drawings. In figure 1.1, there are different sections and classifications of a diagram: boxed, arrow etc. Each one has its own purpose and arrowed and box can overlap; UML diagrams are a case of this. Spider diagrams are probably the hardest to represent, due to the varying sizes and whether the user draws circles or clouds. Furthermore, specific shape diagrams are conceptually hard to think about as it depends on the domain in which the user is drawing the note. For example, a person in biology may draw a stick person, whereas someone in Computer Science may draw a PC.

Identifying a taxonomy of notes is imperative when considering what to parse from a note as it helps to define a domain of possible classifications. By identifying the classifications it will acknowledge what sections can be parsed by a specific technique. For example, textural information and bullet point lists can be parsed via text-recognition however, for diagram recognition that would involve image manipulation.

### 1.1.2 Similar Systems

With note-taking on digital devices becoming more widely available, there has become an influx in note-taking and organisational applications available for users. These are predominately WYSIWYG (What you see is what you get) editors - which a great deal of flexibility. When evaluating existing systems three were commonly used and they were:

- OneNote
- EverNote
- Google Keep

#### 1.1.2.1 OneNote

OneNote is a note-taking and organisational application made by Microsoft [8], offering the functionality to to add text, photos, drag and drop photos to a plain canvas. In recent times, OneNote has developed functionality to analyse a user's handwriting, from say a stylus, and interpret the text they entered [10]. In OneNote you can insert a note into document and then it would interpret the text from the note.

There is a wide range of product support from mobile based applications to web versions of their software. Office Lens [9] can be used in conjunction with the OneNote to help to take photos and automatically crop the image and then save them to OneNote. This feature is important and should be considered for the *MapMyNotes* application. The process requires you to have a Microsoft account, which you sign in with. When creating notes, OneNote formats collated notes into a series of "notebooks".

One feature which was nice when analysing the system was noticing automatic saving of the note, reducing the need for a user to click save. Additionally, when using OneNote it feels very much like Microsoft Word - with the similar layout that gives most users a similar user experience feel with its intuitive WYSIWYG(What you see is what you get) editor.

#### 1.1.2.2 EverNote

EverNote [4] is a note-taking and organisional application, it is both supported as a web application, bespoke desktop application and mobile applications. EverNote is widely used and would provide a lot of the functionality a user would need to upload their note.

They have released development articles [1] stating that they are able to do OCR recognition on images. This would allow the user to upload an image outputting a list of potential words for the image found. Like OneNote, the notes are collated into Notebooks, offering a WYSIWG editor, giving you full control of the content that you type in. When uploading an image, to the web version, it gives you the option to edit the PDF and images, however it seems as though you have to download another application, specific to your platform, to do this functionality.

According to the website, it does to OCR recognition, however whilst using the web application there was no information regarding extracting of text from the application. Additionally, there seemed to be no way to save the note to a calendar item - only the option to send via a link.

#### 1.1.2.3 Google Keep

Google Keep [5] is a note taking application produced by Google with mobile and website support. Google Keep allows a user to attach an image to their note and attempt to extract the text from an image and save this in the body of the note. In addition, it allows you to tag a title, and add an associated body.

An important design feature is that it does not offer the support of a WYSIWG editor, it is a default text box. They have the option of a “remind me” feature, which will get synced to their calendar as a reminder - but there’s no easy way to add it to a calendar event.

Google keep seems as though it’s more suited for TODO lists and jotting down quick notes, rather than an archiving tool suitable for substantial note taking. Nevertheless, the tagging with labels is a nice feature and the filter by image is a smart tool; this only shows notes with specific images. The simplicity of the user interface and the ease in which text could be extract provides a great reference going forward.

These three existing products are widely used by the every day note taker. They have been developed to a high quality and give the user a full control experience of what their notes can consist of. The automatically cropping of an image is an important process and should be considered when for the application going forward. *MapMyNotes* aims to try and give the user full control of their lecture notes content, so that they can find their notes easily again.

EverNote’s text extraction may differ from *MapMyNotes* as it will intend to give a one to one comparison of the text, rather than a list of potential words.

After the analysis of the existing products there are certain aspects which would be regarded as necessary: a simple way to view the notes, a way to filter the notes and a simple UI which feels more like an application rather than a website.

### 1.1.3 Personal Interest

The author handwrites his notes during lectures and often are discarded in crowded notebooks until they are needed for an assignment or examination.

A calendar event is already stored for every lecture that the author goes to, so it would be nice if there was a way to associate each of the notes taken to that calendar event. This would ensure that all the information is located in one easy place that can be found again, instead of trawling through lots of paper and trying to find the content again. This would aid in reducing the chances of lost notes from paper slipping out of the notebook or pages being damaged due to rain or creases.

## 1.2 Analysis

As the project was originally proposed by Dr Harry Strange, a meeting was arranged to discuss the initial ideas that he wished the application would follow. It was here that it was highlighted that Dr Harry Strange wants to take a photo of his notes, archive them with specific data, make them searchable and integrate them with existing calendar entries he had for a given date.

