The Development of EyeMap 2.0

EMLYN HEGARTY-KELLY

A thesis submitted for the degree of

Master of Computer Science

Department of Computer Science

Maynooth University

February 2020

Supervisors: Dr. Aidan Mooney & Dr. Susan Bergin

Head of Department: Dr. Joseph Timoney

CONTENTS

I INTRODUCTION AND BACKGROUND MATERIAL 1

1 INTRODUCTION & BACKGROUND 2

1.1 Introduction . . . 2
1.2 Motivation . . . 2
1.3 Goals . . . 3
1.4 Research Questions . . . 4
1.5 Contributions . . . 5
1.6 Thesis Overview . . . 5

2 RELATED WORK 7

2.1 Introduction . . . 7
2.2 Eye-Tracking . . . 8
2.2.1 What is Eye-Tracking? . . . 8
2.2.2 Eye-Tracking & Reading . . . 11
2.3 Fonts & Font Metrics . . . 12
2.4 Software Development . . . 13
2.4.1 Stages of the Software Development Cycle . . . 14
2.4.2 Software Development Models . . . 17
2.4.3 Software development Process Used . . . 24
2.5 User Participation . . . 25
2.6 Summary . . . 27

3 EYEMAP 29

3.1 Introduction . . . 29
3.2 What is EyeMap? . . . 29
3.3 EyeMap - Architecture . . . 30
3.4 EyeMap - Key Features . . . 32

3.4.1 Data Parser 32
3.4.2 Data Loader 35
3.4.3 Data Visualisations 38
3.4.4 Data Editor 39
3.4.5 Reading Variables 40
3.4.6 Data Exporter 40
3.5 The need to Redevelop EyeMap 40
3.6 Summary 41

II SOFTWARE REDEVELOPMENT 43

4 USER PARTICIPATION & SOFTWARE REQUIREMENTS 44
4.1 End User Team 44
4.2 Prototype 46
4.3 Team Requirements Meeting 48
4.4 Requirements 49
4.4.1 Requirement Features 50
4.4.2 requirement fixes 54
4.5 EyeMap 2.0 Architecture 55
4.5.1 Django 56
4.6 Software Development Tools 56
4.6.1 Version Control 57
4.6.2 Integrated Development Environment 58
4.6.3 Error Logging and Bug Tracking 59
4.7 Summary 60

5 SYSTEM REDEVELOPMENT 61
5.1 Challenges to Overcome 61
5.2 RAD Task List 61
5.3 Constructing the core architecture of EyeMap 2.0 63
5.3.1 Languages used in the development of EyeMap 2.0. . 63
5.4 GitLab Repository 64
5.5 Design and development of EyeMap 2.0's layout 64

5.6 Database creation . . . 65
5.6.1 Data Files Management . . . 66
5.6.2 Database structure . . . 67
5.6.3 Generating the tables for the database . . . 68
5.7 Set up Templates . . . 68
5.8 Web Page Creation . . . 69
5.9 Data Loader . . . 70
5.9.1 Processing the Trials File . . . 72
5.9.2 Processing Data Files . . . 73
5.10 Data Visualisation & Editor . . . 74
5.10.1 Extracting the AOI Information . . . 74
5.10.2 Display the Data . . . 75
5.11 Data Exporter . . . 76
5.12 Additional Work . . . 80
5.13 Summary . . . 81

III EVALUATION & FUTURE WORK 82

6 EVALUATION 83
6.1 System Testing . . . 83
6.2 Requirements checklist . . . 90
6.3 End User Evaluation . . . 93
6.4 Summary . . . 98

7 CONCLUSIONS & FUTURE WORK 99
7.1 Conclusions . . . 99
7.2 Future Work . . . 102

Bibliography 104

Appendix 110

A EYEMAP 2.0 WEB APPLICATION 110
A.1 EyeMap 2.0 web application . . . 110

B COMPLETE LIST OF CALCULATED VARIABLES 111
B.1 Complete list of calculated variables . . . 111

C FURTHER IMPLEMENTATION DETAILS 115
C.1 Constructing the core architecture of EyeMap 2.0 115
C.2 Generating the tables for the database 115
C.3 Set up Templates 118
C.4 Web Page Creation 121

LIST OF FIGURES

Figure 2.1 An example of the eye-tracking camera view on an eye [52]. 8
Figure 2.2 The EyeLink 1000 being used for an experiment with chess. 9
Figure 2.3 The RED-m from SMI on a laptop computer. 9
Figure 2.4 Tobii pro glasses used in a driving experiment. . . . 10
Figure 2.5 Heat-Map from SR Research of fixations on an image. 10
Figure 2.6 Fixations and saccades during reading. Note: saccades are drawn like this to show the hop or jerk that is involved with a saccade between fixation points 11
Figure 2.7 Diagram showing the viewing angle of the fovea, parafovea and periphery 12
Figure 2.8 Proportional font v monospace font [45]. 13
Figure 2.9 Font Metrics [15]. 14
Figure 2.10 The Software Development Process as shown and discussed in the report from the NATO Software Engineering Conference 1968 [32]. 15
Figure 2.11 The Software Development Cycle stages [12]. 16
Figure 2.12 The Agile model development cycle. 18
Figure 2.13 The Scrum model development cycle [49]. 19
Figure 2.14 The Feature Driven Development cycle [42]. 20
Figure 2.15 The Extreme Programming Development cycle [48]. 22
Figure 2.16 Rapid Application Development (RAD) cycle [48]. . 24
Figure 3.1 EyeMap Architecture [47]. 30
Figure 3.2 Use Case Analysis of EyeMap [47]. 31
Figure 3.3 An example of XML data structure from EyeMap. [47] 33

Figure 3.4 An example of Nodes in memory [47]. 36
Figure 3.5 An example of the text HTML material. 37
Figure 3.6 An example showing various texts displayed in EyeMap [47]. 38
Figure 3.7 A zoomed fixation viewer from EyeMap [47] 39
Figure 4.1 prototype of EyeMap 2.0. 49
Figure 4.2 Example of Gitflow [16]. 58
Figure 4.3 Backlog issue creation screen. 59
Figure 5.1 EyeMap 2.0 System Design. 65
Figure 5.2 Full EyeMap 2.0 Database. 67
Figure 5.3 The upload Experiment process. 71
Figure 5.4 *text.html* for a Chinese language experiment. 72
Figure 5.5 Users Experiment list presented on the Home page. 74
Figure 5.6 The different tabs on the Visualise & Edit page. . . . 76
Figure 5.7 Display of Reading Variables and explanations. . . . 78
Figure 5.8 Options for reports to be exported. 78
Figure 5.9 Sample of word report exported to an excel file. . . 79
Figure 5.10 The different AOI's that can be generated. 80
Figure 6.1 List of Backlog tickets. 85
Figure 6.2 Sample Backlog *Bug* ticket being created. 85
Figure 6.3 Ticket opened by Mr Hynes in relation to the database design. 87
Figure 6.4 Misalignment of text ticket, with image, opened on Backlog 89
Figure 6.5 Individual results for the SUS for both survey groups. 97
Figure C.1 Django Core Architecture [9]. 116
Figure C.2 Python code to generate experiment table. 116
Figure C.3 UML class diagram of the experiment table. 117
Figure C.4 The base.html template for EyeMap 2.0. 119

Figure C.5 Example of how the base.html page is extended to other pages. 120
Figure C.6 Utilising Django urls. 121
Figure C.7 New experiment view from Django views. 122

LIST OF TABLES

Table 2.1 Fundamental aspects of PD as found by Halskov & Hansen [20] 27
Table 6.1 Requirements and fixes completion level for EyeMap 2.0 92
Table 6.2 Results of SUS Survey. 96

ABSTRACT

EyeMap was developed for visualising and analysing eye-tracking data. It was specifically designed to get the most information from eye movements data in relation to reading research. EyeMap was developed with more advanced features compared to other software systems for eye movements data including the ability to automatically generate AOIs, and carry out variable calculations for over 150 reading variables. However, given the ever-advancing landscape with software systems, the EyeMap system has developed several issues. It was developed in the flex programming language and with each new update to Adobe AIR, it is becoming outdated and difficult to maintain. Several bugs have arisen in EyeMap that is making it more and more difficult to use including not being able to correctly segment the Chinese language.

While talking to expert users of EyeMap that rely heavily on the tool, they presented ideas on how EyeMap could be improved by developing EyeMap 2.0. This project had the goal of developing develop EyeMap 2.0 using user participation so that current expert users of EyeMap would be able to provide input to the features in EyeMap 2.0 that would improve the features of the original EyeMap. The project used an Agile software development process, Rapid Application Development, to achieve this goal. This allowed the expert EyeMap users to participate heavily in the development and testing of the new system through the Backlog issue tracking software system. The Backlog system allowed the user to easily report any issues and follow the status of the issue ass it was being fixed and tested further.

This thesis discusses the entire process of using the Rapid Application Development model to develop EyeMap 2.0. It will present the tools,

frameworks and libraries that were used to develop this software artefact. The thesis will show that by using modern software development techniques and systems such as GitLab, EyeMap 2.0 can be continuously developed and improved upon, ensuring its future-proofing.

The first contribution of this thesis is to investigate agile software development models and determine which model has greatest potential for a small development team. This model would also have to allow the users of the new software to easily contribute to the development process to produce an easy to use and reliable software application. The second contribution will be to show a variety of tools that can can be used during development. The tools will benefit by maximising the contributions from the users to the project while also assisting with good software development practice for any development project. The final contribution will show how having the intended users of an application advise in the development of a project can only benefit the project. The users can clearly explain how a feature should behave and look. They can also aid with the testing of the created feature, ensuring the correct functionality before the feature is added to the application.

o

ACKNOWLEDGMENTS

I wish to thank my amazing fiance, Ana Susac, who has been my rock and support while I have completed my studies.

I would also like to thank my two supervisors Dr Aidan Mooney and Dr Susan Bergin. They were nothing but patient, understanding and encouraging throughout my time in doing this masters research.

I would also like to to thank Prof Ronan Reilly, Dr Christian Vorstius, Dr Ralph Radach, Xi Fan and Patrick Hynes who I worked with on this development project.

Lastly I would like to thank Keith Nolan and Mark Noone and the other Postgraduate students who I have worked with over the last few years.

DECLARATION

I confirm that this is my own work and the use of all material from other sources has been properly cited and fully acknowledged.

Emlyn Hegarty-Kelly

February 2020

Part I

INTRODUCTION AND BACKGROUND MATERIAL

1

INTRODUCTION & BACKGROUND

1.1 INTRODUCTION

This thesis presents the development process of EyeMap 2.0, a software tool that analyses eye-tracking data from reading experiments. It discusses the entire process from requirements gathering to the delivery of a functioning system. This chapter explores the motivation and the goals behind the project and introduces and discusses the research questions associated with the development of EyeMap 2.0. The chapter concludes by giving an overview of the thesis structure and provides an introduction to each stage of the development process.

1.2 MOTIVATION

The motivation for this project came while trying to see how experts and novices understand and read programming language code [22]. A large part of understanding programming language code is analysing how people read this code. After discussions with experts in eye-tracking and reading studies, EyeMap was presented by Professor Ronan Reilly, a leading expert on reading studies, as a solution to analyse eye-tracking data of how people read.

EyeMap is a data analysis tool that was developed to deal specifically with reading studies and the eye-tracking data that is generated from them. It allows the automatic segmentation of text and the calculation of 150 variables in relation to reading [47]. After some initial trials with EyeMap

and discussions with Prof Reilly, it was found that the software was not being maintained, it was difficult to access the source code, as well as having a number of other issues, described further in Chapter 3.

