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

Design

As the application was developed in an iterative manner, over a series of sprints, class diagrams and design diagrams were not created at the very start of the project. Over a series of sprints designs were iteratively developed regarding the system overview. However, some design decisions were decided at the start of the project. The chapter will clearly explain rationale for the decisions and state whether the design decisions were a result of iterative processes or upfront design.

1.1 Overall Architecture

This section discusses the architecture of the web application as a whole. The process of the design with the web application was developed over a series of sprints, rather than an upfront design.

1.1.1 Class Diagram

An overview of the resultant design of the class diagram is presented, with rationale for decisions made and how an iterative approach was used to reach the concluded design. The class diagram can be found in appendix G, section 7.1.

1.1.1.1 Justification of design

The following section discussing the appropriateness of the design and justification, as well as exploring the evolutionary design. Overall, the design clearly shows the object oriented principle of low coupling high cohesion being used on the project.

Google Services

During the first iterations, the Google Calendar API was only going to be used to parse the calendar events. Therefore, the class `GoogleCalendarService` was created, to extract the code away from the controllers. The version number and API was unlikely to change, but they were formed as constants for the ease of changing if in the future they did. This class needed to perform key operations to extract the events, such as `get_events_based_on_date`. The functions themselves were iteratively developed, initially only using the `execute_request` and `get_list_of_events`. Due to the scope changing with complexity, in the latter sprints further

methods were added.

Initially user's were not a part of the system. However, after implementing the associated user story it was clear that another class to integrate with the Google Plus API would be needed. This followed the similar structure as the calendar class, except for parsing a user's email to persist in the database.

Eventually, it was seen that this design was repeating functionality in places. In both of the classes the `build` and `execute_request` were duplicated, without having class specific content. As a result, the extract class refactoring technique was used to create a super class `BaseGoogleService`. This class encapsulates the logic for building queries and executing them. With both the google plus and calendar classes sub-classing this, it extracts the final aspect of querying to the super class leaving logic and query manipulation to the sub-classes.

Helper classes
Helper classes, in the design, are independent classes that help to modularise the system - whilst collecting related functionality into a single class. Helper classes were developed iteratively, encapsulating similar functionality into one place.

For example the `SessionHelper` class was developed in such a way, that during the first few iterations of design it was only used to ensure that the credentials were appended correctly to the session. This functionality was duplicated, so an additional class was extracted. Over the course of adding user management and storing errors the method's grew.

As noticed, most of the helper classes do not interact with the other classes in any class. There is an exception with the `GoogleServicesHelper` class. In this class majority of the function are static. They're static because they do not interact with any class variables. Prior to the implementation of editing a calendar event, the code was dispersed throughout the controllers. In an effort to reduce the code this helper class was created - but it was quickly decided that it would just be a proxy for calling specific function in each of the services classes. Therefore the functions were converted to static as true OO principles were not needed with these.

Persistence classes
The relationships between the persistence classes are not discussed in this section, see section **??**. It is worth noting that designing the persistence classes was again an iterative process through the sprints. For example, for majority of the sprints the title attribute in the `NoteMetaData` class was not added. It wasn't until the design changed and this field needed to be added a reflection was added.

There are a series of `save` functions in the application, these were added to the design when the controllers were constantly adding it to the database session. There needed to be a succinct way to save that specific instance. Therefore, database session code was extracted to this method, allowing each instance object to be saved.

In parts of the application, there needed to be a way to extract information from the database. To aid in readability, static methods such as `find_meta_data` was created to keep domain related functionality together, but without having the need to create a specific instance.

Binarisation
The `BinariseImage` class is the model version of the image segmentation script see section **??**. The class would interact with the controllers and be called when a user uploads their file. It was important to have this as a model, due to the core functionality it offers to the system.

1.1.2 CRC cards

To aid with the design, class collaboration cards (CRC) was drawn up for each feature. The user-story was decomposed into tasks and each of the tasks had associated CRC cards. This helped to think about the design. The overall design shown in section 1.1.1 is a result of the diligent planning with the CRC cards.

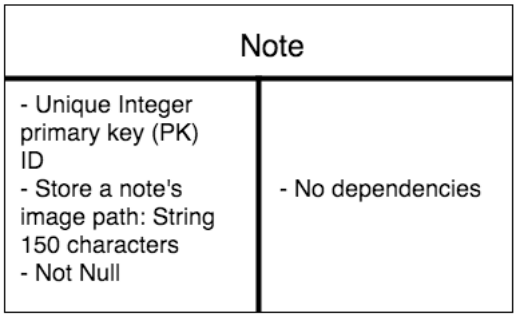


Figure 1.1: An example from Sprint 3, showing a CRC card at the very beginning of creation.

Figure 1.1 shows an example of a CRC card at the very beginning of the note class creation. The left hand side helped to think about methods and attributes for the class. The right hand side shows the responsibilities, where the note may interact with other classes.

These classes were evaluated and considered throughout each feature from a user-story. There were times during the design that these helped to keep a simple design, adopting YAGNI. For example, the note has an image path attribute, this was going to be extracted into its own relation, but after evaluating at the time a one-to-one relationship would not add any benefits to a system. Therefore the CRC card enabled a concise clear design, which allowed reflection and evaluation.

Appendix **??** section **??** shows a range of CRC cards used iteratively on the application.

1.1.3 User interaction

After decomposing the problem that a user would need to be able to add a note, edit and save to a calendar. A activity diagram was thought about to consider the flow of the application.

