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

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 recap, 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) aims to produce tests prior to the implementation of features. As a result, all implementation code is supported by a series of tests. Figure 1.1 shows the TDD cycle.

Initially a test is created, this would then fail due to no implementation code. 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.

With TDD it could have gone one of two ways: a group of tests were created for a feature, or

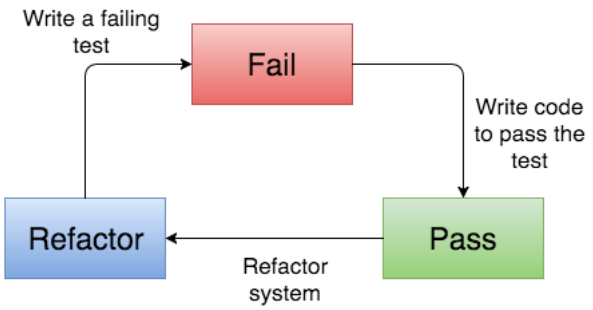

Figure 1.1: The cycle of TDD during the development stages of the application

one test for one bit of functionality. The latter approach was chosen, ensuring that the design was carefully considered.

Using TDD allows for the domain of the problem to be considered before implementing code to solve the issue. The user-stories were deconstructed into a series of tasks. There tasks were then formulated into different tests (unit, integration and acceptance). The cycle was then repeated.

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 sprints of testing, pytest [26] was originally being used to create test classes, with test classes sub-classing `unittest.TestCase`.

Flask tests were refactored mid-way through a series of sprints to use Flask-testing [15]. This offers better testing support for Flask application, allowing the creation of a fake application and providing the functionality to run a live server for testing, which would be needed for acceptance tests.

1.3 Mocking tests

The purpose of mocking is to change the output of a function to a value which is returned every time [12]. 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.

As discussed by Mathew Dale [4] when writing tests, mocks are needed because of external factors out of a developer's control. As the Google API does not support specific environment API's, such as production or development, 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, the test could query the API once and pass the test, however on the next query it may fail due to a different return value; this requires mocks to be used. The mock would return a specific value every time, ensuring consistency among tests.

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 the data was mocked from the returned text functions to provide consistency.

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 [11] 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
```

In code example 1.1 shows the syntax which was initially used in the tests, for mocking. This would result in many of the tests becoming unreadable obfuscated. Additionally the do not repeat yourself (DRY) principle was violated, by duplicating much of the codebase.

```
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()
```

Investigating 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. Initially 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 example **??**.

Often when testing there needed to be way to alternate the output of a function. In the Python mock library side effects allowed the mock object to return different values.

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

Figure 1.3 shows the use of the "side effect" API. From the above example, it outputs Google response first, then when `execute_request` is called for a second time `new_event` response is fired and so on. An example usage of this would be for when the Google Calendar needed to get a user's events. Firstly, in the controller it would get a list of events and then it would get a singular event. Since the same function call was used, a test needed to be added to check the return values were indeed correct.

Overall, mocking produced a substantial amount of the 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 as this would interfere with the actual data in the system.

The config was overwritten to include a test SQLite database. This ensured that a test database was used, which was cleared and created for every test. This allowed each test to be tested independently of one another.

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 PASSED
test_user.py::TestUser::test_creating_a_user_should_return_the_correct_email PASSED
test_user.py::TestUser::test_find_a_user_by_email_address_should_return_user PASSED
test_user.py::TestUser::test_finding_a_user_by_email_address_which_doesnt_exist PASSED
test_user.py::TestUser::test_user_function_to_save_a_user_successfull PASSED

========================================================================== 5 passed in 1.83 seconds ===
```

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 section **??**.

The tests consisted of checking that the response codes were correct, any redirects were correctly redirected; this tested to ensure the application flow was implemented correctly.

Where there was interactions with the database in the routes, then tests were conducted to ensure the routes performed the correct persistence. 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 trickier 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 storing data 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 [10].

Listing 1.2: An example of how sessions were handled and modified in the tests.