Taking into consideration the need for EyeMap to be redeveloped and the expert user group who were using EyeMap, including Prof Reilly, were eager to be involved in the development of a new version of the system. The decision was made to focus on the development of EyeMap 2.0 and enhance the features of the original EyeMap. The expert user group that relied heavily on EyeMap for their data analysis would provide valuable information and experience of EyeMap and how the software was used. The goal of this project was to develop EyeMap 2.0 and improve on the state of the art tool that was EyeMap.

1.3 GOALS

This project has one main objective, which was to redevelop EyeMap. However, there were a lot of considerations required to achieve that goal, including asking the following questions:

- Why does EyeMap need to be redeveloped?
- Is it possible to use the same framework that was used to develop the original EyeMap?
  - If not, what would be a better framework?
- Are there any extra requirements for the new version of EyeMap?

In order to answer these questions the author consulted with the current users of EyeMap and other researchers interested in using the software. These people are active in the research area of using eye-tracking for reading studies and have extensive experience with the original software. It would be beneficial to see how they interact with the current version of

EyeMap, consult with them as to how the new system should operate and discuss with them what the outputs of the new software should be. These researchers and users of EyeMap would be crucial to the success of the development of EyeMap 2.0.

With all of these questions and possibilities the goal of the project became a lot clearer. The project aim was to create EyeMap 2.0 by redeveloping the original EyeMap and using user involvement to improve on the original software. EyeMap 2.0 would be based on the work completed by the original developer of EyeMap, Dr Siliang Tang, with added functionality where possible, according to the needs and wants of the expert team of users.

1.4 RESEARCH QUESTIONS

Once the goals of the project were clear, a number of research questions were constructed. The first question relates to eye-tracking, and what it is and how experiments in eye-tracking are conducted. The second research question related to the tools and frameworks that could be used for this development project. Before any development could take place it was important to know and understand what EyeMap is and how the software operates. The last question related to the improvements that could be implemented to EyeMap.

The research questions are:

1. What is eye-tracking and why is it important?
2. What software models, tools, and languages are appropriate for a project of this nature?
3. What is EyeMap and how can it be improved?

1.5 CONTRIBUTIONS

The main contributions of this thesis are:

- To show an appropriate software development model for a project with a small development team in conjunction with a group of expert users that rely on the software to complete their research.
- To show a variety of tools that enable good development practices along with tools that maximise the contributions from the expert users to the software.
- Why having intended users of an application to consult during the development of software can only enhance any software application that is made.

1.6 THESIS OVERVIEW

This section will provide a brief overview of what will be discussed in each of the thesis chapters. Chapter 2 investigates eye-tracking and the processes associated with it when carrying out experiments. It describes the motivation behind the development of the original EyeMap. The chapter reviews the software development process and explores which development model would be best suited for the development of EyeMap 2.0. The chapter then discusses the philosophy behind user participation, so as this would be used to gain as much input from the expert group on the project as possible.

Chapter 3 provides an in depth analysis of the original EyeMap system. It explores the architecture and key features of this software and continues to describe why the decision was taken to develop EyeMap 2.0 rather than to try and extend EyeMap.

Chapter 4 introduces the users involved in the project. It describes their experience levels working with EyeMap. It will also describe the process of constructing the software requirements for EyeMap 2.0, from consultations with the users. This chapter also provides the Software Requirements Specification for EyeMap 2.0. It outlines the programming languages and frameworks to be used in developing EyeMap 2.0 and the reasons behind each of the final decisions made for the software development process.

Chapter 5 describes the redevelopment process used in the development of EyeMap 2.0. It outlines the task list that was created to develop EyeMap 2.0 and shows the workflow to complete this list and implement the features require for EyeMap 2.0.

Chapter 6 presents the evaluation of the EyeMap 2.0 system. It shows how the Backlog system was used to evaluate and test EyeMap 2.0. The chapter then presents a requirements checklist to highlight what features were completed on the RAD task list. The chapter finishes by presenting the results of a usability survey conducted on EyeMap 2.0 by the end user team and by individuals with no prior knowledge of eye-tracking or EyeMap 2.0.

Chapter 7 will discuss the conclusions for the EyeMap 2.0 system. It will show the results from the research questions and lessons learned during the process and discuss how the process can be improved for further development. The chapter will go on to discuss future work for EyeMap 2.0 and what can be done to contribute more to both EyeMap 2.0 and the types of reading studies that can be completed.

2

RELATED WORK

2.1 INTRODUCTION

As outlined in Chapter 1 this project is focusing on the redevelopment of EyeMap. EyeMap is a piece of software that was written to analyse eye-tracking data for reading research. To successfully redevelop EyeMap it is important to firstly understand "What is eye-tracking and why it is important?" This chapter will show the origins of eye-tracking technology, what eye-tracking is and why it is important. The chapter will then begin to answer "What software models, tools, and languages are appropriate for a project of this nature?" by investigating the software development cycle and the different processes that are involved in each stage. EyeMap has a core base of expert users that are participating in the development process. This chapter will look at what is User Participation to gain a greater understanding of how to utilising the knowledge that these expert users have and ultimately help to answer "What is EyeMap and how can it be improved?". These three areas combined will ensure EyeMap 2.0 can be developed to a high standard.

This chapter will begin by discussing eye-tracking and what it is. It will proceed to look at the software development cycle and investigate different software development models that can be used in the redevelopment of EyeMap 2.0. The chapter will conclude by investigating User Participation and show how the users of the original EyeMap can be involved in the development of EyeMap 2.0.

2.2 EYE-TRACKING

2.2.1 WHAT IS EYE-TRACKING?

Eye-tracking is the measurement of eye movements and the position of the eye relative to a persons head. The device used to measure this movement is an eye-tracker. One of the earliest eye-tracking devices was built in 1908 by Edmund Huey. The device was built to observe the gaze direction by attaching pointers to contact lenses for reading studies [28]. It was not until 1937 that the first eye movements were recorded by Guy Thomas Buswell in the University of Chicago. He achieved this by using light reflected from the eye and recording this on film [6]. Modern eye-tracking uses the same underlying principle. A camera is focused on the eye and then infra-red light is shone into the eye. Using the angle between the centre of the pupil and the reflection from the infra-red light the underlying algorithms can track what position the subject is looking at on the screen [52]. Figure 2.1 shows an example of the camera view on an eye as infra-red light is shone onto it. The orange cross is the centre of the pupil and the white cross is the corneal reflection [52].

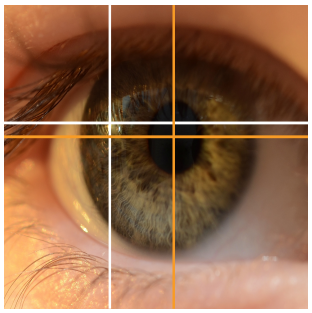


Figure 2.1: An example of the eye-tracking camera view on an eye [52].

Eye-trackers are used for research in several fields, including psychology, psycho-linguistics, marketing, human-computer interaction, and product design. Due to the widespread use of eye-trackers, there are many compa-

nies that produce eye-tracking hardware and analysis software. Arguably, the top three companies in the field are Tobii, SMI and EyeLink [51]. Each of these companies work with both stationary and mobile eye-trackers. A stationary eye-tracker is set in a fixed location where participants in experiments are not required to move (see Figure 2.2, where mobile eye-trackers are designed for use in experiments where the participant may be required to move freely (see Figure 2.4) for example when conducting an experiment that involves participants to be driving. Examples of stationary eye-trackers include the EyeLink 1000 from SR Research [44] (Figure 2.2) or the SMI RED-m [41] (Figure 2.3) and an example of a mobile eye-tracker is the Tobii pro glasses [50] (Figure 2.4).

Figure 2.2: The EyeLink 1000 being used for an experiment with chess.

Figure 2.3: The RED-m from SMI on a laptop computer.

Figure 2.4: Tobii pro glasses used in a driving experiment.

Each of these companies uses their own proprietary analysing software to analyse the data from their eye-trackers. They produce heat-maps, metrics based on AOI's and data export and segmentation. Heat-maps are a graphical representation of fixation data of where most fixations were on an image, see Figure 2.5 as an example. Although each company produce extremely successful software packages. They are not suited for calculating the variables required for reading studies. These reading variables will be discussed more in Section 3.4.5.

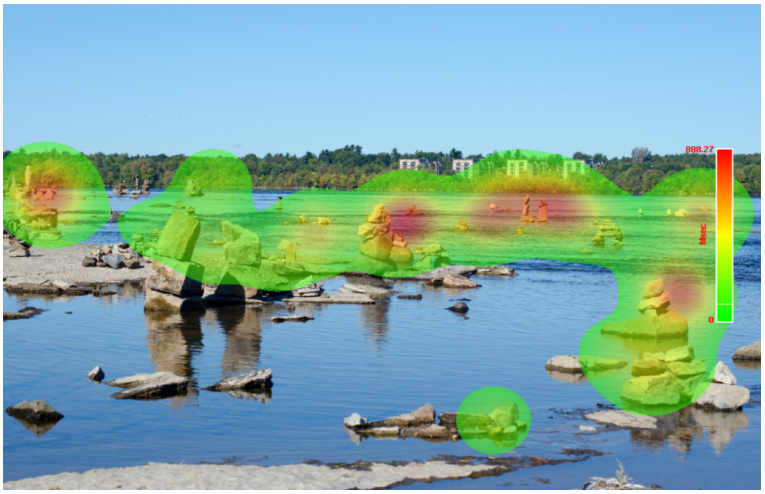


Figure 2.5: Heat-Map from SR Research of fixations on an image.

2.2.2 EYE-TRACKING & READING

The first recorded evidence of eye-tracking in reading is credited to Louis Emile Javal for his work observing eye movements in 1879 by using mirrors to watch the eye while reading. It was noted by Javal that the movement of the eye was not smooth during reading but saccadic in nature [26]. A saccadic movement of the eye is a simultaneous movement or quick jerk of both eyes between fixations in the same direction [7]. That same year Ewald Hering confirmed this saccadic movement of the eyes in reading by listening to the eye, by placing a rubber tube on the eye and then listening for a clapping sound and comparing the occurrences of the clapping sound to afterimages, an image that continues to appear in the eyes after a period of exposure to the original image, that were taken while reading [23]. A fixation is a point generally between two saccades in which the eyes are relatively stationary. Figure 2.6 shows fixations and saccades.

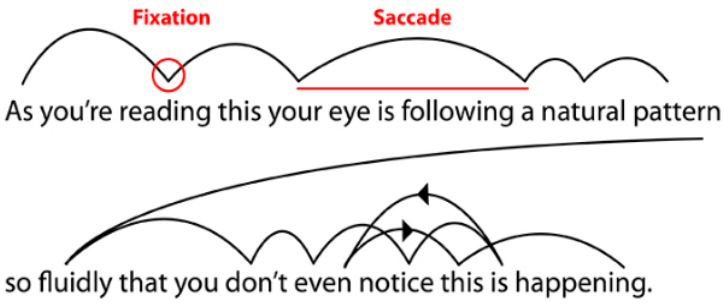


Figure 2.6: Fixations and saccades during reading. Note: saccades are drawn like this to show the hop or jerk that is involved with a saccade between fixation points

Fixations and saccades are important measures when conducting reading studies. The mean fixation duration is said to be between 200 - 300 ms depending on if it is silent or oral reading being completed, and the mean saccade size is 1.5°(about 6 letters) to 2°(about 8 letters), again depending

on silent or oral reading [38]. The visual resolution limits or acuity of vision is an important constraint when reading. The highest acuity occurs at the fovea. Figure 2.7 provides an example of this where the centre of the fixation is on the word fox, the fovea area is from the centre of the fixation to 1°in any direction. The parafovea area is 1°- 5°away from the fixation point, and the periphery area is more than 5°from the fixation centre [2]. A saccade moves a fixation, or the fovea, to the word that is being processed next [39]. Irwin [25] showed that the cognitive process continues during a saccadic movement, while it was also shown by Martin [31] that no new visual information is obtained.

