# 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

**1 Testing** **1**
1.1 Overall Approach to Testing . . . . . 1
1.1.1 Test-driven-development . . . . . 1
1.2 Automated Testing . . . . . 2
1.3 Mocking Tests . . . . . 2
1.3.1 Unit Testing . . . . . 4
1.3.2 Integration Testing . . . . . 4
1.3.3 Handling sessions . . . . . 4
1.4 Acceptance Testing . . . . . 5
1.5 User Testing . . . . . 6
1.6 Tesseract Testing . . . . . 7
1.7 Image thresholding Testing . . . . . 7

**Appendices** **9**

**A Tesseract data results** **10**
1.1 Table . . . . . 10

**Annotated Bibliography** **11**

# LIST OF FIGURES

1.1 The cycle of TDD during the development stages of the application . . . . . . . . 1
1.2 Example Unit test for the user class. Each of the tests pass . . . . . . . . . . . . 4
1.3 An example of the acceptance tests running. It shows that the time to run the tests have increased considerably. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
1.4 A simple framework showing the steps of analysing each of the training examples for a statistical measure for how successful the training process was. . . . . . . . . 7
1.5 A graph showing the success rate of the Tesseract training results over 12 examples. 8

## LIST OF TABLES

# Chapter 1

# Testing

This chapter discusses the testing strategy which has been implemented on the project. This includes unit, acceptance and user testing utilised throughout various parts of the application.

## 1.1 Overall Approach to Testing

To recape, an agile approach was adopted throughout the project. From the process, test-driven-development was used throughout the application for almost all aspects of testing.

### 1.1.1 Test-driven-development

Test-driven-development(TDD) was adopted throughout all aspects of the application. All implementation code as an test which covers the purpose of the implementation. Figure 1.1 shows the TDD cycle.

A sensible test was created and, following the cycle, this would fail when first tested. 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 possible for the current implementation.

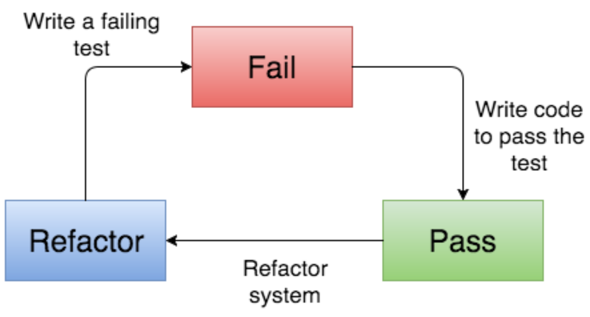


Figure 1.1: The cycle of TDD during the development stages of the application

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.

## 1.2 Automated Testing

One thing to note is that Flask's testing documentation is very sparse and is of low quality.

During the first few iterations of tests pytest was originally being used to create test classes, making all the classes extent $unittest.TestCase$.

Flask tests were refactored mid-way through the series of sprints to use Flask-testing[Cite]. This offers better testing support for Flask application, allowing the creation of dummy application and providing the functionality to run a quick server for testing.

## 1.3 Mocking Tests

Mocking during tests is changing the output of a function to return a specific value which is known about [CITE]. It was established that certain tests would need to be mocked, because some data would change from test to test. It was identified that *all* interactions with Google API's, any interactions with Tesseract and the Session would need to be mocked.

It is best practice when developing application that the developers should not hit a production URL during development and testing. As the Google API does not support specific environment API's, then all URLs would be to a production URL. The issue this raises when testing is ensuring that the tests are isolated and pass every time. For example, if the test queried the API one day it will return a specific result, but when queried another day it may return another result; this requires mocks to be used. The mock would return a specific value every time, ensuring tests can be reliable.

The principle is the same for testing Tesseract in the web application. If more training was conducted then the results from the test on the image would change, therefore ensuring that the tests have consistency data was mocked from specific functions - to check they returned the correct output.