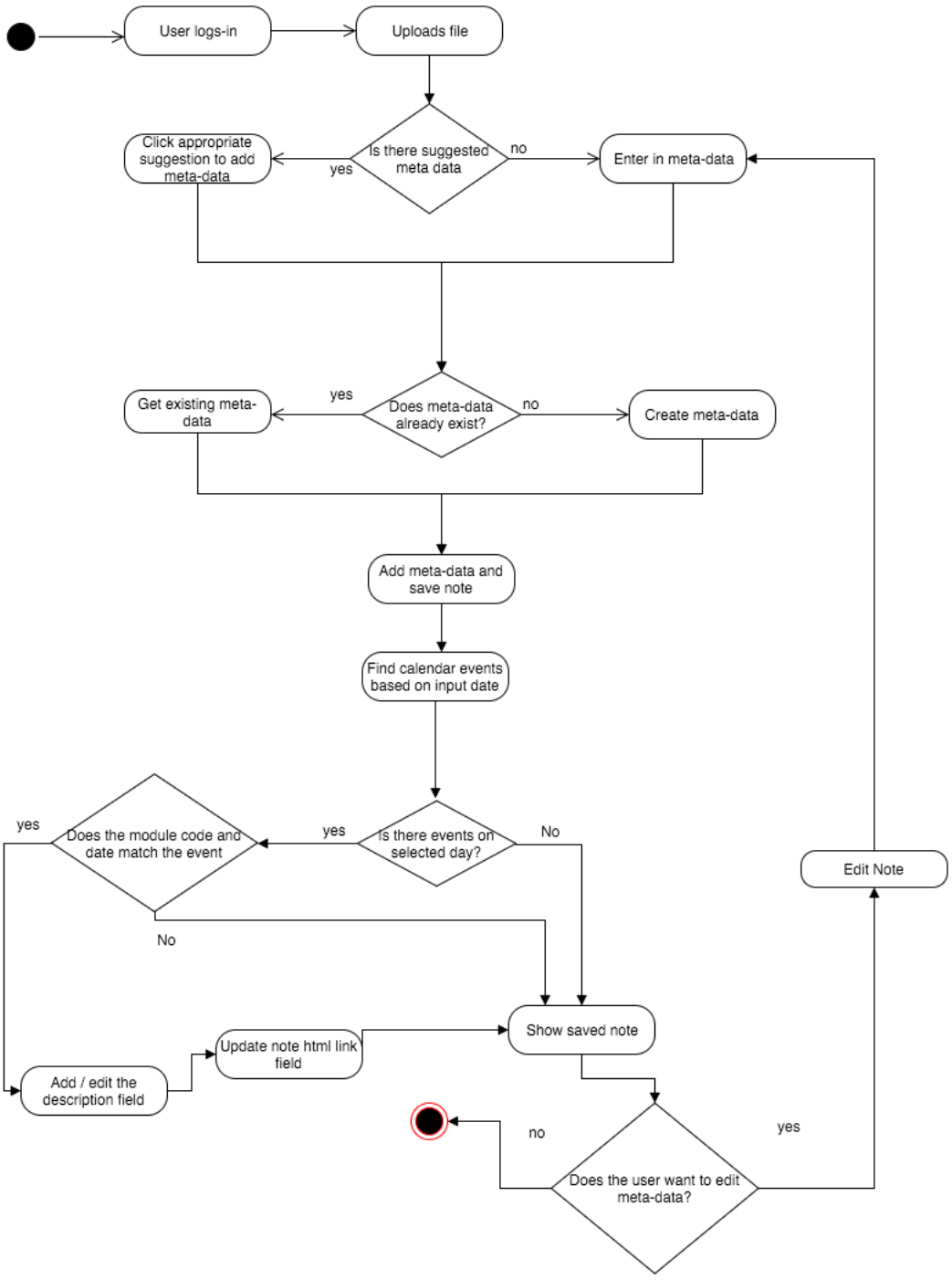


Figure 1.2: An activity diagram to show how to save a note and the integrations with the calendar too.

Over the series of sprints, the design for the activity diagram expanded and grew. The resultant is depicted in Figure 1.2. To begin with the user activity where the user logs in was not incorporated until the thought of expanding the application for further users was considered.

Additionally the conditional check to see if there was meta-data suggested, from which the user could click on the meta-data to auto-populate the field content, has been developed into the

final design. Instead, of the conditional check just an activity was selected to enter in the meta-data.

Overall, the activity diagram shown shows the final output of how a note is added into the system. This design has been meticulously developed incrementally to show the final output. Each of the iterations would increase the functionality, and as a result show the final output.

1.1.4 Model-view-controller

The application would be designed in an model-view-controller (MVC) approach, so it would be beneficial to show the designs which would show how the system interacted with different components.

1.1.4.1 About MVC

MVC is a design pattern where logic is differentiated from presentation layers, as shown in Figure 1.3.

The controllers aim is to not directly integrate with database and business logic, instead it interacts with a series of models and services. Finally, the controllers will help to request to render specific view files with dynamic content.

The model in the MVC structure has no acknowledgement of the view file. Instead of rendering any form of HTML in the model it is purely data-driven. The sole purpose of the model is to interact with the database and perform any business logic that does not fit in the controller and the view file.

Finally, the view files contain HTML logic with dynamic content passed to the file from the controller. There may be specific logic which impact the HTML displayed, but no direct calls are made to the database layer or the controller. It uses the dynamic content passed in.

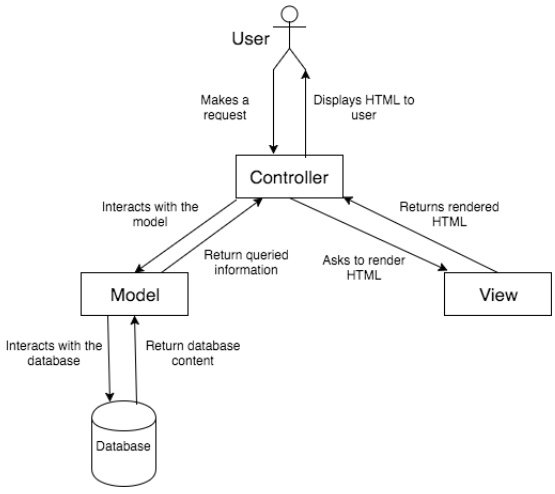


Figure 1.3: A example of how the model-view-controller(MVC) framework integrates.

1.1.4.2 Structuring the web application

Although all the files could not be identified in the design section, the overall structure of the application has been considered.

The primary aim for considering the design of the application would allow the application to reuse aspects of the codebase where appropriate. A module based design was considered, where each section of functionality was its own module - but this was rejected as it felt like the codebase became obfuscated. Therefore the MVC approach was adopted.

The framework chosen, see section **??**, does not support MVC structures out of the box. Routes are expected to be placed in a singular file; this philosophy is carried through to the models. This was not chosen as the structure of the application as it reduces the clarity of what the code was supposed to do by over-complexing where dependencies between different classes are supported.

To overcome this, Blueprints were used. These are essentially controllers, like those found in Ruby on Rails. Annotations were used to define a route and a blueprint was associated to one file.

Models were separated into their own directory, and a one-class per file was adopted to keep the design clean and simple. This ensured the design of the classes was considered before creating such files.

It is worth considering the view files. The view files were the only section of the web application structure which underwent and iterative process. Initially, the view files would represent the entire DOM tree in a singular file (duplicating headers, scripts etc). However, this is not optimal.

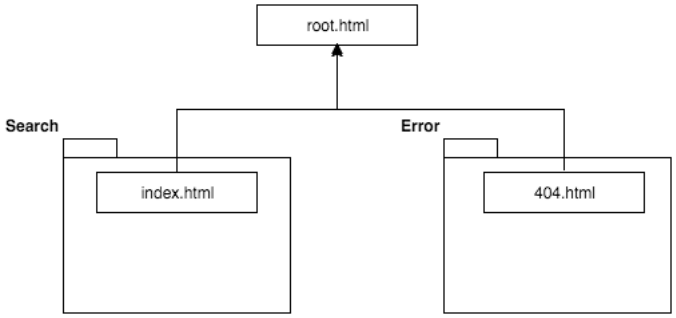


Figure 1.4: A diagram illustrating how extension in Jinga html template engine works.

Figure 1.4 shows the result after the sprint which the design was improved upon. All template files now extend “root.html”, overriding the “content” block. This ensures that the do not repeat yourself (DRY) principle is adhered to and HTML, such as the navigation, are only declared once.

1.1.4.3 Constructing URLs

Often overlooked when considering a design is the URL structure. The design not only aids the developer, but the user using the page to clearly know the intention of that page. Typically there are two types of URLs RESTful-like and query strings.

During the iterations, especially when new functionality was being considered, specific routes were thought about carefully. In the search user-story, standard procedure was followed. Query strings create URLs such as: `/search?module_code=cs31310`; representing the query string

as key-value pairs. During the search feature, it was decided that this approach would be adopted so that the user can easily bookmark the page - as well as creating a common URL structure.

RESTful urls help to show the a hierarchy of content. Exposing a user to such URL helps them to clearly identify their content. In iteration for displaying a note `/show_note/1`, was decided for the URL; it is easier to read than `/show_note?note_id=1` - instantly showing to the user the point of the page.

For the user task viewing notes, it was worth noting that traditional RESTful URL's would be changed for readability. `/view_notes/` was used, when a proper RESTful url may be `/notes/`. This offered more semantic meaning to the URL structure.

It was worth considering the URL structure and way in which the application displays the content to a user, without it a user may be left confused at what the page is trying to convey.

1.2 Image processing

In the very early sprints, the image processing design went through several substantial iterations. Each of the tasks relating to the user story to binarise an image, had design implications.