### 1.2.1 Parsing a note

In conjunction with the information gathered a taxonomy of notes was collated, helping to deconstruct what a note consists of. Analysing the taxonomy produced a comprehensive breakdown of what could be parsed as text. After seeing that text formed a main component of a note the key efforts of the application would focus on parsing the text. Diagrams, graphs and images were considered when thinking of what should be extracted from an image, however this was placed as a task for the future. This required a task to investigate how to reliably extract the text from an image.

### 1.2.2 An OCR tool

Handwriting recognition has been an active research project for a while. There could have been the possibility of creating a bespoke handwriting recognition tool, using machine learning techniques, but that would distract from the actual problem which is this available tool to archive notes.

Therefore an OCR tool would have to be chosen to analyse the text. Choosing a sensible OCR tool with good recognition rates would be important - so a task was created to explore and look at possible solutions.

### 1.2.3 What to parse from the note

From research conducted into Google Keep it was clear that analysing the text would be a great aspect to include in the application. The real question is what should be parsed from the note? By looking at the overall structure of the application and what it entailed then it was agreed to just parse the note's associated meta-data: the title, lecturer, date and module code. Recalling that Google Keep parses all the text and EverNote gives a list of suggested words, it was decided that a tool would be developed to suggest the meta-data but it does not automatically tag the meta-data.

### 1.2.4 Structuring of notes

In conjunction with analysing what to parse a sensible structure would have to be applied to notes used in the application. A task to create and find a good set of rules would have to be collated to ensure that notes could be parsed confidently. This reduced the complexity of incorporating natural language processing in the application, which would be more than can be achieved in the timeframe.

### 1.2.5 OCR for the authors handwriting

After research into OCR technologies, such as Tesseract [16], it was established that analysing handwriting is a complicated process. Instead of trying to train it on a lot of dummy data, it would be trained to recognise the author's handwriting. A task was created to train the user's handwriting data and this would run throughout the duration of the project.

### 1.2.6 What platform is most suitable

During the meeting with Dr Harry Strange one of the core features that was needed was for the application to be accessible regardless where the user is. After the research was conducted all the aforementioned software tools have web application version of their application. A mobile application was considered but only one version of the application would be made, either Android or iPhone, therefore preventing other phone users from using the application. A bespoke desktop application was considered for a long time, however, the user would have to run the application ensure that a lot of the infrastructure systems are set up correctly. As a result a web application was chosen - following research found; the next steps were to consider appropriate tools to use.

### 1.2.7 What should the application do

From analysing all three of the chosen research systems, it was clearly identifiable that they all have the ability the view all notes, searching, deleting and adding and editing a note. Taking the ideas on-board, these were set as a high-level task and something that the core system *must* do.

Reflecting on the premise of the application, that it was to aid the organisation lecture notes, it was concluded that the best way to search for notes would be by module code, as most University students would want to find specific module notes. This created the high level task that notes must be searchable by their module code.

### 1.2.8 Calendar Integration

From evaluating the systems it was noticed that there was not a clear way to integrate into a calendar. Reflecting on the conversations with Dr Harry Strange, integrating with the calendar was important for keeping the different systems together. From a survey [12] Google calendar is the most popular calendar application, therefore due to time constraints Google calendar was the choice of integration and other competitors such as Microsoft would not be implemented. This formulates the task of integrating the calendar into the application to save the url of the note to a specific event.

### 1.2.9 Objectives

As a result of the analysis of the problem, the following high-level requirements were formulated:

1. Investigate how to extract handwriting text from an image - this will involve looking into ways OCR tools can interpret handwriting.
2. Train the OCR to recognise text of the author's handwriting.
3. Produce a set of a rules which a note must comply to.
4. Produce a web application to form the core part of the product. This includes allowing a user to upload an image, display the image. Add appropriate tagging to a note such as module code.
5. The user must be able to search for a given module code, showing the fill list of notes based on the module code entered.
6. The backend of the application must conduct basic OCR recognition, analysing the first 3 lines of the notes.
7. The backend must integrate with a calendar to archive the notes away later to be found again.

### 1.2.10 Comprising with objectives

Some additional compromises were made in-light of the analysis due to the complexity of the tasks at hand.

- It would be nice to have image extraction from a note and incorporating a WYSIWG editor into the application, like OneNote.
- Full OCR on all the characters. This would then output the text to a blank canvas.
- Make the handwriting training generic enough to identify a wide range of users handwriting.

It is worth noting down that my supervisor Dr Hannah Dee felt as though the handwriting training would be too much for the dissertation and should be done as a “maybe”. After much deliberation it was decided to include it, but as a background process.

## 1.3 Process

Software projects often have a degree of uncertainty with requirements at the beginning, these projects lend themselves to an Agile approach. Whereas more structured applications with requirements which are well known are suited to a plan-driven approach.

For this project there are a lot of tasks which are not 100% definable at the start of the project. In addition to this certain tasks, such as training the author's handwriting data, can not be truly estimated down to a fixed time. Often new requirements would emerge from weekly meetings and only high level requirements were in-place from the start of the project. As a result, a plan-driven approach such as the Waterfall model would not be appropriate, and an Agile methodology was implemented.