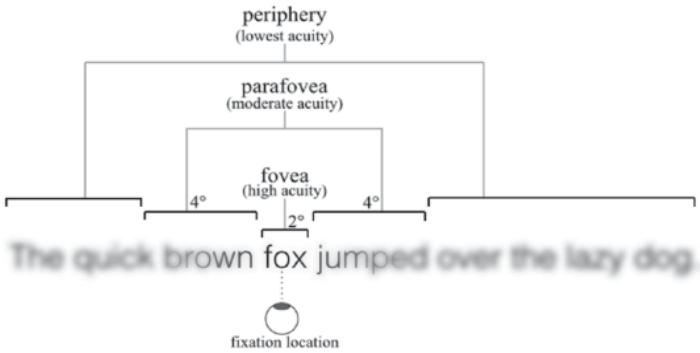


Figure 2.7: Diagram showing the viewing angle of the fovea, parafovea and periphery

2.3 FONTS & FONT METRICS

Fonts are an important part of what makes EyeMap work for AOI generation, so it is important to understand different aspects of fonts & font metrics. There are two different types of font widths, namely proportional fonts and monospace fonts. A proportional font is a font where different characters have different widths, for example, the width of the character "i" is narrower in width to the character "m". An example of this can be seen on the bottom line of Figure 2.8. A monospace font has uniform character

widths for all characters in the font. An example is presented on the top line of Figure 2.8 where the characters "i" and "m" have the same width.

Figure 2.8: Proportional font v monospace font [45].

Figure 2.9 provides a detailed look at font metrics. The baseline is the line that the text is located on. The advancement shown on the character "j" is another term for the character width. For EyeMap the combined character widths of a word would create the AOI width for that word. For the AOI height, there are three measurements that need to be taken into account. These are the ascent, descent and line gap or leading. These three measurements are all presented in Figure 2.9. The ascent is from the base line up to the top of the tallest character. The descent is from the base line down to the lowest character and the leading is the from the bottom of the descent to the top of the ascent on the line of text under the current line.

2.4 SOFTWARE DEVELOPMENT

In 1968 the NATO Science Committee organised a conference on Software Engineering and this conference recognised the many problems with software development including design, production and service [32]. The term "Software Engineering" became more popular and widely used after this conference and the report they produced formed the foundation of the in-

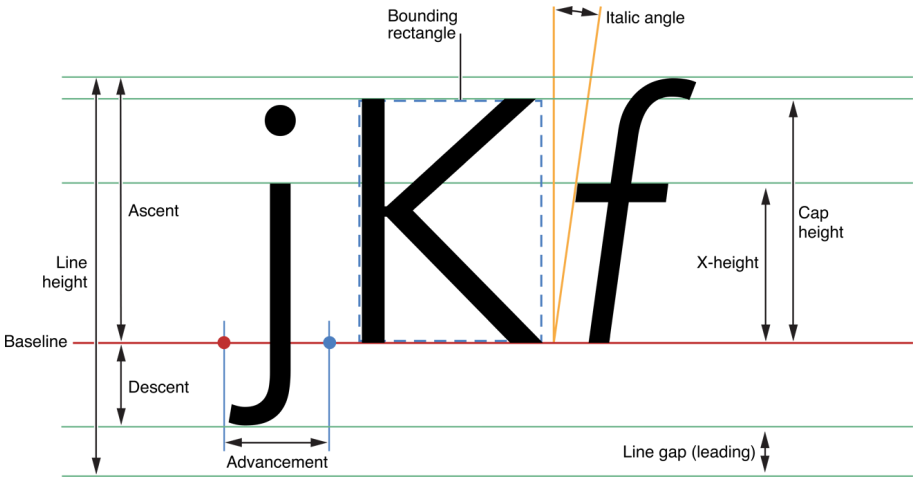


Figure 2.9: Font Metrics [15].

dustry. Figure 2.10 shows the software development process as discussed by Selig [32] at the NATO conference while Figure 2.11 shows the modern interpretation of the same process. These two figures represent the different stages that every development team goes through while developing any software project.

2.4.1 STAGES OF THE SOFTWARE DEVELOPMENT CYCLE

Apart from the addition of a planning stage in Figure 2.11, it is apparent from both Figures 2.10 & 2.11 that the stages of the software development cycle have not changed much over the last 50 years. These core principals form the basis of the software development models that are widely used today and shall be discussed further in Section 2.4.2. The stages of the Software Development Cycle are: Planning, Analysis, Design, Implementation, Testing & Integration, and Maintenance. These stages can be further described as follows:

1. **Planning** - The planning phase of a project is where the requirements gathering takes place. This involves discussions with stakeholders, customers and sales teams about market research, cost analysis and

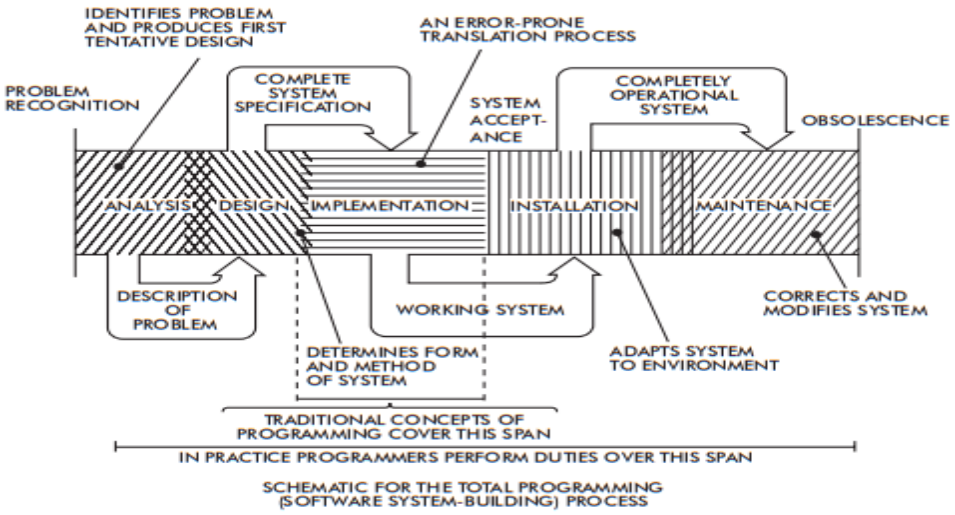


Figure 2.10: The Software Development Process as shown and discussed in the report from the NATO Software Engineering Conference 1968 [32].

feasibility studies to see if the software can be created within budget while deciding on the required functionality.

2. **Analysis** - During the analysis phase the development team will create a software requirement specification document (SRS) for the project (see Section 4.4). This document is created from the results of the planning stage to clearly identify the scope of the project and remove any inconsistencies that may have arisen. The SRS is meant as a guide for the developers to follow so that they may have minimal interruptions, delays and add-ons in the implementation phase.

3. **Design** - In the design phase the development team study the SRS to decide on the new system architecture and plan the correct tools and methods to use to build the software. They will also look at the screen layout, functionality and other business documents of the software. With this information, they will also generate design docu-

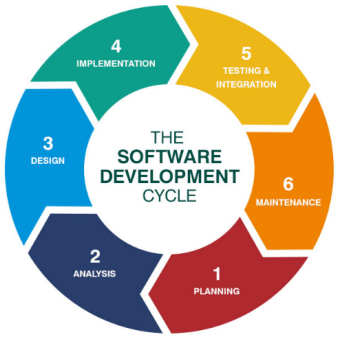


Figure 2.11: The Software Development Cycle stages [12].

ments to go with the SRS to further reduce any delays in the implementation phase.

4. **Implementation** - In the implementation phase, the software development team will build the software based on the design documents and SRS using the system architecture that was chosen. The number of software tools that the team can use for development at this point is vast, with each team having their own preferences depending on the framework and the type of project. Along with their chosen tools they use a software development model, which shall be discussed in more detail in Section 2.4.2, to keep the team focused and on track during this stage.

5. **Testing & Integration** - This phase checks if the software meets quality standards. The development team regularly check the system in development to ensure that:

   - The requirements are met for the project.
   - There are no errors or bugs in the project that need to be fixed.
   - The software is of a high quality to meet the stakeholders' expectations.

6. **Maintenance** - Once the software has been released to the customer there may be further issues that arise at a later stage. These issues

need to be fixed so that the software remains at a high standard and continues to meet customer expectations.

2.4.2 SOFTWARE DEVELOPMENT MODELS

Since the NATO Science Committee Conference on Software Engineering, a large number of software development models have been proposed. There are several software developmental models such as incremental models, v-models, iterative models and Agile models as Agile models are known to incorporate user participation. And thus the development of EyeMap 2.0 an Agile software development model was chosen.

The manifesto for Agile software development was created by a group of 17 people from software development backgrounds in 2001 [4]. It focuses on iterative and incremental development. The requirements and solutions evolve through collaboration between teams that consist of developers and users. As can be seen in Figure 2.12 this model works with development cycles, whereby a feature is developed and given to the users for approval. If the feature is approved it is deployed to the software and if the feature is not approved the requested changes or fixes and required adjustments that are recorded and it is added back to the start of the development cycle where they are then processed by the developers again. The agile model is an adaptive model that incorporates PD and can adapt quickly to requested changes [24]. There are numerous different forms of Agile methods but the following four models each have their own advantages and disadvantages and each could be potentially used for the redevelopment of EyeMap 2.0. The four models are:

1. Scrum Development
2. Feature Driven Development
3. Extreme Programming

4. Rapid Application Development

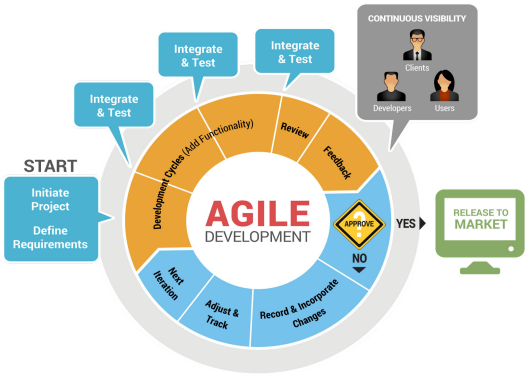


Figure 2.12: The Agile model development cycle.

1. *Scrum Development*

The Scrum model development cycle is presented in Figure 2.13. In this process, the team appoints a *scrum master* who controls each sprint, or development cycle that the teams carries out and ensures the vision of the software remains the same throughout. The appointed *scrum master* prepares the architecture and Sprint Backlog. Each sprint decided on by the *scrum master* can change quickly based on feedback from users. As a direct result of the nature of the sprints, the process is much better suited to small teams. These teams should contain experienced members that can adapt and change quickly to the ever-changing sprints, and with each sprint cycle, the team incrementally delivers features of the software and a valuable product [40]. As outlined by TatvaSoft [48], the advantages and disadvantages of scrum development are described next.

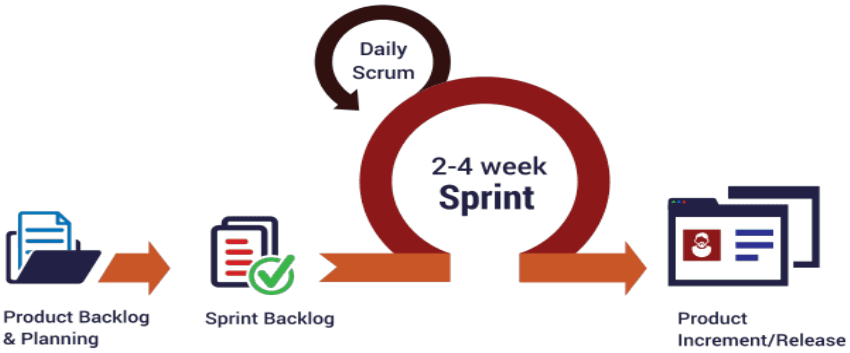


Figure 2.13: The Scrum model development cycle [49].