During the first few sprints, whilst understanding how mocks work with Python, there was a lot of duplication with the mocking services. The library mock [CITE] uses annotations above test functions to signal a mock value.

Listing 1.1: An example of using mocks, following the annotation pattern

```
@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
```

Mocking example 1.1 shows the syntax which was initially used by the tests. This would result in many of the tests becomming unreadable and easy to follow exactly what the point was. Additionally the do not repeat yourself (DRY) principle was violated, by duplicating much of the codebase.

Looking the mock API documentation, patching object calls was discovered. Implementing this solution reduced the amount of code for mocking specific functions. The annotations were removed from the top of function tests. Initally it was not entirely clear how to implement these patch functions. Eventually the patch was included in the $setUp$ and $tearDown$ functions, as shown in figure **??**. The code for mocking is a lot more succinct.

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


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

Often when testing the code there needed to be ways in which the output varied depending on when it was called. In Python mock library they had the functionalty for that, and it was called side effects.

```
def setUp(self):
  # some code
  self.google_patch = mock.patch.object(GoogleCalendarService, ”execute_request”)
      self.google_mock = self.google_patch.start()
      self.google_mock.side_effect = [self.google_response, self.new_event, self.
          google_response, self.updated_response]

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

Figure 1.3 shows the use of the “side effect”. It is essentially, a way to define a series of output. From the above exame, it outputs google response first, then when $execute_request$ is called for a second time $new_event$ response is fired and so on. This was implemented due to the way the code was implemented in the controller, where multiple calls to the same function were executed, but different results were needed from the calendar.

Overall, mocking produced a substantial amount of testing. It was not considered when designing the system that mocking would be needed, it was only when testing happened that it was realised it was needed. The tests were refactored to incorporate the mocking facilities.

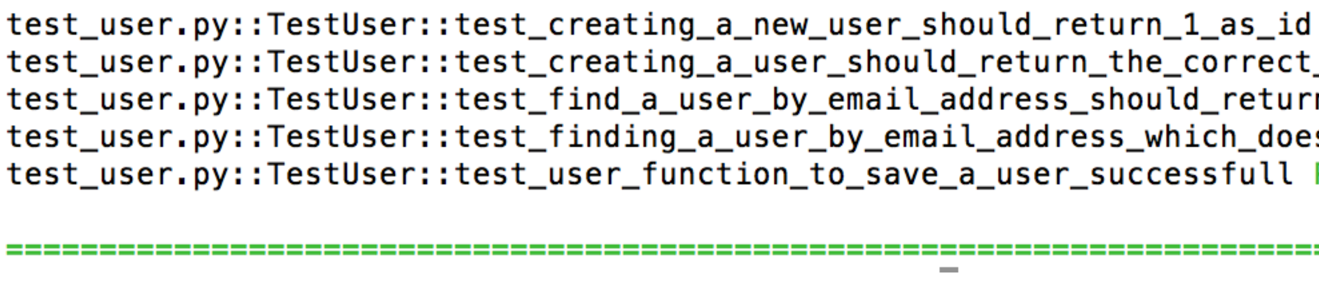
### 1.3.1 Unit Testing

A Unit test is used for models where functions can be tested in isolation to ensure that they perform the correct operations. In the application unit tests were created for every model.

Sometimes there was a cross over between database transactions and testing specific functions. In these cases, using the actual database would not be used for testing against. This is not considered the best design [CITE], so a test database was initialised.

The config was overwritten to include $app.config['testing'] = True$ and $app.config['SQLALCHEMY_{DA}$
$sqlite : ///test.sqlite'$. This ensured that a test database was used, which was cleared and created for every test - this ensured that the database did not influence other tests.

Very much like the function names of the classes, the test cases were as descriptive as possible. This formed part of the documentation of the application.