### 1.3.1 Scrum overview

Scrum [13] is a methodology used by teams to improve productivity where possible. Due to this being a single person project, a Scrum approach has to be modified. Sprints are set time-boxes where tasks are completed. These vary from one to four weeks in length but a shorter sprint means the developer can act on quicker feedback.

Scrum organises its work into user stories to ensure client valued work is being completed. They are normally collected at the start of the project and put into the backlog, which is a collection of client values work. At the start of each sprint user stories are selected from the backlog with an estimation on complexity performed. Finally, at the end of the sprint a review and retrospective is conducted to analyse the sprint, identifying what went well and what could be improved.

### 1.3.2 Adapted Scrum

During the project this methodology was embraced and adapted. A one week long sprint was adopted which coincided with a weekly supervisor meeting. Epics (a high level version of a story) are the start of the project to reflect the work completed in the analysis phase.

Each user story was formulated as: “As a ¡role¿ I want to ¡feature¿ so that ¡resolution¿”. This gave specific client value that was known to have a purpose. Each of these stories were estimated on their complexity and compared to a “goldilock” task (a task which all other tasks are evaluated against).

For planning a sprint, the planning poker [15] technique was adopted; user-stories are estimated on a scale of 1, 2, 3, 5, 8 etc. When a task was estimated about 15 story points, it would be reflected upon to ensure the scope was fully understood - this would be broken down to sub-stories where appropriate.

At the end of a sprint a review and retrospective was conducted in the form of a blog post [6], instead of in a team. The retrospective was used to analyse what was achieved in the sprint, what went well and what needed to be improved upon. During this time, pre-planning was conducted to formulate a series of tasks to complete in the next sprint; this was agreed by the customer (Dr Hannah Dee).

Communication between myself and my supervisor was key to determine what needed to be completed. It was discussed if what was suggested was achievable in the week sprint, based on the total story points completed in the previous sprint; if twenty story points were completed in sprint 3 then 20 story points were estimated for sprint 4 - associated user stories were brought forward.

The project was managed on the open source management tool Taiga.io [17] which was invaluable, and provided built in functionality such as burndown charts per sprint. This shows how well story points are being completed and are used as an analytical tool.

### 1.3.3 Incorporated Extreme Programming

In tandem with Scrum, Extreme programming [2] principles were integrated into the development process; merciless refactoring, continuous integration and test-driven development were borrowed from its principles.

#### 1.3.3.1 Test-driven development

Test-driven development (TDD) is the process of writing tests prior to the implemented code. This allows the developer to think about the design prior to its implementation and can form part of the documentation [7].

TDD follows three cycles: red, green, refactor. Initially the test fails, then it passes then you refactor to keep the simplest system.

#### 1.3.3.2 Continuous Integration

Continuous Integration tools were a core part of the process in this project. Typically used to ensure that code is checked into a repository it was used to ensure that the application could be built in an isolated environment and pass all the tests.

#### 1.3.3.3 CRC cards

Class, responsibilities and collaboration (CRC) cards [19] were used during the design section to consider how different classes were to be created and the responsibilities they share. This principle from Extreme Programming helped to keep the design simple and not convoluted.

# Chapter 2

# Image pre-processing for Tesseract

## 2.1 ImageMagick

In order to produce the best possible results with Tesseract, there had to be a pre-processing stage beforehand. Initially, the image was converted to grayscale using ImageMagick[CITE] but this showed that training was difficult and was not yielding the success rates which was needed.

Pursuing using ImageMagick for the pre-processing stage the image was converted to Monochrome. This managed to be able to pick up more characters in the image, but in doing so picked up too much noise around the image. This distorted the characters and made it hard to recognise characters with the correct confidence that Tesseract required.

Dr Hannah Dee suggested that maybe OpenCV[cite] image manipulation would be useful for the pre-processing step, instead of using ImageMagick.

## 2.2 OpenCV

OpenCV is an open source image processing library. There was a choice of using the C++ or the Python bindings. Due to the applications core infrastructure it was agreed that Python would be easier to implement.

An investigation began by looking into the OTSU binarisation method and other versions of the adaptive thresholds. The need for a binarisation method, instead of using the default image, is that it reduced the ambiguity of the light sources on the image and any shadows which may occur during the capturing of the image.

After doing some investigations into different thresholding algorithms it came to light that Tesseract OCR uses OTSU thresholding as its underlying image processing algorithm. This quickly explained why a lot of characters were not being identified from images, when looking at the output of the experiment that were conducted. From this it could be seen that using OTSU as a pre-processing step for Tesseract would not return the best results.

Looking at the experiment of different thresholding techniques it was clear that adaptive mean and adaptive gaussian returned the image clearly binarised. There is no shadow or dark patch over the image and the text on the page is legible.

## 2.3 Line removal

After using Tesseract to train my handwriting on plain paper it showed that some sentences it could parse accurately, but those which are on a slant or slightly skewed, because of the lack of lines, returned a worse recognition rates.

To try and straighten the lines of text normal lined paper was used, but quickly found this was not good when binarising it as it often left unconnected lines which were represented as a series of dots. If the erode function was used on the image, it would lose the quality of the characters correctly identified.