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

Figure 1.2, displays an example on how the session had to be modified in the integration tests. After the session transaction context has exited then the session has been modified for that test.

Acceptance testing initially proved to be problematic. When testing with the sessions, the selenium tests would not acknowledge that a session had been set, like in Figure 1.2, causing the tests to fail. 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 overcome to provide working tests.

1.4 Acceptance testing

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

To implement these tests a tool would be needed to check what the application looked like on a web-page, the tool selected was selenium for Python [22]. 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 [27].

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 example of the selenium tests being useful was checking the output colour from Tesseract. The logic in the view file determined the colour and Selenium was able to confirm the logic was correct.

```
tests/test_acceptance_edit_meta_data.py::TestAcceptanceEditMetaData::test_edit_form_is_displayed_on_the_page PASSED
tests/test_acceptance_edit_meta_data.py::TestAcceptanceEditMetaData::test_edit_form_populates_existing_information_correctly PASSED
tests/test_acceptance_edit_meta_data.py::TestAcceptanceEditMetaData::test_ensure_the_fields_have_required_key PASSED
tests/test_acceptance_edit_meta_data.py::TestAcceptanceEditMetaData::test_when_editing_the_date_it_shows_unable_to_save_to_calendar_if_no_event_was_found PASSED
tests/test_acceptance_edit_meta_data.py::TestAcceptanceEditMetaData::test_when_editing_the_date_updates_event_link_should_be_new_html PASSED

========================================================================== 5 passed in 16.09 seconds ===========================================================================
```

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; one of the disadvantages is the time taken to run the test-suite. Due to the complexity with loading data correctly to the view file, then these tests are imperative to ensure the user expects to see the correct content.

1.5 User Testing

Due to the application aiming to solve a problem, a set of potential user's were asked to perform a user study of the application. Their responses were analysed and their opinions 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 date-time exif-data 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.

Figure 1.4: A pie chart from the Google forms questionnaire that the users conducted showing that they would not use the application for archiving their notes.

One interesting reflection of the user-case study was that people would not use the application, as shown in Figure 1.4. 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 learnt as it progressed through the training process.

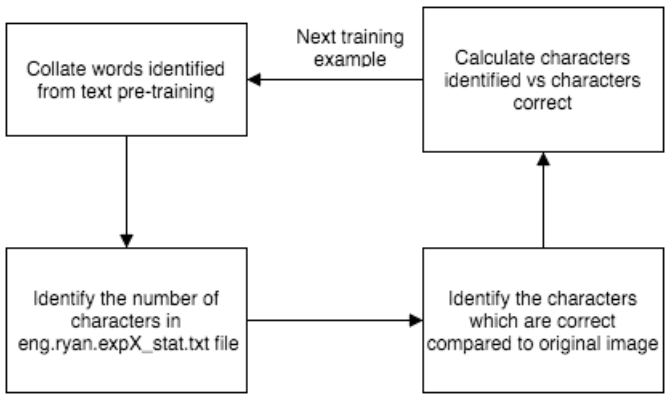

Figure 1.5: A simple framework showing the steps of analysing each of the training examples for a statistical measure for how successful the training process was.

Figure 1.5 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.

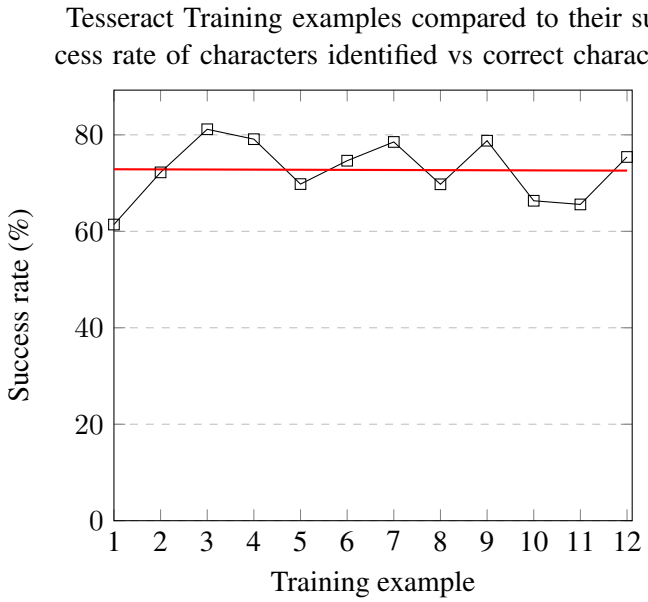

Figure 1.6: A line-graph showing the success rate of the Tesseract training results over 12 examples.