```
test_user.py::TestUser::test_creating_a_new_user_should_return_1_as_id
test_user.py::TestUser::test_creating_a_user_should_return_the_correct_
test_user.py::TestUser::test_find_a_user_by_email_address_should_retur
test_user.py::TestUser::test_finding_a_user_by_email_address_which_doe
test_user.py::TestUser::test_user_function_to_save_a_user_successfull

==================================================================
```

Figure 1.2: Example Unit test for the user class. Each of the tests pass

Figure 1.2 shows an example of the unit tests for the user class. The functions were decomposed and tests were associated for each function. In cases both edge-cases and normal expected results were selected.

Refer to appendix x for full unit test cases.

### 1.3.2 Integration Testing

Due to the application having external routes, then testing them to ensure the correct response codes was important. These were the first tests written for the routing sections, and implemented from the design considerations in X.

The tests consisted of checking that the response code was correct, any that redirects were correctly redirected to test the flow of the application.

Where there was integrations with the database in the routes, then tests were conducted to ensure the routes performed the correct persistance. For example, when posting to the meta-data URL, then checks were created to ensure that the module code was being persisted correctly.

### 1.3.3 Handling sessions

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

During testing, sections of the system are tested in isolated environments. If the route to show all the notes was tested, it would require the user to be logged in - therefore stored in the session. However, when testing this there will be no session set when running the test. Flask had a solution to testing the session handling [CITE].

```
with self.client.session_transaction() as session:
        session['user_id'] = self.user_id
```

Figure 1.3.3, displays an example on how the session had to be modified in route testing. After the session transaction context has exited then the session has been modified for the test.

However, issues arose during acceptance testing, where session modification was more cumbersome. When running selenium tests, the webdriver runs on a different process to the Flask applicaton [cite]. Therefore, session modification as shown in Figure 1.3.3, would not modify the session. After a lot of problems, it was confirmed that the session helper would have to be mocked, in the $create_app$ function, which is initialised when a test is created.

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

## 1.4 Acceptance Testing

Acceptance tests tests the system as a whole. This includes testing to ensure that the front-end functionality is what is expected when the user views the webpage.

To implement these tests a tool would be needed to check what the application looked like on a webpage, the tool selected was selenium for Python [CITE]. It can integrate with the DOM (document object model), and perform automated browser testing.

When testing with selenium there is the option to test with a standard browser, Chrome, Firefox etc, or it can be tested on a headless browser (via PhantomJS). A headless browser can access a webpage the same as Chrome, but it does not display a graphical user-interface [CITE].

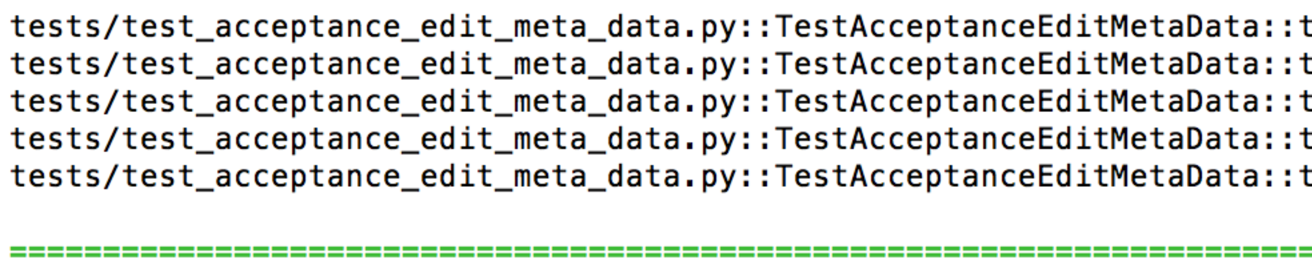
An example of an acceptance test may follow a pattern like:

1. Go to page /search.
2. Find the search field.
3. Enter the text "CS31310"
4. Click submit
5. Find "searched-item" from the DOM.
6. Return whether it equals "CS31310"

The above example shows that unlike other tests, which test the specific feature or functionality, this test checks what is displayed to the user via interactions with the web browser. The test then passes or fails based on assertions.

Another capability when testing was to check the output from the text colour from Tesseract. As there's logic to determine the colour in the application, then the test would help to determine if the correct colour was identified.

One disadvantage to this testing strategy is that it increases the time it takes to run the test-suite.