The main advantages of the scrum development model are:

- Due to the free forming of Scrum sprints it is good for projects where requirements documentation is not vital for the success of the project.
- Development steps are clearly visible as a result of frequent updating of the project progress.
- Daily meetings help measure individuals contributions to the project, thus increasing the overall productivity of the team.

The main disadvantages of the scrum development model are:

- Development model suffers if time or cost estimation is inaccurate.
- It is not good for large development teams.
- Experienced team members are required for this process to work well, inexperienced or novice members can slow down the development.

2. *Feature Driven Development*

Figure 2.14 shows the development steps of feature driven development. The first step is for the development team to develop a model of the whole system. Next the team breaks the system into smaller, more manageable,

features that are to be implemented. The third step is for the development team to create a plan for development based on the feature list. The next two steps are then handled by smaller dynamically formed feature teams, with each small team being responsible for designing and ultimately developing or building the feature. Once the feature has been developed it is presented to the end user for approval before the feature team moves onto the next feature. At each stage, the feature teams are updating previous features as they progress through the feature plan [14]. The main advantages and disadvantages of feature driven development [48] are described next.

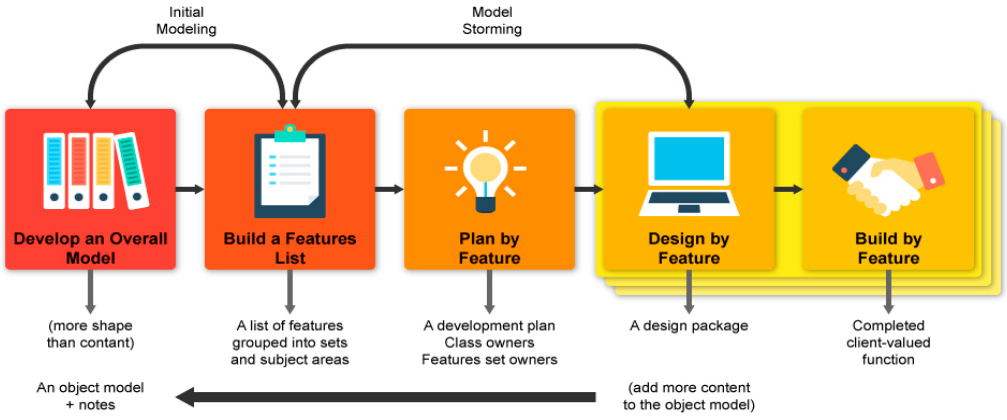


Figure 2.14: The Feature Driven Development cycle [42].

The main advantages of the feature driven development model are:

- As a result of the development team structure, larger projects develop quicker and with more success.
- Following the steps of the process allows the project to be completed and in a short and timely manner.
- The process is built on standards for the development industry, so it aids with development following best practices.

The main disadvantages of the feature driven development model are:

- As a result of the smaller feature teams feature driven development is not suited for smaller projects, and as such would not suit a lone developer.
- The lead developer must act in multiple roles, meaning the rest of the team has a high dependency on this person.
- No documentation is provided to clients, meaning that it is hard for them to know exactly what features from the system is being provided to them for testing.

3. *Extreme Programming*

Extreme Programming is a software process that relies heavily on user involvement. It allows the development teams to build software around vague or constantly changing requirements. It takes simple to the extreme, by breaking down tasks to their most simple form and using this as a starting point, which allows rapid feature development through unit testing and customer input. The method allows developers to deal with problems proactively [35]. The five stages of Extreme Programming, as presented in Figure 2.15, are:

1. Requirements
2. Stories
3. Test Cases
4. Tasks
5. Completion

The requirements stage gathers the requirements for the system from the end users. Once the requirements have been gathered the development

team builds the user stories for the system. The stories contain different user experiences and use case scenarios for the system. The test cases are generated to ensure each use case does what it is supposed to do and ensures the system does what it is required to do. The tasks stage of the model is the development stage of the model. Once the test cases are written in the test case stage, the team can then develop each feature according to the test cases to ensure the correct functionality is completed by the system. On completion of the stage of the task, the user carries out final testing before the system is approved for completion. The main advantages and disadvantages of extreme programming [48] are described next.

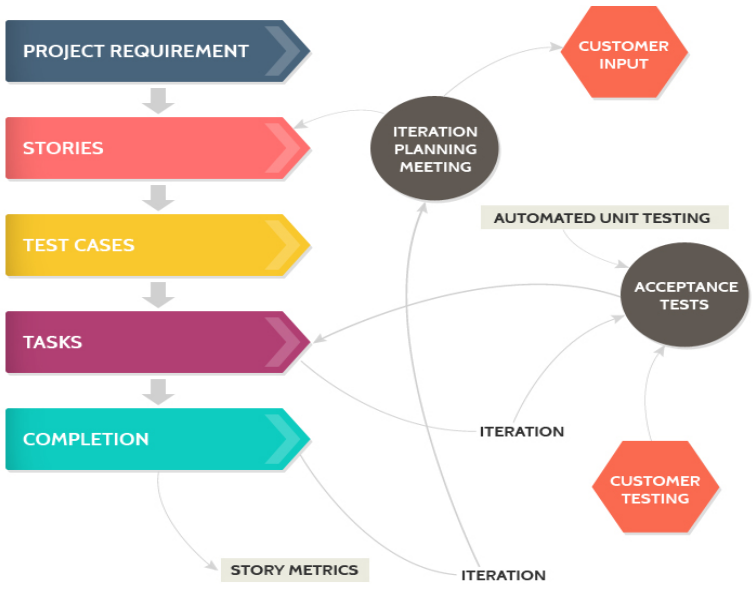


Figure 2.15: The Extreme Programming Development cycle [48].

The main advantages of the extreme programming development model are:

- There is a large emphasis on user involvement.
- The process commits developers to their schedule by creating plans and schedules.

- As it develops with modern development methods, quality software is produced.

The main disadvantages of the extreme programming development model are:

- The process is only as effective as the team members leaving little room for inexperience or novices.
- Meetings are required frequently meaning there is a high cost on clients.
- With a large number of items that may require development changes it can be difficult for the development team to manage the changes.

4. Rapid Application Development

Rapid Application Development (RAD) was proposed by James Martin in his 1991 book "Rapid Application Development" [30]. It takes a user-centred approach to software development. This software process quickly produces prototypes of the system to be developed. This is possible as users are integral members of the team and are deeply involved with every aspect of developing the system from requirements to testing [29]. Figure 2.16 shows the RAD cycle starting from the requirement stage that is planned out using PD. The next stage is prototyping that cycles through the design and construction of features. The user also participates in the design of the feature so that once a feature is completed they are given to the end user for testing. Finally, when the end user has approved the feature it is integrated into the final system. The main advantages and disadvantages of RAD [48] are described next.

The main advantages of RAD are:

- It reduces the risk and efforts on the developer as a result of working with users.

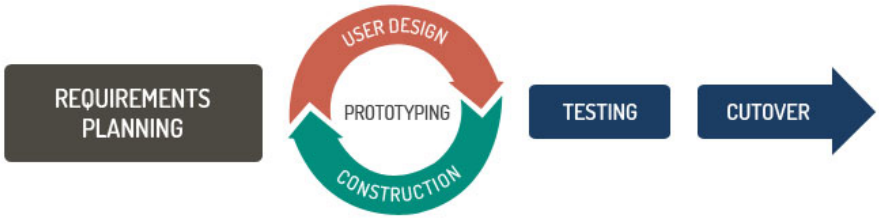


Figure 2.16: Rapid Application Development (RAD) cycle [48].

- Users and clients review the system quickly which can catch any potential difficulties early on.
- User feedback and involvement is highly encouraged which improves the overall software.

The main disadvantages of RAD are:

- There is a strong dependency on the team and individual to identify and carry out the development of the system.
- As a result of the prototyping that involves the user, only systems that can be modularised can be developed using this process.
- Highly skilled and knowledgeable developers and teams are required to use this process.

2.4.3 SOFTWARE DEVELOPMENT PROCESS USED

After reviewing and dissecting the four aforementioned agile methods it was decided that the process that would best suit the redevelopment of EyeMap 2.0 was RAD. The reasons for this decision can be summarised as follows:

1. Scrum development would work but experienced developers are required to carry out the process effectively, and the experience levels of some of the team are unknown.

2. Feature driven development is for large development projects and does not suit a lone developer as is the case with the redevelopment of EyeMap 2.0.

3. As with Scrum development experienced team members are required with extreme programming for maximum efficiency, also regular meetings with users or clients. As a result of the distance between the developer and users regular meetings will not always be possible for the redevelopment of EyeMap 2.0.

4. RAD involves the users in every aspect of the development process from initial designs to testing. The heavy user involvement required for RAD also means that any potential errors in the software are caught early on. The original EyeMap can also be modularised as will be seen in Chapter 3.

With RAD selected as the development model, the next section will show what Participatory Design is and how it can improve the RAD cycle.

2.5 USER PARTICIPATION

The users that were involved in the redevelopment of EyeMap have vast expertise in the fields of eye-tracking, reading research and utilise EyeMap regularly. To attempt to answer the research question "What is EyeMap and how can it be improved?" engaging with these users is vital as they are most suited to helping to answer this question and ensure that the development of EyeMap 2.0 is completed correctly and to specification. To assist the development process user participation will be used. user participation has its roots in participatory design (PD) and understanding this philosophy will demonstrate how the strengths of the users were used in this approach.

"Participatory design is an approach to engineering technological systems that seeks to improve them by including future users in the design process. It is motivated primarily by an interest in empowering users, but also by a concern to build systems better suited to user needs" [34]. It is a movement that started from workers unions in Scandinavia in the 1970s and 80's [43]. The movement aimed to empower workers and bring democracy to the workplace. This was achieved by allowing workers to partake in the development of new technologies and practices in the workplace by working with the developers. The movement gave the workers more rights and allowed them to take more control of their workplace [43]. In 1990 the Computer Professionals for Social Responsibility founded the Participatory Design Conference, which has run bi-annually since [18]. The conference presents research from people in design, development and implementation of information and communication technologies and services [33].

Kim Halskov and Nicolai Brodersen Hansen [20] conducted a review based on ten years of research papers presented at the conference between 2002 and 2012. In the review, they identified three general definitions of "User Participation". These are:

1. Implicit - Users are part of the design/development process.
2. Users' point of view - the users' point of view takes into consideration what the users deem important and suggests they make the best decisions based on their knowledge.
3. Mutual learning - This is the transfer of knowledge between users and designers/developers.

Halskov and Harisen also identified and discussed five fundamental aspects of PD which can be seen, and defined, in Table 2.1.

When people spend every day using a system, they become experts in that system and know what its strengths and failings are. If there is a

Table 2.1: Fundamental aspects of PD as found by Halskov & Hansen [20]

Politics	People who are affected by a decision should have an opportunity to influence it
People	People play critical roles in design by being experts in their own lives
. Context	The use situation is the fundamental starting point for the design process
Methods	Methods are means for users to gain influence in design processes
Product	The goal of participation is to design alternatives, improving quality of life

change being made to the system the users are often in the best place to make the decisions about any changes. The changes they effect would be more substantial and the finished product would improve the workflow quality for the users. This philosophy greatly benefited the development of EyeMap 2.0 and will be evident throughout the RAD process.

2.6 SUMMARY

The chapter answered the question "What is eye tracking and why is it important?" by looking at eye-tracking and showed how it began with Huey's original design for an eye-tracker in 1937 [28] and progressed to the numerous modern companies that exist today including SR Research SMI Vision and Tobii. An important use for eye-tracking is in reading research and the different measures for eye movements were explored and demonstrated to show EyeMap is a valuable piece of software.