This led to a creation of my own lined paper, one which would have thick lines and one which the lines could be filterable. The premise of using the blue lined paper was to correctly identify the blue lines and remove them from the image, leaving uniform lines of text which could be analysed. This process went through a couple of iterations.

To begin with it was decided to just extract the text from the image which fell between a black to gray threshold colour - excluding the blue lines from the image. This was followed up by the morphological operation, erode, to remove some of the remaining pixels left over by the lines. This worked to an extent and showed the binarised image with some loss of pixels on the characters and some were completely undistinguishable.

Although this worked well, it wasn't the most suitable solution as often line pixels were left in the image and would be picked up by the OCR resulting in an erroneous output from the system. So, again, it went through another iteration.

The next solution would binarise the image more elegantly and would actually remove the lines from the image and clear up the image to just leave the image binarised, with no blue or black lines.

[CITE REFERENCE USED TO DO THIS] Firstly, it would read the image as grayscale and then apply a median blur to the image. This was needed to try and ease the noise and smooth the image. Much like the first iteration adaptive threshold with a gaussian mean was used to binarise the image. Afterwards, a mask was collected containing the horizontal black lines, using the structuring elements "MORPH_RECT". This was eroded and dilated to try and remove the lines as much as possible. Intermediate masks so that black text was passed onto a new canvas - aiming to get as much of the text from the images across as a possible. This naturally carried some of the disjointed lines across, and suffered the same issues the previous iteration had to deal with. However, this iteration includes the use of finding contours.

The contours were considered using because a way was needed to find connected components and filter out the lines. Naturally, if the lines have been dilated and eroded then they will not be connected closely. Therefore, by choosing connected components on the image it will correctly get the characters from the image. Once the connected components of the characters were identified these were transferred to a blank mask, with a final erosion to clear up the image.

This technique has worked well in removing the blue lines from the page, whilst keeping the character quality. It clearly shows a binarised image with the text on the page clearly identified, and it worked on multiple examples. What is more impressing is that because of the adaptive threshold and different morphological operations then it can clearly identify characters in terribly lit photos.

# Chapter 3

# Handwriting training with Tesseract

[CITE THAT GUYS THESIS]

Handwriting recognition is still an active research project and one which is constantly evolving. There are many complexities with handwriting such as whether to analyse cursive or non-cursive text. The project could have been taken in a couple of direction: write my own handwriting recognition system or use an OCR tool. Since the premise of the application is to provide a software tool for a user creating my own handwriting recognition system is not viable. As a result, an OCR tool has been used.

Consideration went into the different OCR tools out there, with commercial vs non-commercial and there was one open source technology which seemed very reliable, Tesseract.[CITE PAPER ON COMPARISON].

Tesseract is an open source C++ library for analysing handwriting, the current stable version at the time of writing the report is 3.04. It is mainly interacted through a command line application, and that has what has been used in this project.

## 3.1 Training

To begin to analyse the characters on a page, there was a decision that the training process would involve analysing non-cursive handwriting. Although this limits the user experience, analysing handwriting is a challenging task in itself, so by making it non-cursive there was a better chance of good recognition.

Secondly, another caveat with training is that the training process would be consisted on the author's handwriting, and not on the general public. This again, limits the application to the user, but that could be expanded in the future.

In order to successfully, train the handwriting then the pre-processing steps described in chapter 2 are required. Outputted from the image pre-processing is a tiff file - which then used for the Tesseract training.

Firstly, the image has to be in the following format: ¡language¿.¡font¿.exp¡expnumber¿.tiff, for example the following file would be valid: eng.ryan.exp1a.tiff.

# Chapter 4

# Design

As the application was developed in an iterative manner, then no upfront formal class diagrams were conducted to design the system. Regardless, from the initial requirements it was clear that there was a set of design decisions that had to be made - these mainly were the implementation features. These would not change too significantly throughout the iterations.

## 4.1 Implementation tools

The following discusses the design decision in the implementation tools that were used during the project. This section provides rationale for key design decision which would impact the application.

### 4.1.1 Programming language

As the application was determined to be a web application then a design decision had to be made regarding the specific language to use for the server side. Normally for a server side application there would be a choice of languages to use such as: PHP, Ruby, Python, Java or more recently Node has increased popularity [CITE].

It was quickly decided in the first few meetings that OpenCV [CITE] would be needed to be implemented in the application at some point. Having previously used OpenCV with Python bindings it was known that Python could successfully integrate with a Python based application. Research was conducted to see if PHP or Ruby had some form of OpenCV wrapper, but after a little research there was not a lot of wrappers out there that looked promising.

As a result it was quickly established I'd have to use either Python or Java's wrapper for the application. Java applications tend to be more useful for large commercial applications, using enterprise software systems like Java EE[CITE]. That level of complexity was considered to be overkill for this application, therefore Python was chosen for the main language implementation due to the lightweight abilities, the ability to create web applications and knowledge of integrating with OpenCV.

### 4.1.2 Database management system

When considering the database aspect, one has to consider the different types of management systems available. The general consensus falls in the NoSQL vs SQL category. SQL management systems by their very nature are suited to data which is structured and has little variations. Whereas NoSQL is more suited to applications where there may be different values from time to time between the data.