Early work was conducted to investigate how to prepare images for the Tesseract engine. Imagemagick was initially used by converting the image to grayscale - but this yielded poor results. After further design decisions were made to convert the image to Monochrome it was decided that the next iteration produced a better image segmentation pipeline.

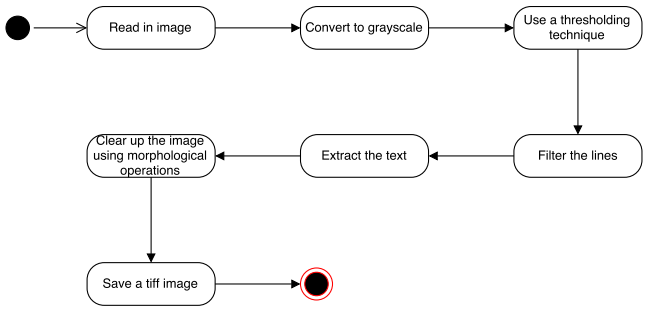


Figure 1.5: An activity diagram to depict the design of the algorithm for the image segmentation.

Figure 1.5 shows the overall activity of how the image processing will be intended to be implemented, after early design work showed binarisation was more complex. Further descriptions of specific implementation can be found in the implementation section.

This high-level activity diagram was the result of continuous design in the first few sprints. Initially a design was drawn up to just binarise the whole image, but due to implementation issues, this caused too much noise. Therefore, a new algorithm had to be established.

The blue-lined paper was one way to overcome that. Filtering the lines from a solid lined paper, would ensure less noise was on the image, creating a better binarised image. Overall, the activity diagram depicts the algorithm of taking a mobile phone photo, filtering the lines, binarising the image and extracting a tiff image. The tiff was selected as a design decision, as Tesseract input requires a tiff file.

This design initially considered blue to be important, but it was produced too much noise in the implementation. As a result, instead of trying to extract the lines, it was decided that the lines should be filtered and we should only extract the text. This was the final iteration of development on the binarisation script.

1.3 Tesseract

Early on in the sprints Tesseract was identified as the OCR tool of choice. Evaluations into the different tools available: [CITE - Case study] performs a case study using Tesseract as their OCR tool to analyse printed text in an image. Patel et al, also discuss the comparison against a proprietary OCR tool, Transym [CITE].

The results concluded by Patel et al is that Transym only yielded a 47% accuracy on 20 images compared to 70% accuracy using the Tesseract engine.

The first few iterations gave a concrete overview that the first three lines of the text would be parsed, forming the information for the meta-data. Appendix **??** shows the layout of the three lines.

It is worth acknowledging that the test-data used for the Tesseract training had a design element attached to it. When considering what the test data should consist of, there had to be a variety in the data. Panagram's, the quick brown fox is the most common, is a good way to represent text as it contains all alphabetical characters. [CITE website used]. This would give Tesseract the best possible chance at learning different characters - due to there being an abundance of each letter.

1.4 Entity-relation design

Whilst creating the CRC cards, it allowed considerations to be made about the relations and how they are connected. Through each user-story analysed it was reflected upon if that would affect the entity-relation design.

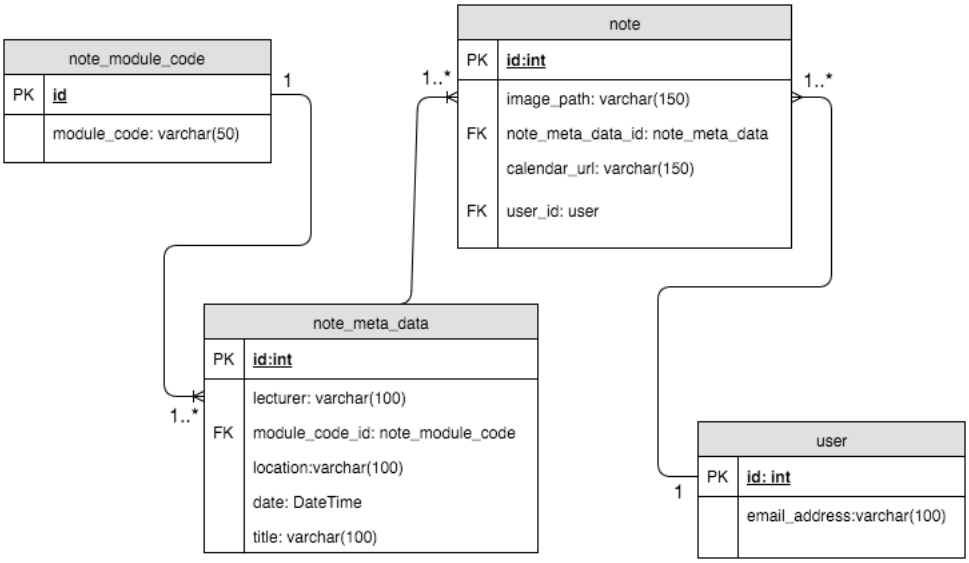


Figure 1.6: The final result of the entity-relation diagram. After a series of iterations.

Figure 1.6, shows the output from the result of continuous design on the database diagram through the iterations. The design per iteration matched that of the story. For example, although Meta-Data was known to be added into the system, this was not added until that story was brought into the sprint - instead just a Note relation was created.

1.4.1 Justification of design

Below is a justification of the designs through the iterations.

1.4.1.1 Note

During design persisting the note was one of the first entity-relation design decisions which was made. The attributes selected for the `Note` relation best justify what a note consists of. Firstly, the note contains an image link, which is a relative path to the image. This was persisted to ensure that it could be found easily. When the story for implementing user's was actioned, an additional field containing the user's id was added to the relation.

During the implementation of adding a URL to a calendar event, the calendar URL was persisted to the database of the associated note. The event ID could have be saved, but the URL was decided to stored so additional queries were not made to the external service. Furthermore, a note will only have one URL.

When implementing the note's meta-data, a relation was created and the foreign key was added to the note relation. This was created to ensure that a note must have associated meta-data.

1.4.1.2 Note_meta_data

The `note_meta_data` relation was created in its own relation to reduce data-redundancy, following the principle of normalisation in relational databases. The content could be duplicated for multiple notes, if a user tags the same meta-data to more than one note. As denoted from the relationships: a note will have a singular meta-data item, but the meta-data item could have many notes.

With attributes lecturer location and datetime - these were the initial design decisions made to be included in the meta-data. However, in a later iteration it was decided a title would be preferable; this was added to the relation. The date field is a date-time instead of a string due to integration with the calendar requires specific date-time strings, making it easier to parse.

Initially developed with the module code in this relation, in subsequent iterations the module code was extracted and a foreign key was used.

1.4.1.3 Module code

The module code was developed into its own relation to prevent data-redundancy. A user may enter multiple notes for the same module code - as a result the database would only need to include one reference of that module code. The relationship between the meta data and the module code is explicit: the meta data must contain one module code but the module code can have more than one meta-data item.

1.4.1.4 User

This was not added to the application until around sprint 5. Details regarding oAuth can be found in **??**. However, the user will have an email address and that would be stored. It is in its own relation from a logic perspective: every time a user signs up to the system they are not creating a note instantly, therefore a relation was created to separate this logic. The foreign key was added to a note, so that a note can only have one user - and a user can have multiple notes.

Overall, a succinct collection of relations have been developed which aim to solve the issues of the data-redundancy, by providing solid rationale for the resulting design.

1.5 Sprints

1.6 User interface

With the web application being a core part a series of user interfaces (UI) was collated at the start of each breakdown of the story - if there was a significant change in the UI.

The user interface had make the web application feel like an application, rather than a traditional website. This was identified from the background analysis where many systems felt like an application. The colour scheme was aiming to be simplistic, using the Google colour style guide. An alternative of bootstrap was considered, instead of designing bespoke CSS. Although it

has a built-in responsive theme, due to the over-kill of the additional files a simpler approach was adopted.