```
tests/test_acceptance_edit_meta_data.py::TestAcceptanceEditMetaData::t
tests/test_acceptance_edit_meta_data.py::TestAcceptanceEditMetaData::t
tests/test_acceptance_edit_meta_data.py::TestAcceptanceEditMetaData::t
tests/test_acceptance_edit_meta_data.py::TestAcceptanceEditMetaData::t
tests/test_acceptance_edit_meta_data.py::TestAcceptanceEditMetaData::t

======================================================================
```

Figure 1.3: An example of the acceptance tests running. It shows that the time to run the tests have increased considerably.

In Figure 1.3 it shows that the time to run to run 5 tests increased to 16.09 seconds. These tests are costly on time to run, but they ensure that the logic in the view files displays the correct dynamic content.

## 1.5 User Testing

Due to the application aiming to solve a problem, a set of likely user's were asked to perform a user study of the application. Their responses were analysed and their optinions on whether the software met their aims was collated.

Prior to the actual scheduled user-testing, feedback was given regarding the displaying of the Tesseract output confidence. These "over the shoulder" comments were along the lines of: "It would be great if you could click the identified text and it would automatically populate the text boxes". This was then implemented as a result from pre-user testing.

Further issues which were identified during the user testing were:

- Uploading a JPG image off a phone, which does not have the correct datetime exif key causes the application to fail.
- Uploading an image with a previously uploaded filename caused the application to display the old file.

These issues were caught and modified thanks to extensive user-testing of the application.

One interesting reflection of the user-case study was that people would not use the application. They were quick to defend the applications quality, but the use-case for them taking notes was not present. They much preferred to write up their notes from the lecture for memory retention.

## 1.6 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 progressed through the training processes.

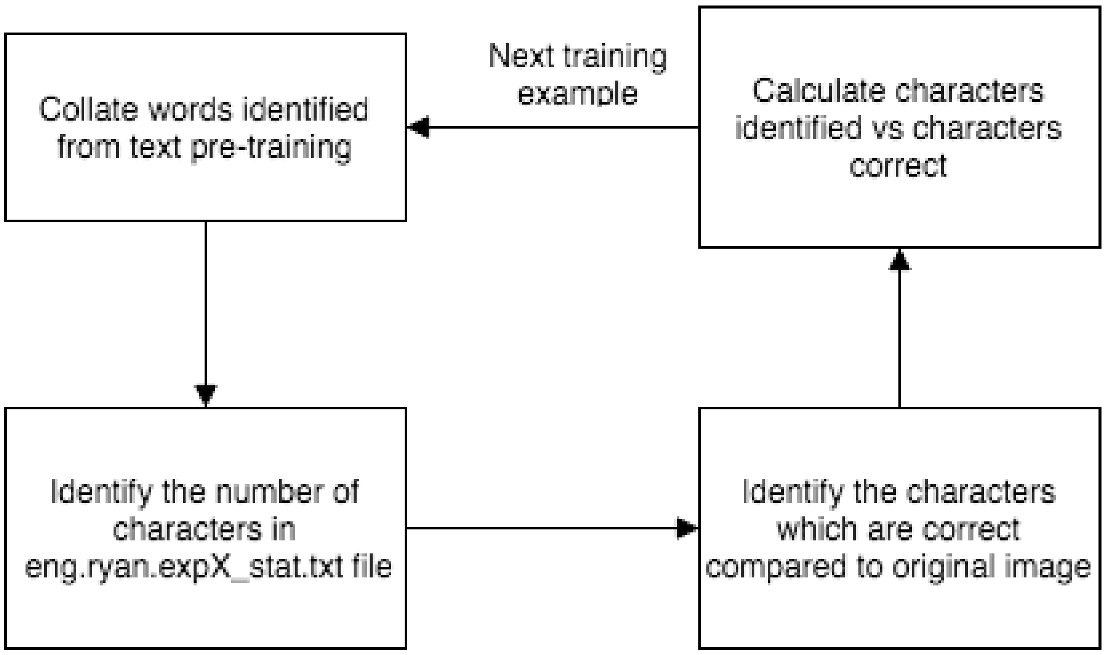

Collate words identified from text pre-training
Next training example
Calculate characters identified vs characters correct
Identify the number of characters in eng.ryan.expX_stat.txt file
Identify the characters which are correct compared to original image

Figure 1.4: A simple framework showing the steps of analysing each of the training examples for a statistical measure for how successful the training process was.

Figure 1.4 shows a simple framework for analysing how well Tesseract trained the data. After the statistics has been collated then a graph was constructed to show the trends.

Figure 1.5 shows the output analysed from the Tesseract training. It shows each training example with an associated correct character recognition. The conclusions clearly show that there is no improvement from the Tesseract output after around the 3rd example. A horizontal linear regression line shows that it has peaked at around 72% correct recognition rate. Refer to 1.1 for the full statistics.