The chapter also began to answer two of the other research questions, "What software models, tools, and languages are appropriate for a project of this nature?" and "What is EyeMap and how can it be improved?". To redevelop EyeMap it was shown that the RAD process would best suit the size of the development team, the distance between members of the development team and the philosophy behind PD would utilise the expert's knowledge to improve EyeMap in the development of EyeMap 2.0. Chapter 4 will further discuss these two questions and provide further clarity.

3

EYEMAP

3.1 INTRODUCTION

This chapter begins to answer the research question "What is EyeMap and how can it be improved?". It provides a more in-depth look at the architecture and key features of EyeMap. The chapter presents reviews the workflow within EyeMap, from how users prepare data to displaying the data on the screen and then exporting the reading variables for further analysis. The chapter will also describe why the decision was taken to develop EyeMap 2.0.

3.2 WHAT IS EYEMAP?

EyeMap was a first of its kind software developed for visualising and analysing eye-tracking data [47]. It was first registered on SourceForge as open source software in May 2010 by Dr Siliang Tang [13]. SourceForge is a web-based service that offers a centralised online location to control and manage open-source software projects. It was designed as a platform-independent software tool that operates independently of the major companies that produce eye-tracking hardware and analysis software. EyeMap was designed specifically to get more information about how people read from eye movement data for reading research. EyeMap provided software with more advanced features for reading research beyond other software systems for eye movements data like the Tobii Pro Studio. The features that were built into EyeMap include text stimulus presentation, area of in-

terest extraction, eye movement data visualisation, and experimental variable calculation while supporting the analysis of binocular eye data and proportional and non-proportional fonts. Given that different languages use a wide variety of different textual representations, the allowance for proportional and mono-spaced fonts means that EyeMap is well suited for reading research in a wide number of languages.

3.3 EYEMAP - ARCHITECTURE

EyeMap was developed using Adobe AIR. Adobe AIR is a framework that allows developers to build applications with HTML, JavaScript, and Action Script; it also allows developed systems to be deployed across different operating systems easily. Figure 3.1 presents an overview of the architecture of EyeMap. It can be seen that there are two main components within the architecture: the "Function Module" in blue and the "Data Files" in yellow. Each component consists of smaller modules. The Key modules in these components will be discussed in Section 3.4. The use case analysis of EyeMap, that is shown in Figure 3.2 shows the complete use case scenario of initially preparing and loading data to the exportation of the processed data.

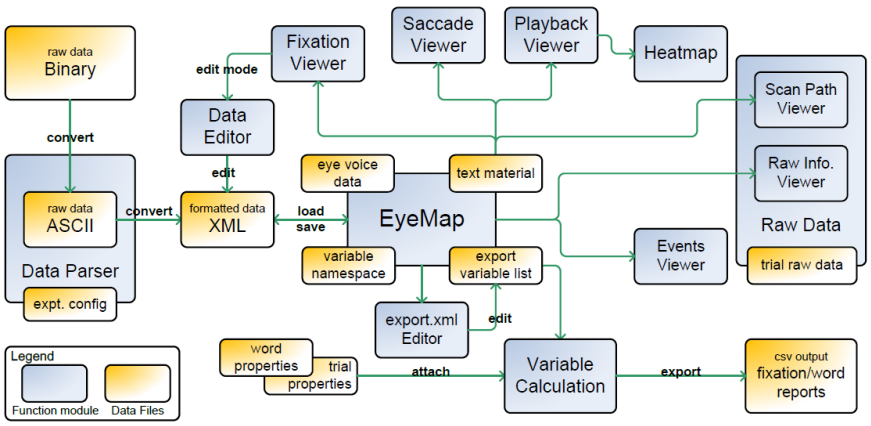


Figure 3.1: EyeMap Architecture [47].

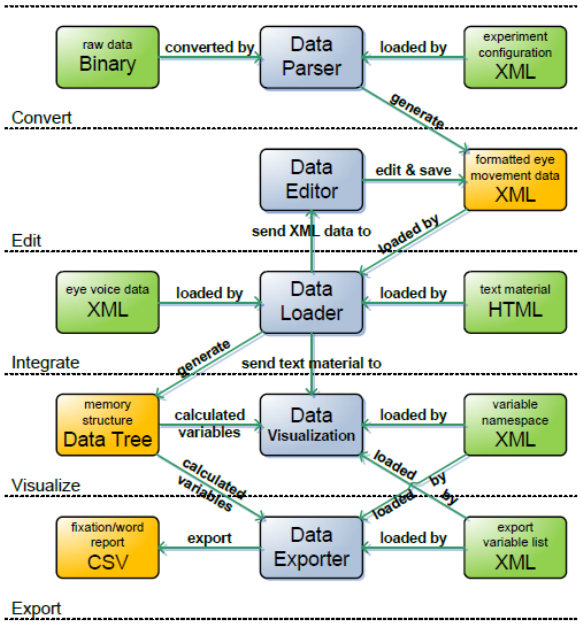


Figure 3.2: Use Case Analysis of EyeMap [47].

Analysing Figure 3.2 it can be seen that the raw binary data contains the experimental data, this is the data that is outputted by the eye-tracker hardware in a format that can be interpreted by EyeMap. Combined with the raw binary data and the configuration for the experiment in XML format the Data Parser (see more in Section 3.4.1) generates the formatted eye movement data in XML format. This XML file is then loaded by the Data Loader (see more in Section 3.4.2), which is also responsible for loading the optional eye voice XML data, this is eye-tracking data that also has connected voice recordings, and the text HTML material. The Data Loader can then either send the loaded XML data to the Data Editor (see more in Section 3.4.4) to be edited or to the Data Visualisation (see more in Section 3.4.3) for viewing. When the Data Loader passes the data to the Data Visualisation it also generates a memory structure for the calculation of Reading Variables (see more in Section 3.4.5). The calculated variables are also available to the Data Visualisation module so that the user can see the variables as they work with the data. Finally, the data and calculated variables are available for selection in the Data Exporter (see more in Sec-

tion 3.4.6) where they can be exported in either the word or fixation report in CSV format for further analysis.

3.4 EYEMAP - KEY FEATURES

As discussed in Section 3.3 there are several important modules contained within EyeMap. These are:

1. Data Parser
2. Data Loader
3. Data Editor
4. Data Visualisation
5. Reading Variables
6. Data Exporter

Each of these 6 Key modules shall now be discussed in more detail.

3.4.1 DATA PARSER

Eye-tracker manufacturers output the eye-tracking data in their own way, meaning the lack of standardisation could be a problem when data is shared between researchers for analysis purposes. Most eye-tracking hardware tools provide application programming interfaces (API) to convert the data into an ASCII file. This is ideal for data sharing but not ideal for data processing, due to the processing of the ASCII files. Based on the work of Halverson and Hornof [21] and the EU funded project COGAIN [3] EyeMap developed a novel device-independent data format specifically for reading experiments. This XML format takes into account the related

events that occur during a reading experiment. An example is represented in Figure 3.3.

```
<root>
    <trial id="0">
        <fix>
            <eye>R</eye>
            <st>927307</st>
            <et>927470</et>
            <dur>164</dur>
            <x>100.1</x>
            <y>104.7</y>
            <pupil>1501</pupil>
            <raw sx="-2.7" sy="0.9" ex="2.4" ey="-0.4"/>
            <id>0</id>
        </fix>
        <sacc>
            <eye>R</eye>
            <st>927471</st>
            <et>927515</et>
            <dur>45</dur>
            <x>102.6</x>
            <y>104.1</y>
            <tx>508.6</tx>
            <ty>133.3</ty>
            <ampl>11.66</ampl>
            <pv>506</pv>
             <id>0</id>
         </sacc>
    </trail>
</root>
```

Figure 3.3: An example of XML data structure from EyeMap. [47]

It can be seen that there is one root node for each experiment and the trial nodes for the experiment are contained within that. A trial node contains all the information for a single trial in an experiment. In this example the trial has an id of 0. There are two main types of data contained within the trial that are represented by their own nodes, namely fixations and saccades, and each one respectively contains all the data for that fixation or saccade. As can be seen in Figure 3.3 each fixation and saccade has their own attributes. A break down of the data contained in the fixation and saccade nodes is:

Fixation

```
<fix>
    <eye> - This represents the eye the fixation is associated
        with - L for Left and R for Right.
    <st> - This represents the start time of the fixation in
        milliseconds.
    <et> - This represents the end time of the fixation in
        milliseconds.
    <dur> - This represents the duration of the fixation in
        milliseconds.
    <x> - This represents the average x coordinate of the
        fixation.
    <y> - This represents the average y coordinate of the
        fixation.
    <pupil> - This represents the average pupil diameter for the
        current fixation.
    <raw> - This represents the shifted pixels of the start (sx/
        sy) and end (ex/ey) x, y coordinates to the average x, y
         coordinates of the current fixation.
    <id> - This represents the unique id of the current fixation.
</fix>
```

Saccade

```
<sacc>
    <eye> - This represents the eye the saccade is associated
        with, L for Left and R for Right.
    <st> - This represents the start time of the saccade in
        milliseconds.
    <et> - This represents the end time of the saccade in
        milliseconds.
```

```
    <dur> - This represents the duration of the saccade in
       milliseconds.
    <x> - This represents the x coordinate of the starting
       position of the saccade.
    <y> - This represents the y coordinate of the starting
       position of the saccade.
    <tx> - This represents the x coordinate of the end position
       of the saccade.
    <ty> - This represents the y coordinate of the end position
       of the saccade.
    <ampl> - This represents the visual angle traversed by the
       saccade.
    <pv> - This represents the peak values of gaze velocity in
       visual degrees per second.
    <id> - This represents the unique id of the current saccade.
</sacc>
```

Each fixation or saccade has their own unique id represented by an *<id>* tag where even numbers are used for the right eye and odd for the left eye. As a result of this coding scheme to represent which eye a particular fixation or saccade is associated with, the Data Editor and Data Visualisation can integrate and synchronise for binocular eye data easier.

Once that data is prepared in this XML format it can then be passed to the Data Loader for editing and visualisations.

3.4.2 DATA LOADER

The Data Loader prepares the formatted eye movement XML data from the Data Parser as a tree structure containing six levels from the root to the leaves. The levels are the experiment level, the trial level, the word level, the gaze level, the fixation level and the saccade level. This tree is

an abstract node class that is made up of a set which contains an id, a reference to its parent node, a reference to its predecessor and successor and a set of attributes and a tuple that records all child nodes. These Nodes are also used for the calculation of the reading variables that will be exported later in the system.

A visual example of a snapshot of these nodes in memory can be seen in Figure 3.4. It can be seen that the trial level contains word child nodes. Focusing further on the text in the centre of the image on word 5 ("stinking") and its corresponding word node it contains two gaze nodes. The first of these gaze nodes contains two fixation nodes and they, in turn, have a saccade node each.