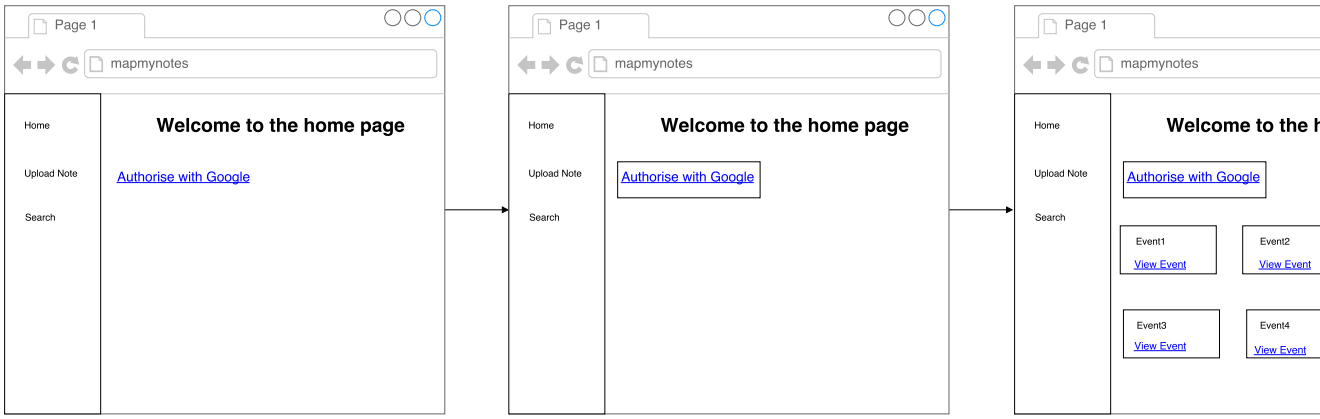


Figure 1.7: From left to right, the homepage wiremock through the different iterations and the change of requirements

Figure 1.6 shows the exploratory wiremock design completed prior to the UI interface. From the early iterations it was just an authorise button, then as a requirement was added to show the user events from the last seven days it was mocked up to reflect this.

Further mockups available in Appendix **??**.

1.7 Implementation tools

The following sections discuss the implementation tools and their purpose within the application.

1.7.1 Programming language

The programming language would not change per sprint or over an iterative process - as a result this was identified in sprint 0, where additional spike work was completed.

As a web application was being developed investigatory work was completed into the analysis of suitable server-side languages. Typically traditional server-side applications language are: PHP, Ruby, Python, C#, Java and JavaScript, which has increased in popularity [46].

Decomposing the analysis in the early sprints determined that OpenCV would be utilised on the project. OpenCV's source code is written in C++, however Python and Java bindings are available. Additional research was conducted to see if a reliable wrapper for either PHP or Ruby was available, and after a lot of investigation it was concluded there was not.

C++ is not considered a standard web application development language removing it as a viable option for the web application. Java applications are predominately large commercial applications, using a range of enterprise software - often renowned for their performance abilities [33]. This approach felt too cumbersome for a proposed light-weight application.

By being constrained by design decisions to use OpenCV and a reluctance to use Java, then Python was selected as the most suitable language. Python offers a lightweight and an easy to learn syntax that produces readable code, allowing a object-oriented paradigm to be followed. Additionally, its support for OpenCV is sufficient for the application.

1.7.2 Framework

As Python was being used as the language of choice, this narrowed down the frameworks available. Frameworks are useful for handling more complex features like routing and session handling - leaving the developer to focus on more domain specific issues. Exploratory work was completed in the early sprints to find a suitable tool. The frameworks Django [10], Flask [14] and Bottle [6] were evaluated.

Some frameworks constrain the developer's to specific implementations through abstracted classes whereas some offer more flexibility. Whilst evaluating Django, a full framework, it was concluded that such a large framework was excessive for this application and rejected as a choice for the framework.

Flask and Bottle are classified as "micro-frameworks", offering a lightweight structure, allowing developers to have more control on the structure. On face value, Flask and Bottle appear to be very similar; they are both lightweight with a similar syntax. After evaluation both frameworks it was concluded that Flask has a larger support community compared to Bottle - along with more reliable documentation.

As a result, Flask was chosen as the framework which will be used throughout the application. Spike work was completed into evaluating Flask's viability for the application and was decided to be a good choice.

1.7.3 Continuous integration tools

Continuous Integration (CI) is normally used in development teams to ensure that all code is checked into the repository. As it was changed for a single person project, so did the point of using it; it was used to ensure every commit passed all tests when pushing to the repository.

After identifying CI would be used in the analysis stage, an appropriate tool would have to be chosen. Jenkins was an initial choice; it is a standalone Java application which a repository can be synced to.

Travis CI, is a CI tool in the cloud which can be synced to a GitHub repository. Tests can be run during every commit of the application and details regarding if it errors, passes or fails is available.

A key design decision was to be able to check the build process quickly. With Travis it has a web interface clearly showing the output. Jenkins would require additional set up of specific build scripts for every branch in the application. Travis would be easier to set up and would easily integrate with the application.

1.7.4 Version control

Version control was used on the project to ensure that code was under specific versions. The project was created on a private Git repository on GitHub. Git was chosen for its familiarity and GitHub is a well known place for handling Git based solutions; Travis CI integrated well with GitHub.

It is worth making a mention on the Git flow which was used. As each story was implemented a branch would be created in the form of: `feature/<summary_of_story>`, such as `feature/logout`. Branches were created from the development branch. All code was pushed to its own separate branches, and only when Travis CI passed a pull request was created on GitHub. Once Travis had successfully passed all the pull request tests, and it was safe to merge then the branches were merges into development.

1.7.5 Development environment

The text-editor, Atom, was used for the majority of the project. It was a lightweight tool which accommodated for Python syntax. However later in the project, when refactoring became more cumbersome due to the increase in code base - PyCharm community edition was used as it offered better refactoring functionality.

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

Testing Results

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

3.1 Unit tests

3.1.1 Binarise image

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

3.1.2 Calendar item

3.1.3 DateTimeHelper

3.2 Acceptance tests

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

3.2.1 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 C.2: Acceptance test being conducted for the homepage, to ensure that the homepage displays the correct content.

3.2.2 Add meta-data

```
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 C.3: Acceptance test being performed to ensure that meta-data can be added to the correct note.

3.2.3 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 C.4: Acceptance test being conducted so that a note's meta-data can be edited successfully.

3.2.4 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 C.5: Acceptance test to ensure that a user can search for a module code and it displays their notes.

3.2.5 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 C.6: Acceptance test being conducted to ensure that all the notes can be viewed.

3.2.6 Show a note

```
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 C.7: Acceptance test being conducted to make sure that a singular note can be viewed correctly.

3.3 Integration tests

3.3.1 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 C.8: Integration tests carried on the add and edit meta url to ensure the system worked well together.

3.3.2 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 C.9: Integration tests conducted on the homepage to ensure that the routes were accessible.

3.3.3 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 C.10: Integration tests conducted for the logout route ensuring the routes are logged out.

3.3.4 Oauth

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

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

Figure C.11: Integration tests conducted for the oAuth route which interacts with the Google API.

3.3.5 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 C.12: Integration tests conducted for the search URL to ensure searching works correctly.

3.3.6 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 C.13: Integration tests implemented to ensure that the note can be displayed properly.

3.4 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 C.14: Integration tests implemented to ensure that a user can upload their images to the application.

3.5 User

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

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

Figure C.15: Integration tests implemented the user route is working correctly and a the user gets added to the database.