## 1.7 Image thresholding Testing

Due to the nature of the image processing scripts it was quickly realised a methodical approach to testing would not be beneficial, due to almost constant spike work.

TDD would not be performed before considering the design for the image bininarsation script. Instead the testing would consist of a more viaual check of the outputted image to see if the result was clear enough, with little noise. Once the script had been developed to a reasonable level of satisfaction the spike work would cease, and testing would ensue.

The code was re-written and performed using TDD. It involved analysing numpy arrays re-

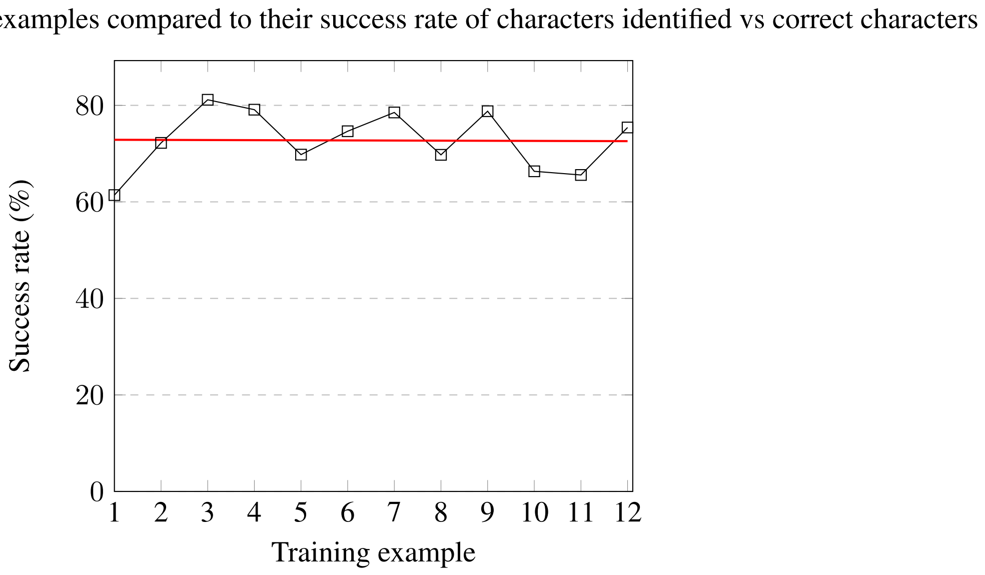


Figure 1.5: A graph showing the success rate of the Tesseract training results over 12 examples.

turned from OpenCV and checking, for instance, if any of the array contained black values.

ADD reference to appendix to show examples of output

# Appendices

# Appendix A

# Tesseract data results

This chapter shows the table outputting the results from the Tesseract training phase.

## 1.1 Table

| Experiment | Character Identified | Characters Correct | Correct Percentage |
|---|---|---|---|
| 1 | 114 | 70 | 61.40 |
| 2 | 252 | 182 | 72.22 |
| 3 | 345 | 280 | 81.15 |
| 4 | 335 | 265 | 79.10 |
| 5 | 288 | 201 | 69.79 |
| 6 | 276 | 206 | 74.63 |
| 7 | 326 | 256 | 78.52 |
| 8 | 400 | 279 | 69.75 |
| 9 | 462 | 364 | 78.78 |
| 10 | 401 | 266 | 66.33 |
| 11 | 366 | 240 | 65.57 |
| 12 | 362 | 273 | 75.41 |

# 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] Bottle, "Bottle: Python Web Framework Bottle 0.13-dev documentation," http://bottlepy.org/docs/dev/index.html, last checked 22nd April 2016.

The Python framework was used as a case-study of potential frameworks to use for the application. Discussed in the design section, but rejected as a choice.

[4] P. Developers, "PEP 8 – Style Guide for Python Code — Python.org," https://www.python.org/dev/peps/pep-0008/, last checked 23rd April 2016.

The PEP8 standard was used throughout the codebase as an implementation style guide. It is referenced in the evaluation to discuss the design decision that should have been implemented from the start of the project.

[5] Django, "The Web framework for perfectionists with deadlines — Django," https://www.djangoproject.com/, last checked 22nd April 2016.

The Python framework was used as a case study, looking at the different frameworks available. It was rejected for it being too large for the project.

[6] 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.

[7] 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.

[8] Flask, “Welcome — Flask (A Python Microframework),” http://flask.pocoo.org/, last checked 22nd April 2016.