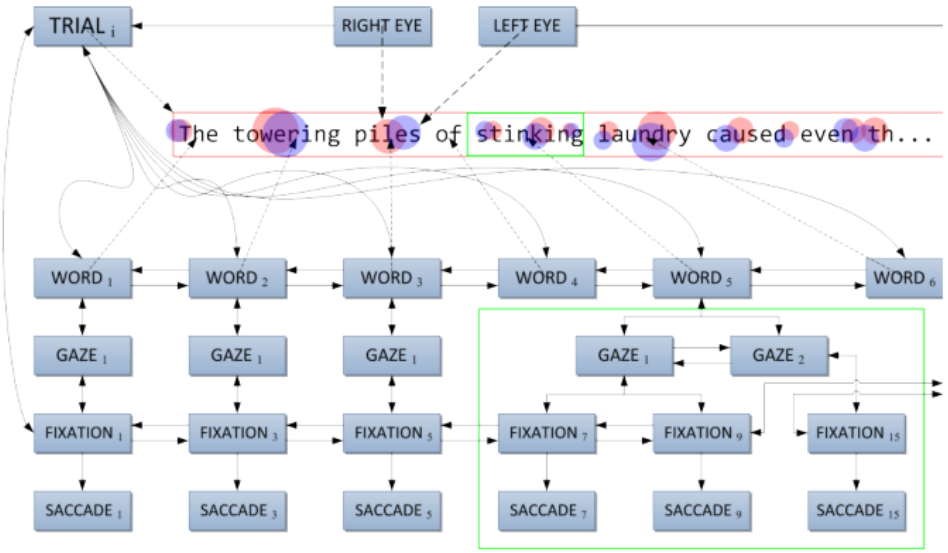


Figure 3.4: An example of Nodes in memory [47].

The data in the tree is updated each time a new trial is selected or an edit to an existing trial is made. This allows for quicker loading times compared to loading all data for all trials at once even though a certain trial is required, which in turn ensures there are not large amounts of data stored in memory at any one stage.

The Data Loader also loads the text stimulus for the experimental trials in HTML format. This is important as the HTML file contains all details

about the experiment, this is shown in Figure 3.5. This is a standard HTML structure where the <*head*> contains the experiment details and the <*body*> contains the text used in each of the trials of the experiment. The <*head*> tag uses 2 methods for the experiment details. The first is inline CSS through the <*style*> tag, which can be seen on lines 3-11 of Figure 3.5. This contains information about text formatting in the experiment, the font that used, the font size and the leading was between lines. The second method is to use custom HTML tags that can be seen on lines 12 & 13 of Figure 3.5. This information provides the starting x & y coordinates of the top left most character of a trial, for all trials of the experiment. The body then contains the individuals' trials for the experiment, with each <*p*> tag containing the text for each trial. The number of trials is represented by the number of <*p*> tags. The example presented in Figure 3.5 contains two trials and these can be seen in lines 17-20 & 21-24. The text contained between each <*p*> tag represents the text presented to the participant of the experiment, it also maintains the same number of lines that were used in the experiment.

Once the Data Loader has loaded the tree and the HTML file it can be passed to the Data Visualisations and Data Editor.

```
1  <html>
2      <head>
3          <style type="text/css">
4              body
5              {
6                  color:black;
7                  font-family:Times New Roman;
8                  font-size:30;
9                  leading:20;
10             }
11         </style>
12         <x>120</x>
13         <y>100</y>
14     </head>
15
16     <body>
17         <p>
18             Since candles were "expensive, the glowing fire in the
19             fireplace was often the only source of light.
20         </p>
21         <p>
22             Tortoises from the Galapagos Islands have tailormade
23             shells.
24         </p>
25     </body>
26 </html>
```

Figure 3.5: An example of the text HTML material.

3.4.3 DATA VISUALISATIONS

EyeMap uses an API in Flex to segment paragraphs into lines, words and characters to create the AOIs for each trial text. This is done based on font metric; it does this rather than analysing an image to get the word boundaries. It provides the users of EyeMap more freedom to create segmentation to aid in the creation of AOIs by incorporating a specific separator in the uploaded HTML text. This is important as there are languages like Chinese that have no word separation, so adding a separator allows the creation of AOIs for each word in the trial. As a result of the flexibility and strength of this EyeMap has been able to analyse texts in numerous languages with proportional and monospaced fonts.

Figure 3.6 shows an example of AOI's in EyeMap with varying languages, fonts and texts. In this sample, three language samples are presented. 1. A partial view of a Multi-line English text with a proportional font. 2. A partial view of a single-line spaced Korean text with a monospaced font. 3. A partial view of a single-line un-spaced English text with a monospaced font. These three examples show the versatility of EyeMap in handling different style fonts and languages.

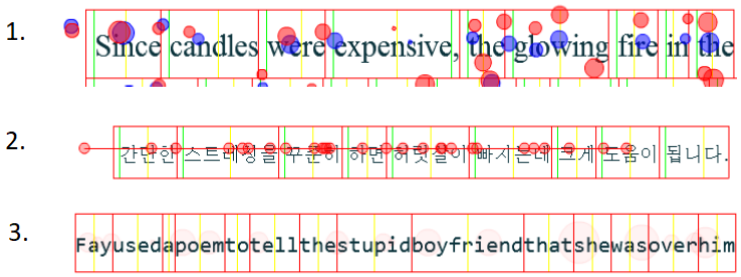


Figure 3.6: An example showing various texts displayed in EyeMap [47].

The next step, after the AOI's have been created, for EyeMap is to display the trial text on the screen with the fixation information for a participant. Each trial can be displayed on the screen in the viewer. It is also possible to change between trials for the participant. The fixations are displayed

along with the text for the trial and resulting AOIs, an example of this is presented in Figure 3.7. The viewer also displaying the word properties as mentioned in the Data Loader at Section 3.4.2. The fixations could be displayed with a proportional radius depending on the duration of the fixation or at a fixed radius. The other possibility is for a scan-path of fixations showing the direction and path taken by the experiment participant when completing this trial. Lastly, they could also be shown using a heat-map of regions where most fixations occurred.

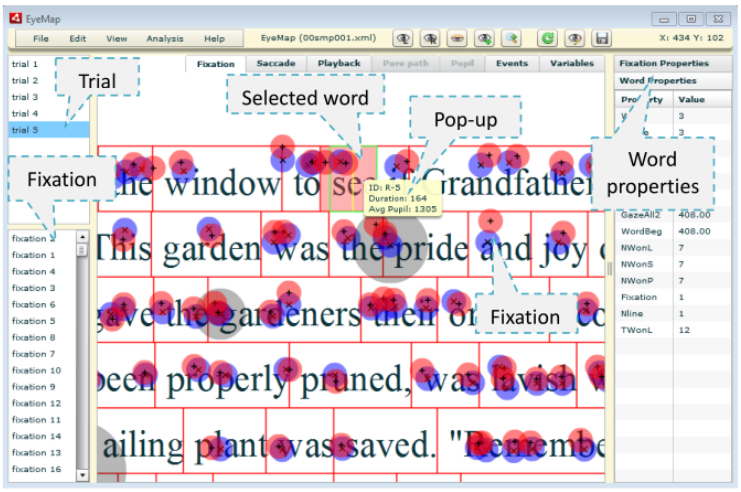


Figure 3.7: A zoomed fixation viewer from EyeMap [47]

3.4.4 DATA EDITOR

The Data Visualisations in EyeMap also allows the user to edit data. The person carrying out the analysis may need to correct fixations after drift. This is where a participant may have shifted in their seat causing their head height or angle of the head to change causing fixations to be above or below the correct positions. Fixations can be edited individually or in groups according to the needs of the user. Once edited the data would be updated in the corresponding nodes of the data tree structure.

3.4.5 READING VARIABLES

The reading variables are a key aspect of EyeMap. 150 variables are calculated by the Data Loader. They are calculated using the node structure described in Section 3.4.2. The variables are calculated using the node structure when the participant XML files are uploaded.

3.4.6 DATA EXPORTER

For the reading variables, the Data Exporter is key in EyeMap. It brings the other features together to generate a data matrix, which is the whole function of the EyeMap system. A range of variables can be selected for export through the viewer in the Data Visualisations (Section 3.4.3) and may output these calculated variables to a CSV file. This file contains the calculations along with the headers for each of the variables. The report is available in two formats, the first is a word level report focusing the variables around the areas of Interest and the second is the fixation report that focuses the variables around fixations.

3.5 THE NEED TO REDEVELOP EYEMAP

EyeMap was created to allow the user access to the wealth of data that is available from modern eye-tracking hardware. Being the first tool of its kind in the field, EyeMap has proven to be an excellent tool. However, after an in-depth discussion with active users of EyeMap, it was found that there are some issues within maintaining the system, while some users would like to add extra functionality.

While investigating EyeMap it was clear there would be a number of challenges to overcome. The first challenge was that it was difficult to access the code base for the system. The second was that it is also no

longer supported by the principal developer. These two challenges have a knock-on effect, as Flex uses Adobe Air to build the code, and as new updates are issued for Adobe Air the code base for EyeMap has developed bugs from deprecated code and dependency issues with libraries used to construct the system. With support no longer available for EyeMap this means that the required fixes to bugs in EyeMap are not being handled. It should also be noted that Flex has moved from an Adobe system to the Apache Open Source Community and this has further knock-on effects for EyeMap and its users. As the code base was not accessible the main challenge is to recreate all of the EyeMap 1.0 functionality from scratch.

Although EyeMap was developed as a Desktop application it later began to move to a web application. This was started by Dr Tang. He took the desktop version and used Adobe Flash to make it into a web application. The long term goal of this move was for researchers to share methodologies and data through a web interface. This further added to the draw of EyeMap and took the first steps in its long term goal. However, the system was never connected to a database so the functionality was never expanded. This also presented a further challenge as to how the data would be stored for use by EyeMap 2.0 users.

3.6 SUMMARY

This chapter has described the features and functionalities of EyeMap in detail and shown how it is an invaluable piece of software for reading research data analysis. The chapter also began to answer the research question "What is EyeMap and how can it be improved?". The main group of users for EyeMap are based at the Bergische Universität Wuppertal, Germany. These users have ideas of how to improve the functionality of EyeMap while maintaining the core system. The proposed improvements and extra functionality will be discussed in more detail in Chapter 4 with the

requirements for EyeMap 2.0 presented along with ideas for enhancements in going from EyeMap to EyeMap 2.0 from current users of EyeMap.

Part II

SOFTWARE REDEVELOPMENT

4

USER PARTICIPATION & SOFTWARE REQUIREMENTS

This chapter will further focus on the two research questions; "What is EyeMap and how can it be improved?" and "What software models, tools, and languages are appropriate for a project of this nature?". The chapter begins by introducing the end user team and describing how the project uses user participation and a prototype of EyeMap2.0 to build a software requirements or RAD task list for EyeMap 2.0. The RAD task list was construed based on the success of EyeMap and its features to try and create an improved software system in EyeMap 2.0. The chapter will continue to suggest tools and programming languages that would be appropriate for the development of EyeMap 2.0.

4.1 END USER TEAM

The three main users involved in the design and specifications of EyeMap 2.0 were also involved in the original design of EyeMap. They have a lot of expertise in reading research and study different areas of reading research using eye-tracking technologies. These users are:

- Professor Ronan Reilly from Maynooth University, Ireland. His research interests are primarily in cognitive science and he is interested in language understanding and reading. His background in psychology and computer science allows him to use computational modelling to develop theory.
- Professor Ralph Raddach from Bergische Universität Wuppertal, Germany. Prof. Raddach is a professor of psychology in the department

of General and Biological Psychology at the university and is a leading international expert in the use of eye-movement analysis to study information processing in reading.

- Doctor Christian Vorstius from Bergische Universität Wuppertal, Germany. Dr Vorstius is a lecturer of psychology in the department of General and Biological Psychology at the university. His research interests include the differences in reading behaviour and the cognitive factors of reading development.

Additionally for this redevelopment, Xi Fan, from Maynooth University, utilised EyeMap 2.0 for a Chinese reading language study. Xi Fan is new to the area of reading research and has not used the original EyeMap. She had over 100 participants in her experiment and she will be contributing to the project in the testing phases as EyeMap 2.0 is developed. The final member of the end user team is Patrick Hynes. Patrick is an Amazon Cloud Services engineer and was conducting an Irish language study and planned on using EyeMap 2.0 to analyse the data from his study.