3.6 View all notes

```
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 C.16: Integration tests to make sure the view all notes url is working and getting the appropriate notes from the database.

3.7 User tests

Appendix D

Tesseract data results

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

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

Appendix E

Example test data

5.1 Calendar week response mock


```
{
"accessRole": "owner",
"defaultReminders": [
  {
    "method": "email",
    "minutes": 30
  },
  {
    "method": "popup",
    "minutes": 30
  }
],
"etag": "\"1234567891012345\"",
"items": [
  {
    "kind": "calendar#event",
    "etag": "\"1234567891012345\"",
    "id": "ideventcalendaritem1",
    "status": "confirmed",
    "htmlLink": "https://www.google.com/calendar/event?testtest",
    "created": "2014-09-10T14:53:25.000Z",
    "updated": "2014-09-10T14:54:12.748Z",
    "summary": "Test Example",
    "creator": {
      "email": "test@gmail.com",
      "displayName": "Tester",
      "self": true
    },
    "organizer": {
      "email": "test@gmail.com",
      "displayName": "Test",
```

```
        "self": true
      },
      "start": {
        "dateTime": "2016-12-01T01:00:00+01:00"
      },
      "end": {
        "dateTime": "2016-12-01T02:30:00+01:00"
      },
      "transparency": "transparent",
      "visibility": "private",
      "iCalUID": "123456789@google.com",
      "sequence": 0,
      "guestsCanInviteOthers": false,
      "guestsCanSeeOtherGuests": false,
      "reminders": {
        "useDefault": true
      }
    }
  ],
  "kind": "calendar#events",
  "nextSyncToken": "synctokenasbebebe=",
  "summary": "test@gmail.com",
  "timeZone": "Europe/London",
  "updated": "2016-03-16T15:13:26.416Z"
}
```


5.2 Google plus response mock