The python framework used as an option. Was used in the design section evaluating the decisions that were made. It was used as the choice of framework.

[9] 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.

[10] 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.

[11] S. Knerr, L. Personnaz, and G. Dreyfus, “Handwritten digit recognition by neural networks with single-layer training,” *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 962–968, Nov. 1992. [Online]. Available: http://dx.doi.org/10.1109/72.165597

A paper describing how a Neural network was build to identify handwritten characters on the European database and the U.S. postal service database.

[12] 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.

[13] 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

[14] ——, “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.

[15] ——, "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.

[16] MongoDB, "MongoDB for GIANT Ideas — MongoDB," https://www.mongodb.com/, last checked 22nd April 2016.

The Mongo DB tool used as a comparison for relational database systems and NoSQL ones.

[17] 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.

[18] Oracle, "Overview - The Java EE 6 Tutorial," https://docs.oracle.com/javaee/6/tutorial/doc/bnaaw.html, last checked 22nd April 2016.

An article which discusses the use of Java as a web application language. It reaffirms the point raised that it is good for performance.

[19] 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.

[20] S. Rakshit and S. Basu, "Recognition of Handwritten Roman Script Using Tesseract Open source OCR Engine," Mar. 2010. [Online]. Available: http://arxiv.org/abs/1003.5891

The paper presents a case-study into the use of the Tesseract OCR engine. It analyses how to use train the data on handwriting based recognition, drawing conclusions on where it's useful - as well as it's downfalls.

[21] 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

[22] 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

[23] 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.

[24] 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.

[25] O. Tezer, "SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems — DigitalOcean," https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems, last checked 22nd April 2016.

Used as a comparison between what relational management system should be used. Used in the design section for a comparision between the different systems presented and evaluated.

[26] 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.

[27] w3Techs, "Usage Statistics and Market Share of JavaScript for Websites, April 2016," http://w3techs.com/technologies/details/pl-js/all/all, last checked 22nd April 2016.

The website shows a graph of how Javascript has increased its market share on recent web applications. Used as part of the design consideration regarding the use of programming language

[28] 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.

[29] 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.