After spending a long time evaluting the system and what could be achieved a SQL management system was selected. To do the basic tagging of meta-data would not warrant a NoSQL database structure, due to the information being structured and similar for every note. Couple this with the fact that there are a specific set of rules in which the notes must follow to tag this meta-data really suggested that the data will be nice and formulated, following the same structure for every note.

After determining that an SQL management system would be appropriate, a suitable technology would have to be chosen. When considering which technology to use, it tends to fall in the: SQLLite, PostgreSQL or MySQL.

#### 4.1.2.1 SQLLite

#### 4.1.2.2 PostgreSQL

### 4.1.3 MySQL

After evaluating the possible relational database management systems (RDMS) it was decided that PostgreSQL qould be the most suitable for my application.

### 4.1.4 Choice of Framework

Just to recall Python was chosen as the language of choice for the application. With this in mind a framework had to be chosen to help aid the development of the project. Frameworks can be interesting things, they are their to aid you but sometimes you're constrained to their rules and regulations. This can lead to many frameworks being bloated with additional options that you would never want or need to use. That said, choosing a framework was an important process.

After analysing what Python web frameworks were available, I shortlisted three different ones: Django [CITE], Flask [CITE] and Bottle [CITE]. The latter two were classified as a "micro-framework" whilst Django being the full blooded framework - like Ruby on Rails [CITE]. After establishing that maybe a heavyweight framework was probably too much for the application and I wanted to have freedom to include my own libraries and feeling as though there's more flexibility, I opted to rule out Django.

This left Flask and Bottle frameworks. On the face value of things, Flask and Bottle look quite similar. They are both lightweight with similar syntax, however when delving a little deeper into the technologies I found that Flask had more of a support community compared to Bottle. As I was not familar with either framework then choosing one which provided a good level of community support in tangent with the documentation was important. On making the decision based on this factor then Flask seemed the more appropriate framework to choose for this application.

### 4.1.5 Continuous Integration tool

Although Continuous Integration (CI) is normally used for development teams to ensure that all code is checked into the repository[cite], there was value in using it for a single person project. Prodomently for the build processes on the branches to ensure that the application builds in an isolated enviornment.

Having used Jenkins CI before [CITE] this was my immediate choice when considering what to use for a continous integration tool. However, whilst considering the posibilities on how to integrate this to the application I discovered the CI tool, Travis CI. This CI tool seemed easy to set up: you link it to the repository, add a travis.yml file (with configurations) and it will run and build the application. This then gives useful messages such as passing, errors or failing.

One key decision whilst considering CI tools was to show be able to check the build process whereever I am. After finding out the Travis CI is in the cloud and links directly with the GitHub repo I opted to use Travis CI for the project.

## 4.2 Overall Architecture

### 4.2.1 CRC cards

To recap, Extreme Programming uses the concept of CRC cards to help to aid the developer to think about the design of the application. Unlike UML, they are throw-away cards which can be edited and changed every time a new feature is implemented. The CRC cards which have been drawn up in the application have been formalised, and a selection of the interesting design decisions and justifications are stated below.

- Use of service objects
- Explain why sections are linked together and describe the behaviour of the application classes and why they would be used.
-

### 4.2.2 MVC Structure

Early on in the process it was decided that an MVC approach would be approached. This was an upfront design decision, that would not likely to be changed.

MVC is a design pattern where you split the logic into a Model View and controller.

- Explain a little bit on MVC
- Keen advocate of MVC
- I wanted components such as the models to be reusable where possible
- Had to change the way that Flask does the controllers to be blueprints to truely make this work.

- Diagram to show how the MVC framework would work.
- Why a package based system didn't work.

## 4.3 User Interface

Although the application was developed in an iterative approach, wiremocks were completed when UI changes occurred so that I could reflect on them.

To begin with the user interface (UI) went through a few iterations of wiremocks to sketch out what it should look like. This iterative approach was applied when actually designing the website. Firstly it was important that the application felt as thought it was an application, rather than a traditional website. Colours from Google Style guide [CITE] was used to keep with nice, clean and bold colours. Bootstrap[CITE] was considered to be used on the application and it did give a default responsive theme built in - however, keeping with the idea of minimalist and not over-bloated, then personally I feel that that bootstrap comes with a lot of the additional material which would not be needed.

With this being a web application, a responsive navigation was always important. A user may take a photo of their note and then upload this to the website off their mobile. This resulted in the web application having to be thought about from a responsive view. I should have done this better and considered this from the beginning and built up from the responsive to the desktop version. However, I did retrospective responsive design and performed media-queries when I needed to.

- Wiremock examples
- Media query code examples.

## 4.4 Other relevant sections

- Why did I use a Calendar Item class to story the class and then format the date from that instance.
- Why Parsing the notes returned the first 3 lines. Why did this have to be done?

### 4.4.1 Optimising Tesseract

- Attempt at going to GrayScale.
- Realised it wasnt good so redesigned it to custom lined paper Why did I do this instead of normal lined paper?
- Why the colour blue ?

### 4.4.2 Tesseract

- why Tesseract, what's different compared to other industry standard ones.