Figure 1.6 shows the output analysed from the Tesseract training. It shows each training example with an associated success rate for the characters identified. 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 appendix B, section 2.1 for the full statistics.

1.7 Image threshold Testing

Due to the nature of the image processing script 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 visual check of the outputted image to see if the result was successfully binarised. 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 following a TDD approach. It involved analysing numpy arrays returned 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

Testing Results

This appendix chapter shows the different sections of the application that has been tested and the test outcomes.

1.1 Unit tests

1.1.1 Binarise image

1.1.2 Calendar item

1.1.3 DateTimeHelper

1.2 Integration tests

1.3 Acceptance tests

The following section displays visual representation of the acceptance tests being executed, and their overall status.

1.3.1 Homepage

```
tests/test_acceptance_homepage.py::TestAcceptanceHomepage::test_once_authorised_it_displays_users_email_address PASSED
tests/test_acceptance_homepage.py::TestAcceptanceHomepage::test_should_display_the_correct_events_in_calendar PASSED
tests/test_acceptance_homepage.py::TestAcceptanceHomepage::test_signing_in_does_not_show_the_sign_in_button PASSED

============================================================================== 3 passed in 9.30 seconds ===============
```

Figure A.1: Acceptance test being conducted for the homepage, to ensure that the homepage displays the correct content.

```
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_clicking_on_date_field_shows_datepicker PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_clicking_on_time_field_shows_timepicker PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_clicking_suggested_lecturer_from_tesseract_populates_lecture_field PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_clicking_suggested_module_code_from_tesseract_populates_module_code_field PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_clicking_suggested_title_from_tesseract_populates_title_field PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_ensure_the_fields_have_required_key PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_form_does_not_show_exif_data_if_image_is_a_png PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_form_exists PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_form_has_correct_url_action PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_form_has_date_of_lecturer_field PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_form_has_lecturer_name_field PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_form_has_location_field PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_form_has_module_field PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_form_has_title_exists PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_form_shows_exif_data_from_image PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_google_calendar_event_shows_when_exif_data_matches PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_google_calendar_response_without_a_date_time_field_ignores_the_response PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_module_field_label_content PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_module_field_label_exists PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_submit_button_exists PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_tesseract_data_is_coloured_correctly_for_confidence PASSED
tests/test_acceptance_meta_data_form.py::TestAcceptanceMetaDataForm::test_tesseract_data_shows_when_image_is_uploaded PASSED

========================================================================== 22 passed in 115.96 seconds ==========================================================
```

Figure A.2: Acceptance test being performed to ensure that meta-data can be added to the correct note.

1.3.2 Add meta-data

1.3.3 Edit meta-data

```
tests/test_acceptance_edit_meta_data.py::TestAcceptanceEditMetaData::test_edit_form_is_displayed_on_the_page PASSED
tests/test_acceptance_edit_meta_data.py::TestAcceptanceEditMetaData::test_edit_form_populates_existing_information_correctly PASSED
tests/test_acceptance_edit_meta_data.py::TestAcceptanceEditMetaData::test_ensure_the_fields_have_required_key PASSED
tests/test_acceptance_edit_meta_data.py::TestAcceptanceEditMetaData::test_when_editing_the_date_it_shows_unable_to_save_to_calendar_if_no_event_was_found PASSED
tests/test_acceptance_edit_meta_data.py::TestAcceptanceEditMetaData::test_when_editing_the_date_updates_event_link_should_be_new_html PASSED

========================================================================== 5 passed in 16.21 seconds ===========================================================
```

Figure A.3: Acceptance test being conducted so that a note's meta-data can be edited successfully.