```
{
  "tagline": "Some Dummy data taglone",
  "verified": "False",
  "circledByCount": 100,
  "objectType": "person",
  "emails": [
    {
      "type": "account",
      "value": "test@gmail.com"
    }
  ],
  "occupation": "A Test Occupation"
}
```

Appendix F

Image Processing

6.1 Pre-blue lined image

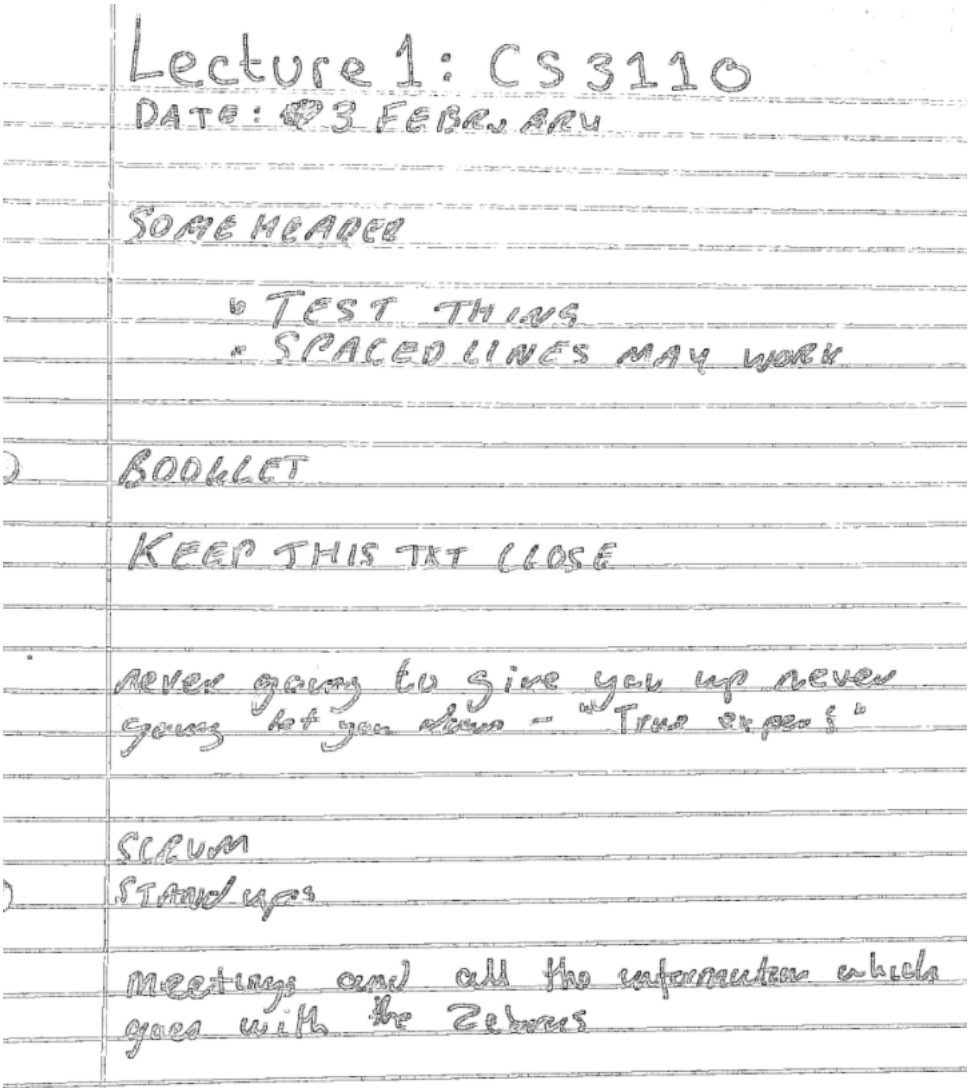


Figure F.1: The adaptive threshold on normal lined paper caused too much noise to be interfered with the Tesseract engine

6.2 Filtering the blue lines

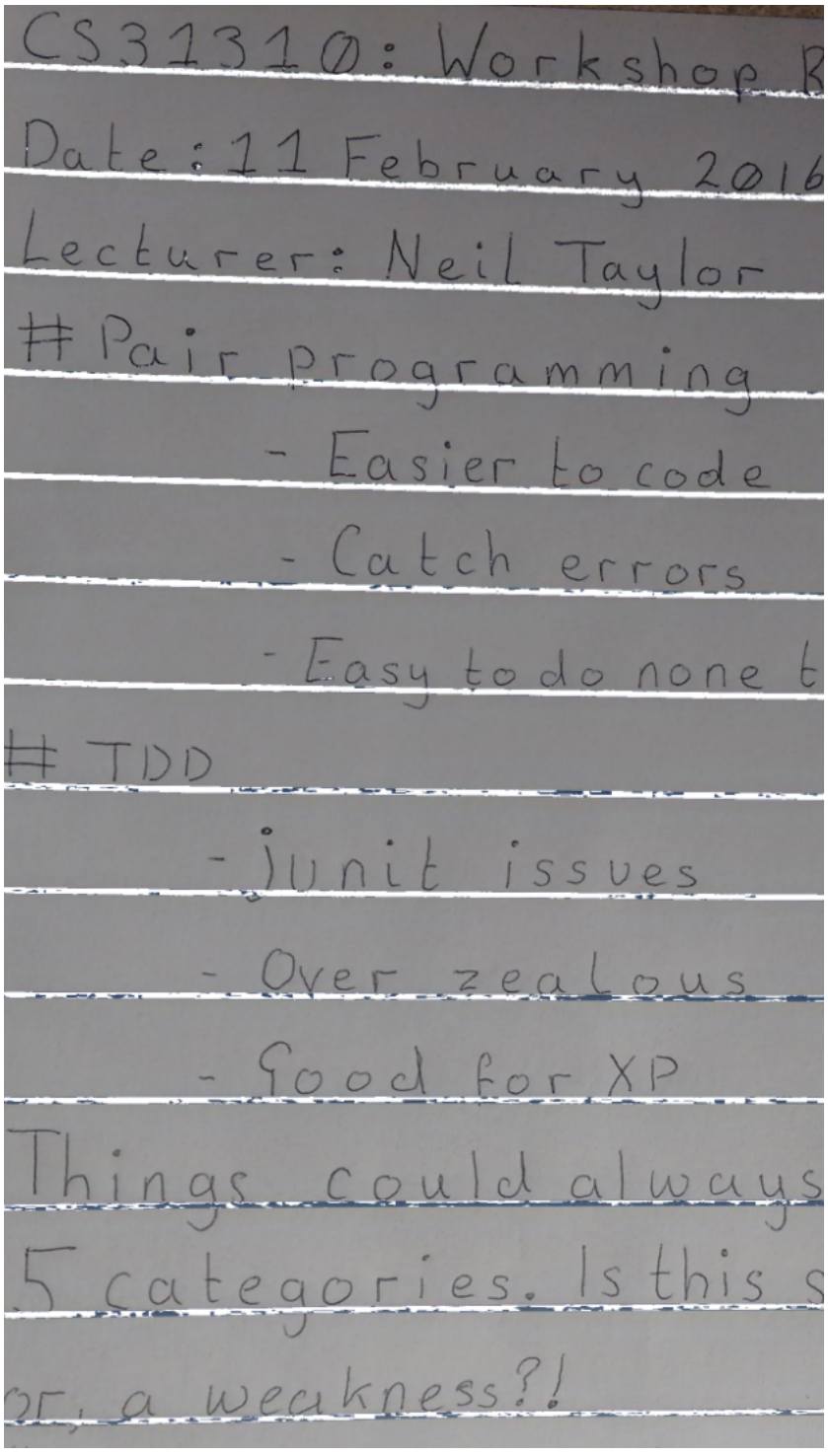
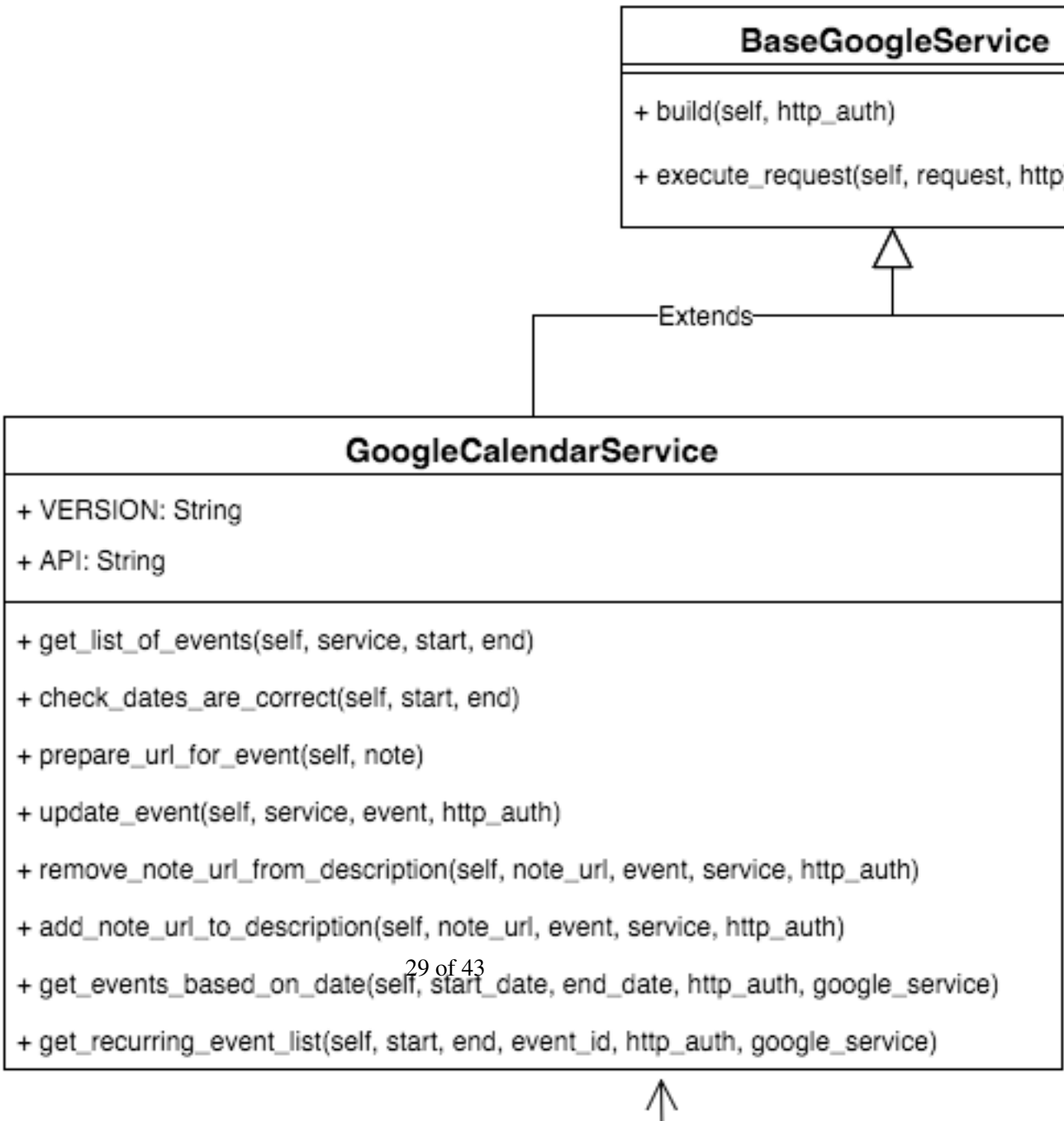


Figure F.2: Blue lines in the adaptive threshold have been identified and removed to be a white colour.

Appendix G

Design decisions

7.1 Class diagram

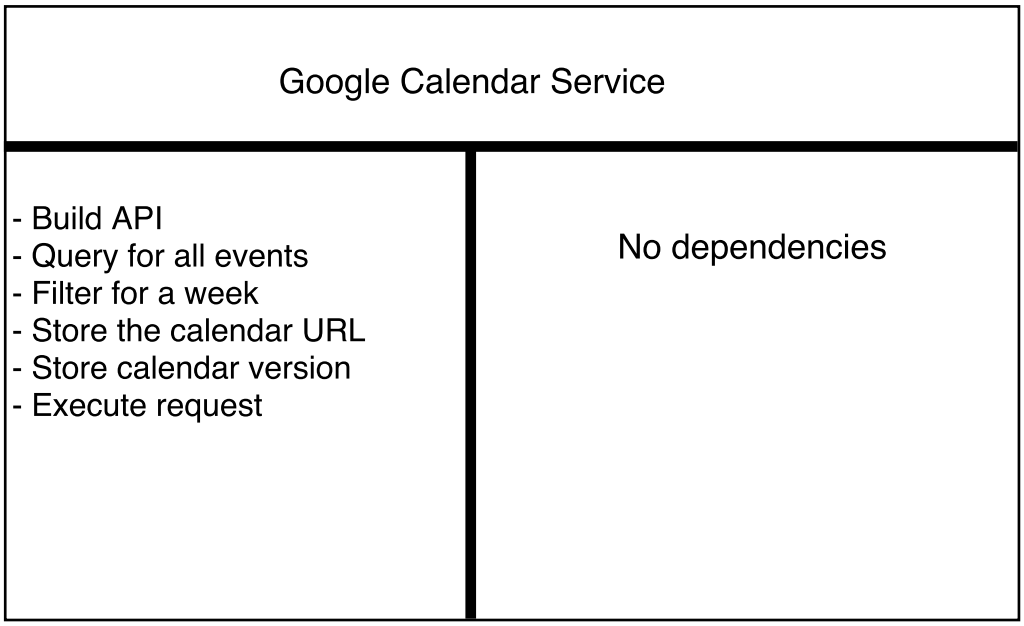


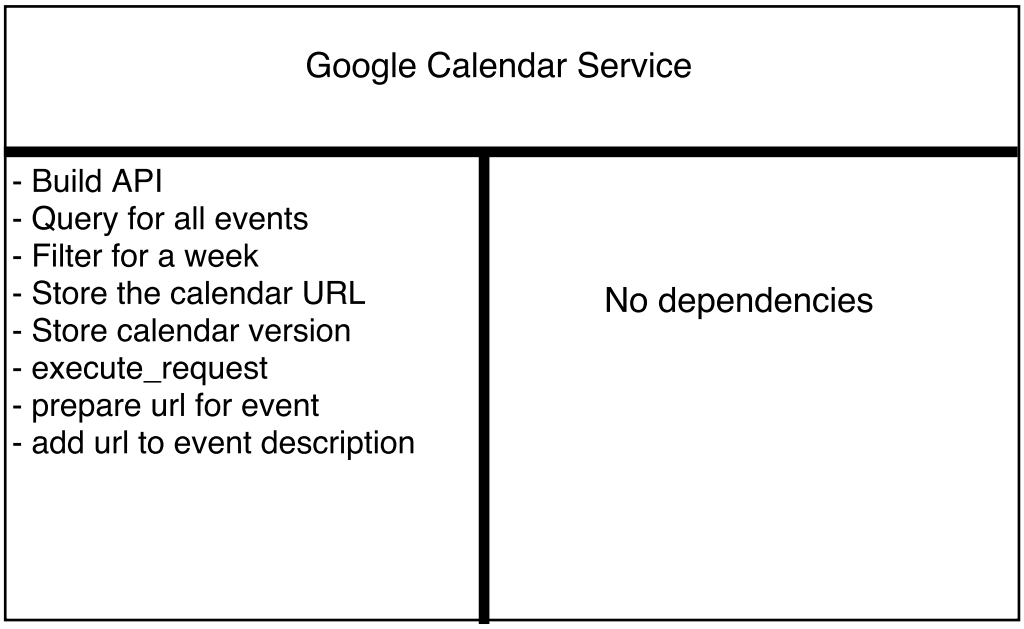
Appendix H

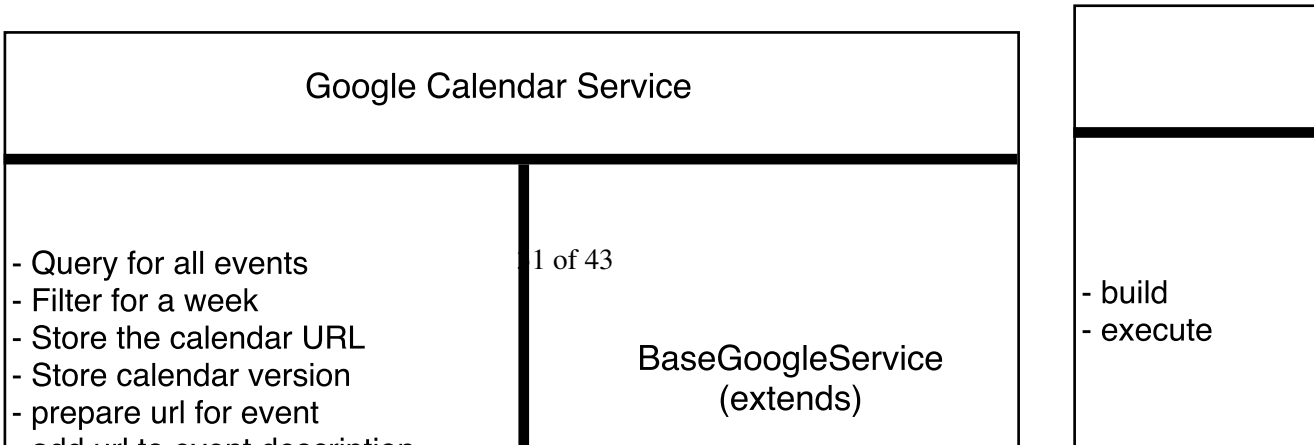
Design suppliments

8.1 CRC cards

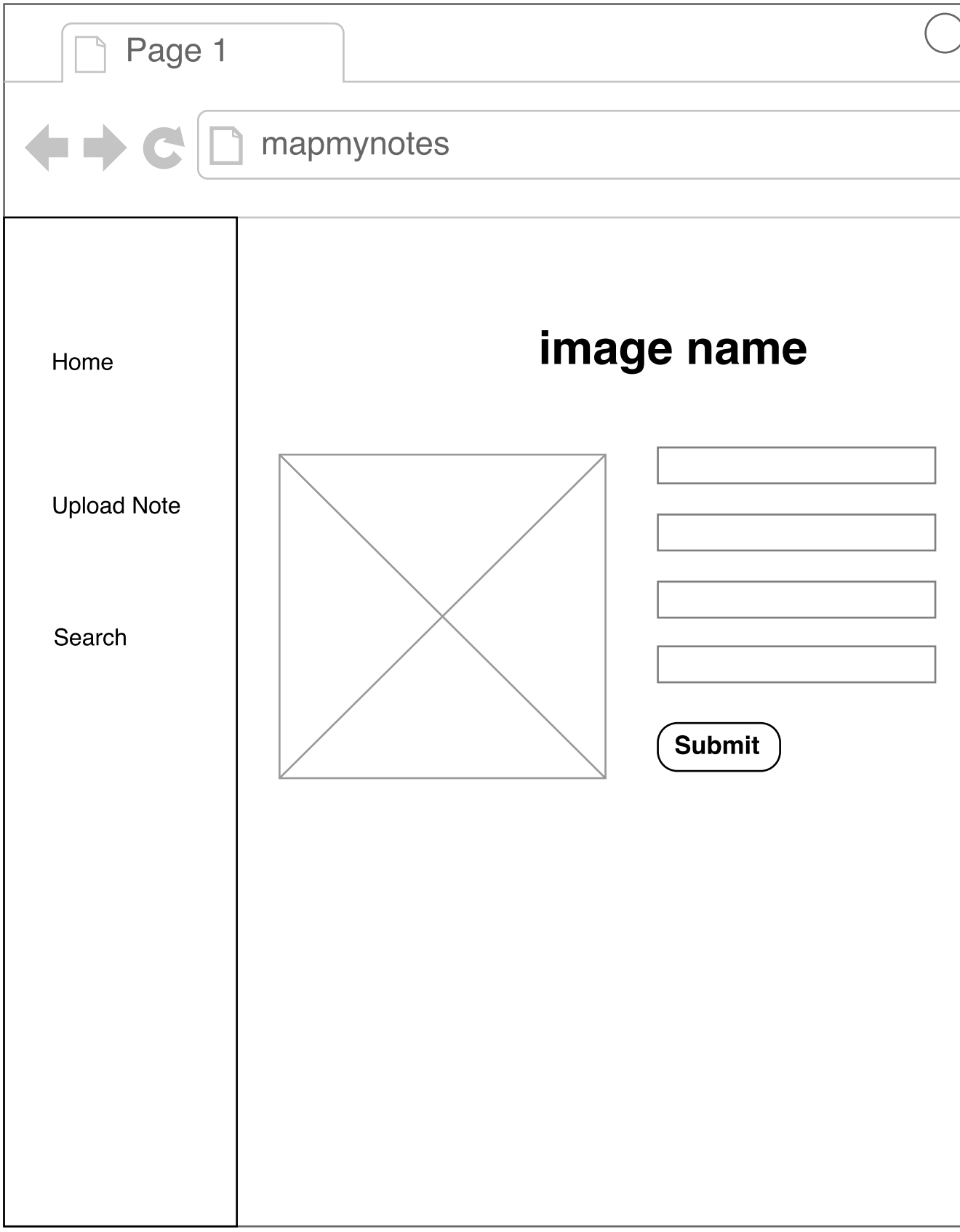
Below is an excerpt of the the examples of a more complex CRC card design in the system. Throughout the the project, each class went through an iterative process of using CRC cards. Therefore, a lot of them have been omitted to save space.


Google Calendar Service
- Build API
- Query for all events
- Filter for a week
- Store the calendar URL
- Store calendar version
- Execute request
No dependencies


Google Calendar Service
- Build API
- Query for all events
- Filter for a week
- Store the calendar URL
- Store calendar version
- execute_request
- prepare url for event
- add url to event description
No dependencies


Google Calendar Service
- Query for all events
- Filter for a week
- Store the calendar URL
- Store calendar version
- prepare url for event
BaseGoogleService
(extends)
- build
- execute

8.2 Wireframes



Appendix I

Scrum process supplementary materials

The appendix discusses some of the additional material to show the process of scrum used as the methodology of choice. Below is a collection of user-stories throughout the sprints.

9.1 Sprint burndown charts

Figure I.1: An example of the burndown chart for a sprint, showing areas where there may have been difficulty.

9.2 Overall burndown chart

Figure I.2: The overall burndown of the sprints during the development period. This clearly shows a consistent work flow up until more knowledge of the project was achieved, going below the expectation line.

Id	User story	Sprint	Story points
1	As a user I want to be able to upload an image of a set of notes so that I can see my note in the application	2	10
2	As a user I want to be able to tag my notes so that all my notes are under the correct module	4	5
3	As a user I want to be able to add information about the notes so that I can reference them in the future	4	15
4	As a user I want to be able to save a note, so that I can find it again later	3	10
5	As a user I want to be able to search for a given module, so that I can find all notes for that module	7	8
6	As a user I want to be able to sign in via google sign in	5	15
7	As a user I want to use Tesseract OCR so that I can identify characters	1	15
8	As a user I want to be able to view the application on a website	2	5
9	As the customer I want to see the image being binarised properly	2	10
10	As a developer I need to train my handwriting, so that Tesseract can recognise my handwriting	10	n/a
11	As a user I want to be able to edit the meta data, so that I can update it in light of a change	5	5