# Chapter 5

# Implementation

The implementation should look at any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex; perhaps third party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

You can conclude this section by reviewing the end of the implementation stage against the planned requirements.

## 5.1 Image processing

## 5.2 Handwriting Training

## 5.3 Web application

# Chapter 6

# Testing

Testing would form the core part of the application for *MapMyNotes*. Due to it prodomently being a web application then testing was an important process. In conjunction, there needed to be a wide-range of testing included to cover all possible approaches.

## 6.1 Overall Approach to Testing

To recall an agile approach was adopted throughout the project. As a result, a test-driven development approach was conducted.

### 6.1.1 TDD

Test-driven development was adopted throughout all aspects of the application. A solution always had an associated test and all code in the application has a purpose with a test behind it. Figure 6.1 shows the TDD cycle.

A sensible test was created and, following the cycle, this would fail when run. The following steps would be to ensure that the tests pass by adding the associated code needed to make sure that it passes - afterwards refactoring occurs to ensure that design is kept simple and as clean as

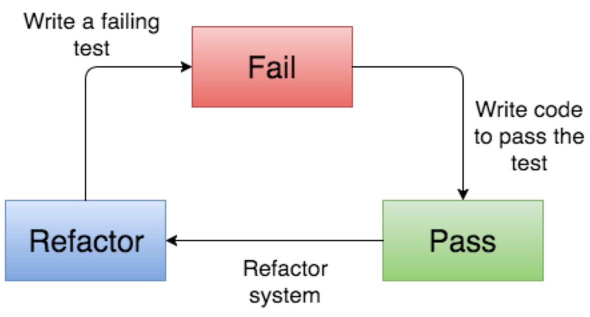

Figure 6.1: The cycle of TDD during the development stages of the application

possible for the current implementation.

This approach could have been modified so that a group of tests were created for a feature and then implement a set of functions. This testing strategy was rejected and a pure one test for one bit of functionality was used. This was mainly to ensure that the design was not being over complicated.

Reflecting on the testing strategy and the design, it really did help to think of the design through a test-driven-development way. It ensured that the domain was fully understood before creating a test. This left the codebase tested fully - through a variety of aspects.

## 6.2 Automated Testing

One thing to note is that Flasks testing documentation is very sparse and is of low quality.

To begin with py.test was originally being used to run the test suite and tests were created with just unittest.TestCase.

## 6.3 Mocking Tests

During testing it would be established quickly that tests would need to integrate with the Google API as well as the Tesseract helper class.

The need to for mocks vary between applications but thinking about the Google API example. [Cite] It is best practice not to hit a live production URL unless its in production. For testing Google's API's do not offer such luxury. After looking around they suggest that mocking would be a suitable idea.

The Tesseract example it may first seem odd to mock the response of certain functions. However, consider the possibilty of training data and then testing the application, then training some more. The results are likely to change - therefore to ensure that the tests have consistency and will work, data has been mocked to return a sensible output. Personally, the test cases was complex and not exactly the most intuitive.

During the first few iterations, whilst getting used to how it works, then working how to deal with mocks were conducted via annotations above the test function.

```
@mock.object(GoogleOauthService, 'authorise')
@mock.object(GoogleCalendarService, 'execute_request')
def test_return_correct_response(self, authorise, calendar_response):
  authorise.return_value = some_json
  calendar_response.return_value = some_more_json
```

This style of the testing syntax causes the codebase to become far too messy and unreadable. Additionally, large chunks of functionality was being duplicated going against my DRY (do not repeat yourself) principle.

Eventually, looking through the API docs for the mock object a patch object call was discovered. This allowed for a much cleaner test code base. These patches could be located in the setUp and tearDown function, replacing the annotations at the top of test functions.

The following code makes the mocking a lot more succinct, keeping it in the setUp functions and reducing the duplication

```
def setUp(self):
  # some code
  authorise_patch = mock.patch()
  authorise_mock = authorise_patch.start()
  authorise_mock.return_value = some_json

def teatDown(self):
  mock.patch.stopAll()
```

### 6.3.1 Unit Tests

After refactoring the testing infrastructure to use Flask-testing fraamework instead of the default python unit test, then unit testing became event easier.

The tests would extend the libraries TestCase class. This allows the application access to the application context (the current application) to test models.

Unit tests were conducted for all the classes in the model directory. Model testing themselves was relatively easy and simple to implement.

### 6.3.2 Route Testing

### 6.3.3 Handling sessions

One of the tricker aspects regarding testing was the handling of sessions. In parts of the application sessions are used to handle specific states of the system, i.e user logged in.

During testing of routes then session handling would not be too complex. The flask documentation gives a clear breakdown of how to modify a session. To begin with you have to open a test client context - this mirrors what a normal client would look like. From there a session transaction context needs to be opened. Once this has been opened then it would allow for session modification.

Issues arose during the acceptance tests where session modification would need to happen. Issues arose due to the fact that selenium would run on a different process to the application. As a result session modification could not occur during using selenium and acceptance test. This caused a lot of problems regarding testing. As a result, more mocking had to be used. This time the session helper would be then ended up mocking.