1.3.4 Search

```
tests/test_acceptance_search.py::TestAcceptanceSearch::test_clicking_view_note_shows_the_note_with_meta_data PASSED
tests/test_acceptance_search.py::TestAcceptanceSearch::test_form_with_search_bar_is_displayed PASSED
tests/test_acceptance_search.py::TestAcceptanceSearch::test_notes_not_included_from_other_modules PASSED
tests/test_acceptance_search.py::TestAcceptanceSearch::test_only_display_the_logged_in_users_notes_not_others PASSED
tests/test_acceptance_search.py::TestAcceptanceSearch::test_searching_for_a_module_that_doesnt_exist_return_message PASSED
tests/test_acceptance_search.py::TestAcceptanceSearch::test_searching_for_form_returns_a_note PASSED
tests/test_acceptance_search.py::TestAcceptanceSearch::test_when_searched_for_it_shows_the_user_what_they_have_search PASSED

========================================================================== 7 passed in 29.43 seconds ===========================
```

Figure A.4: Acceptance test to ensure that a user can search for a module code and it displays their notes.

1.3.5 Viewing all the notes

```
tests/test_acceptance_view_all_notes.py::TestAcceptanceShowNote::test_to_view_all_notes PASSED

========================================================================== 1 passed in 7.24 seconds ==
```

Figure A.5: Acceptance test being conducted to ensure that all the notes can be viewed.

```
tests/test_acceptance_show_note.py::TestAcceptanceShowNote::test_date_values_are_correct PASSED
tests/test_acceptance_show_note.py::TestAcceptanceShowNote::test_delete_link_is_available PASSED
tests/test_acceptance_show_note.py::TestAcceptanceShowNote::test_displaying_whether_event_was_added_a_users_calendar_return_true PASSED
tests/test_acceptance_show_note.py::TestAcceptanceShowNote::test_edit_link_is_available PASSED
tests/test_acceptance_show_note.py::TestAcceptanceShowNote::test_image_loads_on_show_note_page PASSED
tests/test_acceptance_show_note.py::TestAcceptanceShowNote::test_lecturer_name_is_correct PASSED
tests/test_acceptance_show_note.py::TestAcceptanceShowNote::test_location_name_is_correct PASSED
tests/test_acceptance_show_note.py::TestAcceptanceShowNote::test_module_code_is_correct PASSED
tests/test_acceptance_show_note.py::TestAcceptanceShowNote::test_title_value_are_correct PASSED

============================================================================== 9 passed in 42.97 seconds ===============================
```

Figure A.6: Acceptance test being conducted to make sure that a singular note can be viewed correctly.

1.3.6 Show a note

1.4 Integration tests

1.4.1 Add and edit meta data

```
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_add_meta_data_route_get_request_not_allowed PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_add_meta_data_route_returns_302 PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_add_module_code_via_post_request_successfully PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_edit_route_upload_erroneous_date_format_returns_error PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_edit_route_upload_erroneous_time_format_returns_error PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_get_edit_note_information_returns_200_success PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_it_saves_a_note_object_once_the_meta_data_added PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_once_a_note_is_saved_it_redirects_to_show_note PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_post_to_edit_note_changes_the_foreign_key_association PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_post_to_edit_note_different_data_created_new_meta_data PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_post_with_already_existing_meta_data_should_return_instance PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_posting_exisiting_module_code_new_meta_data_new_instance PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_posting_redirects_back_to_show_note PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_uploading_empty_data_returns_error PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_uploading_erroneous_date_format_returns_error PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_uploading_erroneous_time_format_returns_error PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_using_the_different_module_code_should_save_new_code PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_using_the_same_module_code_as_before_if_one_exists PASSED
tests/test_integration_add_edit_meta_data.py::TestIntegrationAddEditMetaData::test_when_session_doesnt_contain_user_id_redirect_homepage PASSED

============================================================================== 19 passed in 6.17 seconds ===============================
```

Figure A.7: Integration tests carried on the add and edit meta url to ensure the system worked well together.