12	As a user I want to be able to remove a note incase I do not want it to appear	5	5
13	As a developer I want to the website to have good styling	4	8
14	As a developer I want to integrate tesseract into the application, so it can read information from a note	8	8
15	As a user I want to be able to view all the notes I have as a user so I can easily find all of them again	6	3
16	As a user I want to view a list of events on the homepage from my calendar, so I can see recent events	6	15
17	As a user I want to be able to save the URL in the calendars event	7	10
18	As a user I want to be able to tag the title of the lecture, so that I can know which one it is.	6	5
19	As a user, when I authorise I want to show my email address and remove the authorise button, so I know I have signed in	6	3
20	As a developer I want to be able to get the date taken from EXIF data, to show information about a note	7	8
21	As a user I want to be able to edit the date and update my calendar	9	8
22	As a developer I want to be able to associate a note with a user	7	5
23	As a user, I want to be able to have automated suggestion of meta data from the image, so that I can know what to tag.	8	5
24	As a user, I want to be able to logout, so that I can close my session	8	5
25	As a user I want to be able to click Tesseract Items, so that it's easier for me to put in the fields.	9	10
26	As a user I want to be able to edit and save to reoccurring events	10	10

Table I.1: A table showing the user stories identified throughout the project, along with the sprint in which they were implemented and associated story points

Annotated Bibliography

[1] "sirfz/tesserocr: A Python wrapper for the tesseract-ocr API," last checked 25th April 2016. [Online]. Available: https://github.com/sirfz/tesserocr

The Tesseract wrapper which was used to extract the data from the image. It gives the confidences and all the words associated to the lines.

[2] "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan. 1979. [Online]. Available: http://dx.doi.org/10.1109/tsmc.1979.4310076

The original paper which OTSU is represented. Although a bit mathematical, some bits were good for reference material on how the algorithm works.

[3] "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

[4] "OpenCV: Extract horizontal and vertical lines by using morphological operations," 2015, last checked 25th April 2016. [Online]. Available: http://docs.opencv.org/3.1.0/d1/dee/tutorial_moprh_lines_detection.html\#gsc.tab=0

A great reference on how to use different morphological operations and adaptive threshold techniques to extract and binarise an image. Used extensively with the image segmentation script.

[5] 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.

[6] 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.

[7] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008, pp. 138–139.

A book which explains how the Gaussian function for the adaptive threshold with OpenCV works. It gives a simple description, one which is easy to follow.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] Fisher, "Point Operations - Adaptive Thresholding," 2003, last checked 25th April 2016. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/adpthrsh.htm

An article explaining clearly and simply how the adaptive thresholding algorithm works. It gives a good level of detail and is concise in its points.

[14] 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.

[15] ——, "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.