Overall, session handling was one of the hardest aspects with testing to get around and find a suitable, yet clean solution.

### 6.3.4 User Interface Testing

### 6.3.5 Stress Testing

## 6.4 Integration Testing

Integration tests would aid in thinking about not just the backend models, but how the information would be presented to the end user.

These tests were implemented in the form of using Python's selenium tool. As previously used before, selenium is a testing tool which allows you to integrate with the DOM and perform automated browser testing. These tests are important to ensure that the user interface is tested for input and ensuring values are correct.

An example of a selenium test to ensure that when the user goes to the file upload page and they're entering information into the meta-data form that associated calendar events are correctlt displayed.

Another useful test where selenium works well is when checking the background colour of the Tesseract output. By mocking the tesseract output the tests were able to test logic such as identifying the confidence score to the correct class name - and therefor the classname being identified with the correct background colour of the text.

## 6.5 User Testing

Due to the application aiming to solve a problem with a set of students then user studies were conducted and teh responses were analysed and evaluated.

## 6.6 Tesseract Testing

Due to there being no code written for the Tesseract training process there were no formal tests conducted for this section. However, what could be tested was how well the Tesseract learned as it was going through the learning process.

A test framework was produced which ran the training process on the image. As mentioned, it outputs a text file which is associated to the words identified. This file was then renamed to $eng.ryan.expx_stat.txt$ so that when the final part of the training process is ran it did not overwrite the file.

A statistical analysis was collected on the outputted

## 6.7 Image thresholding Testing

When writing my script for the image thresholding research it was soon discovered that the script would be very trial and error and very much like a protoyping.

When testing the output from the image, then it would be a visual check rather than a specific image that was looking to be achieved. Looking at each itteration through the scripts improvement it would be evaluated as whether it was a worse or better binarisation script. Once a sufficient level of satisfaction from eye-judgement was achieved then the testing from that script was stopped.

Once the spike work for the script had been completed, TDD performed on the image to turn the prototyped script into clean testable code. Using the outputted numpy arrays from open cv, they were compared against what was believed to be outputted.

When the image converts the white image mask to black it performs checks for the image for any black value in the image.

# Chapter 7

# Evaluation

Examiners expect to find in your dissertation a section addressing such questions as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

Review the discussion on the Evaluation section from the lectures. A recording is available on Blackboard.

# Appendices

# Appendix A

# Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. The key requirement is that we understand what is your original work and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

As an example, you might include a definition such as:

Apache POI library The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the clients existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [**?**]. The library is released using the Apache License [**?**]. This library was used without modification.

# Appendix B

# Ethics Submission

This appendix includes a copy of the ethics submission for the project. After you have completed your Ethics submission, you will receive a PDF with a summary of the comments. That document should be embedded in this report, either as images, an embedded PDF or as copied text. The content should also include the Ethics Application Number that you receive.

# Appendix C

# Code Examples

## 3.1 Random Number Generator

The Bayes Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs [?].

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0 - EPS)

double ran2(long *idum)
{
  /*--------------------------------------------------*/
  /* Minimum Standard Random Number Generator         */
  /* Taken from Numerical recipies in C               */
  /* Based on Park and Miller with Bays Durham Shuffle */
  /* Coupled Schrage methods for extra periodicity     */
  /* Always call with negative number to initialise    */
  /*--------------------------------------------------*/

  int j;
  long k;
  static long idum2=123456789;
  static long iy=0;
  static long iv[NTAB];
  double temp;