1.4.2 Homepage

```
tests/test_integration_homepage.py::TestIntegrationHomePage::test_credentials_not_in_session_return_blank_homepage PASSED
tests/test_integration_homepage.py::TestIntegrationHomePage::test_displays_logout_link_if_logged_in PASSED
tests/test_integration_homepage.py::TestIntegrationHomePage::test_home_route PASSED
tests/test_integration_homepage.py::TestIntegrationHomePage::test_if_not_logged_in_it_doesnt_display_logout PASSED
tests/test_integration_homepage.py::TestIntegrationHomePage::test_sign_in_displays_if_not_authorised PASSED

============================================================================== 5 passed in 0.94 seconds ================================
```

Figure A.8: Integration tests conducted on the homepage to ensure that the routes were accessible.

1.4.3 Logout

```
tests/test_integration_logout.py::TestIntegrationLogout::test_logout_removes_the_credentials_key_from_session PASSED
tests/test_integration_logout.py::TestIntegrationLogout::test_logout_removes_the_user_id_from_session PASSED
tests/test_integration_logout.py::TestIntegrationLogout::test_logout_route_does_not_permit_post_requests PASSED
tests/test_integration_logout.py::TestIntegrationLogout::test_logout_route_returns_a_200_error PASSED

============================================================================== 4 passed in 0.77 seconds ================================
```

Figure A.9: Integration tests conducted for the logout route ensuring the routes are logged out.

1.4.4 Oauth

```
tests/test_integration_oauth.py::TestIntegrationOAuth::test_call_back_route_returns_a_success_status PASSED

============================================================================== 1 passed in 0.65 seconds ==========
```

Figure A.10: Integration tests conducted for the oAuth route which interacts with the Google API.

1.4.5 Search

```
tests/test_integration_search.py::TestIntegrationSearch::test_if_user_not_in_session_return_to_homepage PASSED
tests/test_integration_search.py::TestIntegrationSearch::test_search_route__with_code_can_not_permit_post_requests PASSED
tests/test_integration_search.py::TestIntegrationSearch::test_search_route_returns_200_status_code PASSED
tests/test_integration_search.py::TestIntegrationSearch::test_search_route_with_code_returns_200_status_code PASSED
tests/test_integration_search.py::TestIntegrationSearch::test_search_with_post_request_returns_405 PASSED

============================================================================== 5 passed in 0.89 seconds ==========
```

Figure A.11: Integration tests conducted for the search URL to ensure searching works correctly.

1.4.6 Show note

```
tests/test_integration_show_note.py::TestIntegrationShowNote::test_deleting_a_note_deletes_a_note_from_database PASSED
tests/test_integration_show_note.py::TestIntegrationShowNote::test_deleting_a_note_returns_status_code_200 PASSED
tests/test_integration_show_note.py::TestIntegrationShowNote::test_route_returns_status_code_200 PASSED

============================================================================== 3 passed in 0.86 seconds ==========
```

Figure A.12: Integration tests implemented to ensure that the note can be displayed properly.

1.5 Upload

```
tests/test_integration_upload.py::TestIntegrationUpload::test_get_upload_route PASSED
tests/test_integration_upload.py::TestIntegrationUpload::test_put_upload_route PASSED
tests/test_integration_upload.py::TestIntegrationUpload::test_saving_file_attached PASSED
tests/test_integration_upload.py::TestIntegrationUpload::test_should_not_allow_post_to_show_image_route PASSED
tests/test_integration_upload.py::TestIntegrationUpload::test_should_return_200_error_on_404_page PASSED
tests/test_integration_upload.py::TestIntegrationUpload::test_should_return_image PASSED
tests/test_integration_upload.py::TestIntegrationUpload::test_should_save_the_correct_tif_file_to_upload PASSED
tests/test_integration_upload.py::TestIntegrationUpload::test_show_image_route PASSED
tests/test_integration_upload.py::TestIntegrationUpload::test_uploading_file_status PASSED
tests/test_integration_upload.py::TestIntegrationUpload::test_uploading_right_file_extension PASSED
tests/test_integration_upload.py::TestIntegrationUpload::test_uploading_without_file_attached PASSED
tests/test_integration_upload.py::TestIntegrationUpload::test_uploading_wrong_file_extension PASSED
tests/test_integration_upload.py::TestIntegrationUpload::test_when_uploaded_file_redirects_to_show_image_route PASSED

============================================================================== 13 passed in 5.77 seconds ==========
```