It should be noted that Dr Vorstius and Prof. Radach are based in Wuppertal Germany, Mr Hynes is based in Munich Germany and Prof. Reilly, while based in Maynooth conducts a large portion of research outside of Ireland. Therefore regular meetings were difficult to hold unless they were carried out using online video conferencing. Before the first meeting could occur between all members of the team, a review of the original EyeMap system was performed to understand the different modules within EyeMap, including, all of its features and functionalities, as described in Chapter 3. Following this review, it was then possible to understand the different modules for development as is required for the RAD process. The different modules, discussed in Section 3.4, are: (i) Data Parser, (ii) Data Loader (iii) Data Editor, (iv) Data Visualisation, (v) Reading Variables and (vi) Data Exporter.

The original guidelines for the development of EyeMap 2.0 were to recreate EyeMap in a way that was easy to maintain and possibly expand upon in the future. For this purpose, an initial basic prototype was created and subsequently presented to users at the first official meeting, in January 2017, of the developer and end users.

4.2 PROTOTYPE

The prototype of EyeMap 2.0 was developed try and overcome the challenge of not having access to the EyeMap 1.0 source code and to test the functionality of different libraries for the different modules of the new software. The first consideration was the programming language that the software would be built in. There were several options including Java, C/C++, Flex (like the original EyeMap) and Python. To properly consider this issue, there were some considerations to be made that affect each of the modules of EyeMap 2.0. The considerations for each module are:

- Data Parser - The data would be converted to an XML format so the language must be able to read and process XML files.
- Data Loader - Participant XML files are approximately 400 KB in size and given a large number of participants in an experiment this can grow to be quite large. The language must be able to parse the files and organise the data appropriately.
- Data Editor and Visualisation - The graphical capabilities of the language must be considered here. The language must be able to display and manipulate the data as required.
- Reading Variables - The language must be able to carry out large numerical processing to correctly and efficiently create each of the variables for EyeMap 2.0.

- Data Exporter - The language must be able to export the data for analysis use later, in a universal format that can be used on any operating system.

Flex was ruled out due to the issues that crept into the code base, when it was not maintained, by updates to the Adobe Software. Java and C++ are both appealing languages and have been shown to have rich libraries especially for GUI implementation [11]. In a study investigating the pros and cons of Java for scientific computing, Gudenberg concluded that he would not recommend Java for this purpose due to expensive and inefficient workarounds that are required [19]. In a comparative study of programming languages, Prechelt suggests for certain tasks that Java has a large memory overhead compared to C/C++ but its efficiency is acceptable. He also suggests that a scripting language such as Python performs very well with tasks that have large data and computation while offering advantages in terms of programmer productivity [36]. Given the developers' knowledge of Python and the conclusions by Prechlt, it was decided to use Python to develop EyeMap 2.0. It would also allow EyeMap 2.0 to be made open source at a later stage as Python is widely known and would allow other developers, and researchers, to contribute to the project as they see fit, to coincide with the expanding technology in the eye movements area.

To create the EyeMap 2.0 prototype PySide was utilised. PySide is the Python version of the QT framework, software originally written in C++, for rapid GUI development [37]. PySide focuses on making desktop applications that will run on any operating system. The prototype was a desktop version and quite basic; it was designed to test the QT framework. It allowed for the upload of a single participant XML file and then displayed the data from that file to the screen. It should be noted that there were issues around the editing of the data at this point. However, it pro-

vided the basic idea of what EyeMap 2.0 could look like and showed some basic processes of the Data Loader and Data Visualisation modules.

4.3 TEAM REQUIREMENTS MEETING

The first team meeting for EyeMap 2.0 was held in the Department of Psychology at the Bergische Universität Wuppertal over two days in January 2017. In attendance at the meeting was Prof. Ralph Radach, Dr Christian Vorstius, Prof. Ronan Reilly and the author. This was a significant meeting for the project as it represented the first opportunity for requirements gathering from experts in eye-tracking and users of the original EyeMap that the users required in EyeMap 2.0. The meeting began by discussing the important features and functionalities of the original EyeMap. The topics discussed will be presented in Section 4.4. Each requirement was also given a need and priority level to show how important that feature was. The need levels used were **Required**, **Want** and **Nice** and the priority levels are **Low**, **Medium** and **High**. **Required** is an essential feature of EyeMap and must be included in EyeMap 2.0. A **Want** is a redesigned feature or some new feature that should be included in EyeMap 2.0, and **Nice** is not an essential feature but would add to the features of EyeMap 2.0. For the priority levels **Low** is not a feature that is required urgently while **High** relates to a feature that should be developed quickly.

After the initial discussion Dr Vorstius completed a walk through of how he uses the original EyeMap. This process was invaluable as it provided an opportunity to highlight what worked well in the functionality and what didn't as well as possible features that could be added to the system. The walk through stepped through each module of EyeMap from the Data Parser to the Data Exporter, and he discussed what feature would be **Required**, a **Want** or **Nice** as well as their associated priority level. These requirements will be discussed in more detail in Section 4.4.

The prototype that was developed was also presented at the meeting. The prototype was well received and it was agreed that it was a good proof of concept. Figure 4.1 presents a screenshot of the EyeMap 2.0 prototype which shows the fixation screen with the fixation dots in red for the right eye and in blue for left eye on a trial for an experiment with AOI's.

Figure 4.1: prototype of EyeMap 2.0.

However, given the new requirements, for a web based application, that was received during the walk through, it was no longer fit for purpose as a desktop application. This, unfortunately, meant that PySide was also surplus to requirements and a new framework was required.

4.4 REQUIREMENTS

The main requirement for the project was to develop EyeMap 2.0 and the team requirements meeting provided further software requirements in addition to the features of EyeMap that would be implemented. These requirements are presented in relation to each of the modules presented in Section 3.4. The features for EyeMap 2.0 & associated module will now be presented along with the need and priority level of the feature along with a description.

4.4.1 REQUIREMENT FEATURES

- Loading Multiple Participants
  - Module : Data Loader
  - Need Level: Want
  - Priority Level: Medium
  - Description : An important first step with EyeMap 2.0 is the pre-processing and subsequent uploading of the data files that cover not only the trials for a single participant but all trials for all participants. The requirement here is to process an entire folder in one go and not have the time consuming process of loading each file individually. When this pre-processing is completed, the folder is then uploaded to the system where the researcher can view all participants with their files without having to upload an XML data file for each participant.
- Data Storage
  - Module : Data Loader
  - Need Level: Required
  - Priority Level: High
  - Description : The participant data XML files need to be uploaded and stored to EyeMap 2.0. The files need to be easily accessible from anywhere. This requirement suggests that EyeMap 2.0 should be made into a web application. This would require the files to be stored in a secure database that can be easily accessed by the end user on any computer with an internet connection. Storing the files in this manner would overcome the challenge of users accessing their data and would also assist

with collaboration between researchers, which will be discussed in this section later on.

- Login Functionality
  - Module : Data Loader
  - Need Level: Required
  - Priority Level: High
  - Description : As EyeMap 2.0 would be a web application the system would require secure login functionality to allow users easy access to their experiments and data files.
- Database
  - Module : Data Loader
  - Need Level: Required
  - Priority Level: High
  - Description : As data files are going to be accessible from a web application they need to be stored in a database. This then raised the question of how to store the data files. Should these be loaded from the file as required or is there a better way to load the required files?
- Languages and Fonts
  - Module : Data Loader
  - Need Level: Required
  - Priority Level: High
  - Description : The original EyeMap has a number of quality features that exist around the languages that it can display and interact with while also using proportional and non-proportional fonts for creating AOIs. Different languages have different fonts associated with them and the system would require different

fonts to be available in EyeMap 2.0 to maintain the functionality provided by EyeMap of being able to handle different languages.

- Visualisations
  - Module : Data Visualisation
  - Need Level: Want
  - Priority Level: Medium
  - Description : There are a number of visualisations that exist within EyeMap. They relate to Fixations, Saccades, Scan Paths and Playback. The visualisations are extremely useful in EyeMap and needed to be maintained in EyeMap 2.0. Having the capability to be able to interact and export each visualisation would be extremely beneficial for EyeMap 2.0.
- Variables
  - Module : Reading Variables
  - Need Level: Want
  - Priority Level: High
  - Description : The experimental variables in EyeMap are a key quality feature in the system. They need to be re-implemented in the new system to the same high accuracy and standards. The requirement here is to allow customisation and the possibility to create new variables that can be shared with different researchers.
- Collaboration
  - Module : EyeMap 2.0 System
  - Need Level: Nice
  - Priority Level: Low

  - Description : With any ongoing experiment there can be a number of researchers working on a project, so it is important that they are able to share their data easily with each other. Additionally, as research progresses it is becoming more important for researchers to share their data with collaborators set to verify results. With EyeMap 2.0 it should be easy for the end user to share the data as required both in terms of granting access on-line and providing hard copies that can be exported from the system.

- Batch Export
  - Module : Data Exporter
  - Need Level: Want
  - Priority Level: Low
  - Description : The original EyeMap exports the variables for a single participant of a single experiment for additional analysis. The ideal solution would be to allow the calculated results of multiple participants to be exported at once, potentially in the same output file. This would involve and investigation into how the data is stored and ultimately processed.

- New Variable Calculations
  - Module : Reading Variables
  - Need Level: Want
  - Priority Level: Medium
  - Description : Since the creation of EyeMap there have been new variables proposed that have proven to be useful for reading analysis. The requirement would be to add these variables to EyeMap 2.0.

4.4.2 REQUIREMENT FIXES

The following is a list of fixes for features of the original EyeMap for EyeMap 2.0. Each fix has the module from EyeMap that it relates to, the need and priority levels of the fix and a description.

1. Edited fixations changing the fixation order in the viewer
    - Module : Data Editor
    - Need Level: Required
    - Priority Level: High
    - Description : When editing is carried out on fixations in the data editor some of the fixations change the order in the view panel on the left of the visualisation screen. This is a display error and is required to be fixed for the editing to be carried out successfully and completely.
2. Segmentation of Chinese words
    - Module : Data Loader
    - Need Level: Required
    - Priority Level: High
    - Description : It is important that EyeMap 2.0 works with multiple languages. With English or German it is easy to distinguish between words because they are separated by a space. However, with Chinese there is no obvious separator of words and it is important to allow some indicator to show the separation of words.
3. File naming conventions
    - Module : Data Loader
    - Need Level: Want

- Priority Level: Low
- Description : In EyeMap the file that contains the text material for the experiment must be called *text.html* for each experiment that is loaded into the system and this can be quite difficult when there are two or three variations of an experiment that each have multiple participants. Therefore, a way to differentiate between the text upload would save large amounts of time and energy.

4. Decimal numbers separator
    - Module : EyeMap 2.0 System
    - Need Level: Nice
    - Priority Level: Low
    - Description : In Ireland decimal numbers are written with the "." separator, for example "2.5". However, in Germany the "," is used instead; our same example would be written in Germany as "2,5". Is it possible to fix this for the data that is exported from EyeMap 2.0?

4.5 EYEMAP 2.0 ARCHITECTURE

As discussed in Section 4.2 Python was the programming language of choice for the development of EyeMap 2.0. In relation to the choice of Python, there was a decision around two different frameworks, Django or Flask. Django is a complete framework with the Model-View-Controller (MVC) that is database driven with its own built-in user model which handles API authorisation and authentication, while Flask is more lightweight and requires the setup of different libraries or tools for different functionality. Given the system requirements, as described in Section 4.4 require

a web application with secure login features and is also database driven Django was chosen as the architecture for EyeMap 2.0

4.5.1 DJANGO

Django was publicly released in 2005 and it is currently run and maintained by the Django Software Foundation as a non-profit organisation [10]. Django is built on python and uses HTML, CSS and JavaScript for the development of a web application. It was designed to be fast and reliable and developed in a way that would allow developers to quickly and easily take a project from the design stage through to implementation. It has many features which include, a standalone server for development and testing, an easy to use expandable template system to allow for code re-usability and changeability, and a form serialisation and validation system which translates between HTML forms and values suitable for storage in the database. Security is also provided by Django as it prevents against a number of attacks such as cross site scripting, where users are allowed to inject client side scripts into the browsers of other users, or SQL injection, an attack where a user can execute arbitrary SQL code on a database.