  if (*idum <=0)
  {
    if (-(*idum) < 1)
    {
      *idum = 1;
    }else
    {
      *idum = -(*idum);
    }
    idum2=(*idum);
    for (j=NTAB+7;j>=0;j--)
    {
      k = (*idum)/IQ1;
      *idum = IA1 *(*idum-k*IQ1) - IR1*k;
      if (*idum < 0)
      {
        *idum += IM1;
      }
      if (j < NTAB)
      {
        iv[j] = *idum;
      }
    }
    iy = iv[0];
  }
  k = (*idum)/IQ1;
  *idum = IA1*(*idum-k*IQ1) - IR1*k;
  if (*idum < 0)
  {
    *idum += IM1;
  }
  k = (idum2)/IQ2;
  idum2 = IA2*(idum2-k*IQ2) - IR2*k;
  if (idum2 < 0)
  {
    idum2 += IM2;
  }
  j = iy/NDIV;
  iy=iv[j] - idum2;
  iv[j] = *idum;
  if (iy < 1)
  {
    iy += IMM1;
  }
  if ((temp=AM*iy) > RNMX)
  {
    return RNMX;
  }else
  {
    return temp;
  }
}
```

# Annotated Bibliography

[1] "Evernote Tech Blog — The Care and Feeding of Elephants," https://blog.evernote.com/tech/2013/07/18/how-evernotes-image-recognition-works/, 2013, last checked 25th March 2016.

An article explaining how Evernote does character recognition on images

[2] R. Agarwal and D. Umphress, "Extreme Programming for a Single Person Team," in *Proceedings of the 46th Annual Southeast Regional Conference on XX*, ser. ACM-SE 46. New York, NY, USA: ACM, 2008, pp. 82–87. [Online]. Available: http://dx.doi.org/10.1145/1593105.1593127

This paper was useful on how Extreme Programming can be modified to a single person project. It provided thought on the methodology which should be undertaken on the project and how different aspects of Extreme Programming can be used.

[3] M. A. A. Dzulkifli and M. F. F. Mustafar, "The influence of colour on memory performance: a review." *The Malaysian journal of medical sciences : MJMS*, vol. 20, no. 2, pp. 3–9, Mar. 2013. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3743993/

A paper reviewing whether colour helps with memory retention. Used for the analysis and further confirmation in the taxonomy of notes section.

[4] Evernote, "The note-taking space for your life's work — Evernote," https://evernote.com/?var=c, 2016, last checked 17th April 2016.

The Evernote application is an example of the organisational and note-taking application that this project is looking at as a similar system.

[5] Google, "Meet Google Keep, Save your thoughts, wherever you are - Keep Google," https://www.google.com/keep/, 2016, last checked 17th April 2016.

Google keep is an organisational and note-taking application, it is used as part of the evaluation and background analysis. It was compared against what the application could do.

[6] R. Gouldsmith, "Ryan Gouldsmith's Blog," https://ryangouldsmith.uk/, 2016, last checked TODO.

A collection of blog posts which explain the progress every week through a review and reflection post.

[7] C. Maiden, "An Introduction to Test Driven Development — Code Enigma," https://www.codeenigma.com/community/blog/introduction-test-driven-development, 2013, last checked 17th April 2016.

A blog post giving a detailed description of what Test-driven development includes. Gives supportive detail to discussing that tests can be viewed as documentation.

[8] Microsoft, "Microsoft OneNote — The digital note-taking app for your devices," https://www.onenote.com/, 2016, last checked 13 April 2016.

Used to look at and compare how similar note taking applications structure their application. Used the applicatation to test the user interface and what functionality OneNote offered that may be usedful for the application

[9] ——, "Office LensWindows Apps on Microsoft Store," https://www.microsoft.com/en-gb/store/apps/office-lens/9wzdncrfj3t8, 2016, last checked 17th April 2016.

The Microsoft Lens application which would automatically crop, resize and correctly orientate an image taken at an angle.

[10] ——, "Take handwritten notes in OneNote 2016 for Windows - OneNote," https://support.office.com/en-us/article/Take-handwritten-notes-in-OneNote-2016-for-Windows-0ec88c54-05f3-4cac-b452-9ee62cebbd4c, 2016, last checked 17th April 2016.

An article on OneNote's use of handwriting extraction from an image. Shows simply how to extract text from a given image.

[11] O. Olurinola and O. Tayo, "Colour in learning: Its effect on the retention rate of graduate students," *Journal of Education and Practice*, vol. 6, no. 14, p. 15, 2015.

Discusses a study which shows that coloured text is better for the memory retention rates, than that of non-coloured text. Used during the taxonomy of notes section.

[12] A. Pilon, "Calendar Apps Stats: Google Calendar Named Most Popular — AYTM," https://aytm.com/blog/daily-survey-results/calendar-apps-survey/, 2015, last checked 13th April 2016.

A survery showing that Google calendar was ranked the most used calendar people use. Added to the analysis stage to justfy why Google calendar was chosen instead of other calendars available.

[13] Scrum.org, "Resources — Scrum.org - The home of Scrum," https://www.scrum.org/Resources, 2016, last checked 17th April 2016.

The website for the scrum methodology principles. The website was used to reference the process and methodology which was adapted in the project

[14] T. J. Smoker, C. E. Murphy, and A. K. Rockwell, "Comparing Memory for Handwriting versus Typing," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 53, no. 22, pp. 1744–1747, Oct. 2009. [Online]. Available: http://dx.doi.org/10.1177/154193120905302218

Used to show that there handwriting is still an important part of memory rentention with note taking, compared to digital text

[15] M. G. Software, "Planning Poker: Agile Estimating Made Easy," https://www.mountaingoatsoftware.com/tools/planning-poker, 2016, last checked 17th April 2016.

Showing the use of planning poker with exactly how it was implemented in the application using the scrum based approach.

[16] Tesseract, "Tesseract Open Source OCR Engine," https://github.com/tesseract-ocr/tesseract, 2016, last checked 17th April 2016.

The open source optical character recognition engine which will be used in the application to analyse characters on a page.

[17] Tiaga, "Taiga.io," https://taiga.io/, 2016, last checked TODO.

The project management toold which was utilised to help to keep track of the project's progress throughout the process. Utilised the Scrum tools available that the application gives.

[18] M. Webster, "Taxonomy — Definition of Taxonomy by Merriam-Webster," http://www.merriam-webster.com/dictionary/taxonomy, 2016, last checked 17th April 2016.

A definition of exactly what a teaxonomy is. Clearly labelling it as a classification of a problem.

[19] D. Wells, "CRC Cards," http://www.extremeprogramming.org/rules/crccards.html, 1999, last checked 17th April 2016.

A description of what CRC cards are and why they're useful when considering the design of an application. Used as a reference material throughout the process, as well as during the chapter discussing the process.