Figure A.13: Integration tests implemented to ensure that a user can upload their images to the application.

```
tests/test_integration_user.py::TestIntegrationUser::test_user_route PASSED

============================================================================== 1 passed in 0.77 seconds ======
```

Figure A.14: Integration tests implemented the user route is working correctly and a the user gets added to the database.

```
tests/test_integration_view_all_notes.py::TestIntegrationViewAllNotes::test_redirect_to_homepage_if_user_session_not_set PASSED
tests/test_integration_view_all_notes.py::TestIntegrationViewAllNotes::test_show_all_notes_returns_200_success_code PASSED

============================================================================== 2 passed in 0.72 seconds ==============================
```

Figure A.15: Integration tests to make sure the view all notes url is working and getting the appropriate notes from the database.

1.6 User

1.7 View all notes

1.8 User tests

Appendix B

Tesseract data results

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

2.1 Table

Experiment	Characters 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

Table B.1: A table which shows the statistics from the correctly identified characters during the training process.

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] M. Daly, "Mocking External Apis in Python - Matthew Daly's Blog," http://matthewdaly.co.uk/blog/2016/01/26/mocking-external-apis-in-python/, Jan. 2015, last checked 25th April 2016.

A nice simple blog post explaining why hitting an external API is bad, and there should mocking objects instead.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] ——, "Testing Flask Applications Flask Documentation (0.10)," http://flask.pocoo.org/docs/0.10/testing/\#accessing-and-modifying-sessions, 2016, last checked 24th April 2016.

The testing documentation for Flask which discusses how session modifications should be handled. Used in the implementation and the testing discussion.

[11] T. P. S. Foundation, "26.5. unittest.mock mock object library," https://docs.python.org/3/library/unittest.mock.html, 2016, last checked 24th April 2016.

The mocking library used throughout the application. Although the documentation is for python 3, it works for python 2.7

[12] M. Fowler, "Mocks Aren't Stubs," http://martinfowler.com/articles/mocksArentStubs.html, last checked 25th April 2016.

When deciding whether mocks or stubs were used, Martin Fowler gave a nice concise answer. It turns out all the tests are mocking the behaviour from the external API.

[13] 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.

[14] 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.

[15] C. Heer, "Flask-Testing Flask-Testing 0.3 documentation," http://pythonhosted.org/Flask-Testing/, last checked 25th April 2016.

The documentation page for the testing library Flask-Testing. It was used throughout the project after a refactor realising it offered better support for testing Flask applications.

[16] 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.

[17] 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.

[18] 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

[19] ——, "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.

[20] ——, "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.

[21] 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.

[22] B. Muthukadan, "Selenium with Python - Selenium Python Bindings 2 documentation," https://selenium-python.readthedocs.org/, 2014, last checked 24th April 2016.

The selenium library used for the acceptance tests. It gives good documentation on how to access elements and how to get specific values from the text.

[23] 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.

[24] 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.

[25] 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.

[26] pytest-dev team, "pytest: helps you write better programs," http://pytest.org/latest/, last checked 24th April 2016.

The library was used throughout the development for reference on testing. It was especially useful for mocking test data.

[27] R. Python, "Headless Selenium Testing with Python and PhantomJS - Real Python," https://realpython.com/blog/python/headless-selenium-testing-with-python-and-phantomjs/, Aug. 2014, last checked 24th April 2016.

A demonstration on how to use Selenium with the Python examples. Additionally references the fact what phantomjs is, and it is a headless browser.

[28] 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.

[29] 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

[30] 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

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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

[36] 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.

[37] 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.