[16] 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

[17] 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.

[18] 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.

[19] R. Gouldsmith, "build throws KeyError: 'rootUrl' error on Google Calendar API Issue #208 google/google-api-python-client," https://github.com/google/google-api-python-client/issues/208, 2016, last checked 25th April 2016.

A issue which was raised by the author, regarding an issue experienced with a 3rd party library.

[20] ——, "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.

[21] A. Greensted, "Otsu Thresholding - The Lab Book Pages," http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html, June 2010, last checked 25th April 2016.

A great reference tutorial aiding to identify what OTSU threshold is and how it works in a simple to understand manner, with plenty of example.

[22] 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.

[23] 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.

[24] 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.

[25] 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

[26] ——, "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.

[27] ——, "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.

[28] 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.

[29] 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.

[30] H.-F. Ng, "Automatic thresholding for defect detection," *Pattern Recognition Letters*, vol. 27, no. 14, pp. 1644–1649, Oct. 2006, last checked 25th April 2016.

This paper was interesting as it aided in the dicussion of the different thresholding algorithms. It was good to reaffirm some knowledge gained during the development process.

[31] 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.

[32] Opencv, "Miscellaneous Image Transformations OpenCV 2.4.13.0 documentation," 2016, last Checked 25th April 2016. [Online]. Available: http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html

A description of the adaptive threshold function, which shows that there are two different functionc can be used.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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

[39] 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

[40] 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.

[41] M. Sturgill and S. J. Simske, "An Optical Character Recognition Approach to Qualifying Thresholding Algorithms," in *Proceedings of the Eighth ACM Symposium on Document Engineering*, ser. DocEng '08. New York, NY, USA: ACM, 2008, pp. 263–266. [Online]. Available: http://dx.doi.org/10.1145/1410140.1410197

A great paper which discusses the Tesseract engine by HP researchers. It is used to discuss the idea that OTSU is used as its pre-processing step.

[42] 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.

[43] 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.

[44] 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.

[45] R. Viet OC, "jTessBoxEditor - Tesseract box editor & trainer," last Accessed 6th February 2016. [Online]. Available: http://vietocr.sourceforge.net/training.html

An excellent software package which allows the user to train the box files with a great graphical user-interface.

[46] 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

[47] 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.

[48] 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.