4.6 SOFTWARE DEVELOPMENT TOOLS

As the project would be designed and developed between Ireland and Germany with several users, it was important that the code base used a version control system. This prevents any issues with different versions of the code getting confused between the two countries. The version control system that was used in this development process was GitLab.

4.6.1 VERSION CONTROL

Version control is a system that records changes to a file or a set of files over time so that you can recall specific versions later if required. It allows multiple users to work on a project without having to keep track of who has the latest version of the project or worrying about getting versions of the project confused, and delaying work due to simple errors in not using the latest version. The project uses GitLab as its version control tool, which is built on git. Git is the underlying structure for GitLab. It takes a snapshot of the files on a project and labels them so they can be accessed at a later time. Git also marks who has the latest version and reduces errors with multiple people working on a project. You can roll back changes if they were made incorrectly and keep changes in a separate snapshot or "branch" if they are not ready to go into the project. Using git means that a central repository is created for the project.

Every repository has a master branch, a branch is just a pointer to a snapshot of the code base. The master branch is where the fully working code is kept. This branch should have no bugs or errors and contains the complete working code for the project. To aid with the RAD process, a technique or workflow for development called Gitflow was used. Gitflow is a process during the development of a project that creates several branches to enhance and improve the workflow. It begins with a master branch and then for each new feature a new feature branch is created. This means the code currently being developed is kept clear of the fully working code on the master branch. Once the feature has been completed the feature branch is then merged back into the master branch. Gitflow also has branches for preparing, maintaining, and recording releases of the project.

The workflow of Gitflow, as described in an Atlassian BitBucket tutorial [17], can be seen in Figure 4.2. The workflow can be described as:

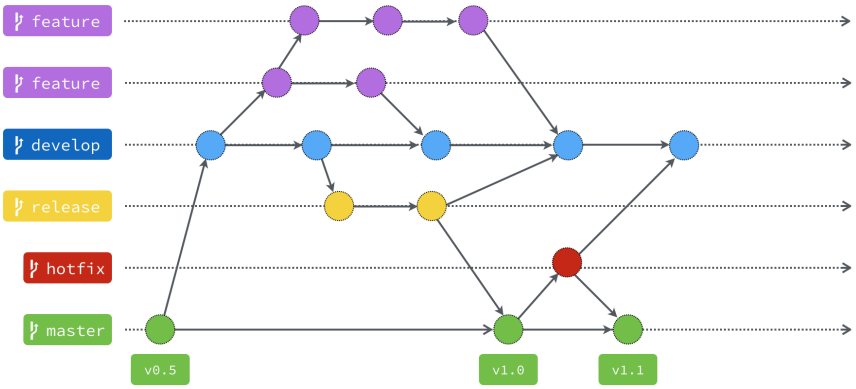


Figure 4.2: Example of Gitflow [16].

1. A *develop* branch is created from the *master* branch
2. A *release* branch is created from the *develop* branch
3. *Feature* branches are created from the *develop* branch
4. When a *feature* is complete it is merged into the *develop* branch
5. When the *release* branch is done it is merged into *develop* and *master*
6. If an issue in *master* is detected a *hotfix* branch is created from *master*
7. Once the *hotfix* is complete it is merged to both *develop* and *master*

The next tool that is required is an Integrated Development Environment (IDE) for development that works with git.

4.6.2 INTEGRATED DEVELOPMENT ENVIRONMENT

For the development, PyCharm from JetBrains [27], was used as the IDE. PyCharm provides support for Django web applications as well as providing access to GitLab for version control, it also manages requirements of libraries and dependencies that would be used in the course of the project.

4.6.3 ERROR LOGGING AND BUG TRACKING

An important part of the RAD development process is the continued integration and testing of new modules as they are introduced to the project. This was another consideration for the team and the ability to report issues and track their progress without long email chains. To manage this problem a web tool called Backlog was employed. Backlog [1] is an issue tracking management tool that allows users to report bugs and issues in a system. Figure 4.3 shows a blank version of a ticket form from Backlog to track issues. It can be seen that everything can be captured and reported about an issue or problem, the issue can be assigned to a user or developer as well as allowing screenshots of the issues to be posted.

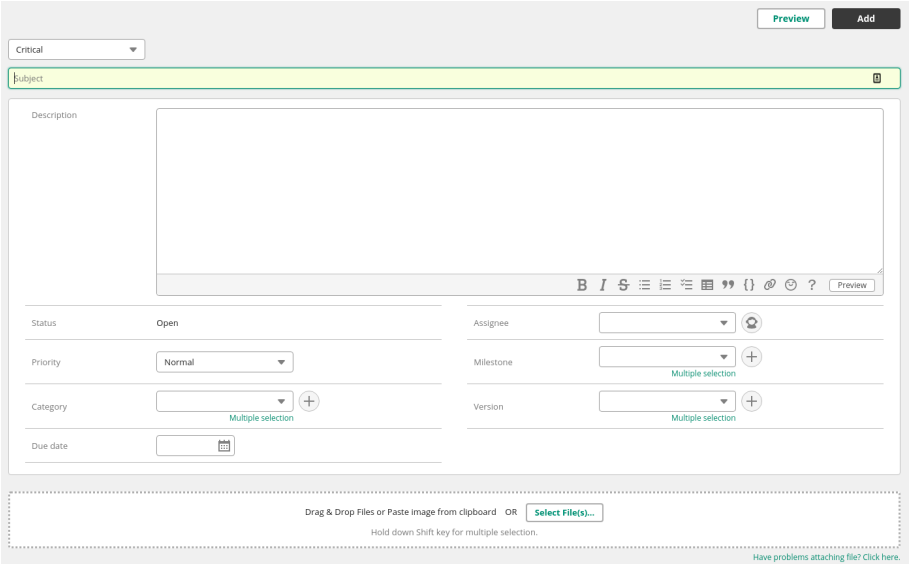


Figure 4.3: Backlog issue creation screen.

With all of these tools and frameworks in place the development of EyeMap 2.0 could begin.

4.7 SUMMARY

The initial meeting for the development of EyeMap 2.0 could be considered a success. It allowed the end user team and developer to build a clear software requirements list that could improve on the original functionality of EyeMap. It was clear that EyeMap 2.0 would need to be a web application, as this would provide the extra functionality in terms of users accessing their data anywhere and on any operating system. The end user team provided extremely valuable information about the features of EyeMap that could be improved. They set clear goals for the architecture of EyeMap 2.0 and what was required of it, which provided the list of requirements and fixes as presented in Section 4.4. In Sections 4.2, 4.5 and 4.6 it was demonstrated that Python would be the programming language of choice for the development of EyeMap 2.0. This decision then provided PyCharm as the IDE for the project with Django discussed as the new architecture. A GitLab repository combined with Gitflow would also enhance the tools and frameworks for the development of EyeMap 2.0 to be successful. Chapter 5 will provide more detail on how these frameworks, tools and structures were used to redevelop EyeMap 2.0.

5

SYSTEM REDEVELOPMENT

As discussed in Section 2.4 the RAD software development process was used for the redevelopment of EyeMap 2.0. This means that the project was broken down into different modules for RAD were each module represents a different task for the project. This chapter presents the modules from the RAD task list and discusses how each of them was developed.

5.1 CHALLENGES TO OVERCOME

Section 3.5 showed that there were some challenges to overcome in the redevelopment of EyeMap 2.0. These needed to be considered at every stage during the redevelopment. These challenges were:

- It was difficult to access the source code for EyeMap 1.0, which would mean that each function of EyeMap 2.0 would have to be re-created from scratch, with the users and working version of EyeMap 1.0 providing guidance on correct functionality.
- As EyeMap 2.0 was being developed as a web application the data for experiments would need to be stored in a secure and safe manner that was easily accessible for users.

5.2 RAD TASK LIST

The modules for the development of EyeMap 2.0 were derived from the modules that existed in EyeMap as described in Section 3.4. The task list

for the RAD process followed these modules. A task was also made for setting up the core Django framework and initialising the GitLab repository. The following is a list of modules or tasks that were created for the RAD process:

1. Set up Core Architecture
2. Initialise GitLab Repository
3. Design and plan the layout of EyeMap 2.0
4. Create the Database for EyeMap 2.0
5. Set up Templates
6. Create Web Pages
7. Data Loader
8. Data Visualisation & Editor
9. Data Exporter

Note: The Data Parser from the original EyeMap is a standalone tool that is written in a combination of Java and Bash scripting, for pre-processing of participant XML data files before they are uploaded to EyeMap. It is fully functional and does not require an update.

As described previously the development of EyeMap 2.0 followed the RAD process. Therefore each task had 3 smaller sections to it (as described in Section 2.4.2.4 Figure 2.16). These sections are:

1. Design
2. Construction
3. Testing

Note: The testing for each section consisted of the system, at that point in time, being sent to the development team and feedback received being incorporated into the next iteration of EyeMap 2.0. The testing will be discussed in more detail in Chapter 6. The following sections will focus on the design and construction of EyeMap 2.0.

5.3 CONSTRUCTING THE CORE ARCHITECTURE OF EYEMAP 2.0

Setting up the core architecture was the first task in the RAD task list. Django creates the basic file structure when starting a project. This includes all of the initial settings files and required files for running your Django project. An example of this can be seen in Appendix C.1.

5.3.1 LANGUAGES USED IN THE DEVELOPMENT OF EYEMAP 2.0.

As EyeMap 2.0 is a web application other programming languages and libraries were included in the development process. These languages are HTML, CSS and JavaScript. To incorporate these languages into the project a static folder was created in the project that would load CSS and JavaScript script files. They could then be called from any page within EyeMap 2.0. Django uses a template system for HTML that will be discussed in more detail in Section 5.7.

The CSS framework that was used in the development process was Bootstrap 3 [5]. Bootstrap 3 is an open-source set of tools that allow developers to create responsive and functional website designs. Bootstrap 3 consists of HTML and CSS templates that create various types of components for websites, or user interfaces, such as buttons, forms and navigation components. It also includes some JavaScript extensions. The main benefit of Bootstrap is the responsive design that it employs. The responsive element means that while creating the interface for a project, the interface would

render correctly on any number of devices, such as smartphones, tablets, and PC or laptop screens. Bootstrap is also compatible with all modern browsers such as Chrome, Firefox, Safari and Opera.

5.4 GITLAB REPOSITORY

To use Gitflow in the development of the project a git repository was set up on GitLab. All the files generated from the project startup were included in the repository. The initial branches created for development were:

- Master - Used to keep the working project files; the *Live* branch mirrors this.
- Live - This is the code base that is hosted on the live web application.
- Dev - This is the main development branch. As a new feature was required a new branch was created from *Dev* for the feature development. As the feature was completed, the branch associated with it was merged back into the *Dev* branch and then onto the Master branch for testing and release.

5.5 DESIGN AND DEVELOPMENT OF EYEMAP 2.0'S LAYOUT

Figure 5.1 presents the initial design of EyeMap 2.0. As functionality was added this figure was the main source of reference for where new features were added.

In Figure 5.1 each structure within the EyeMap 2.0 system is a separate web page within EyeMap 2.0. It was envisaged that the end user would open the EyeMap 2.0 website and register as a user. They would then have the ability to login to the system. Once they successfully had created an account, they would be returned to the Home Page where they could then create a new experiment or select an existing experiment to work with.

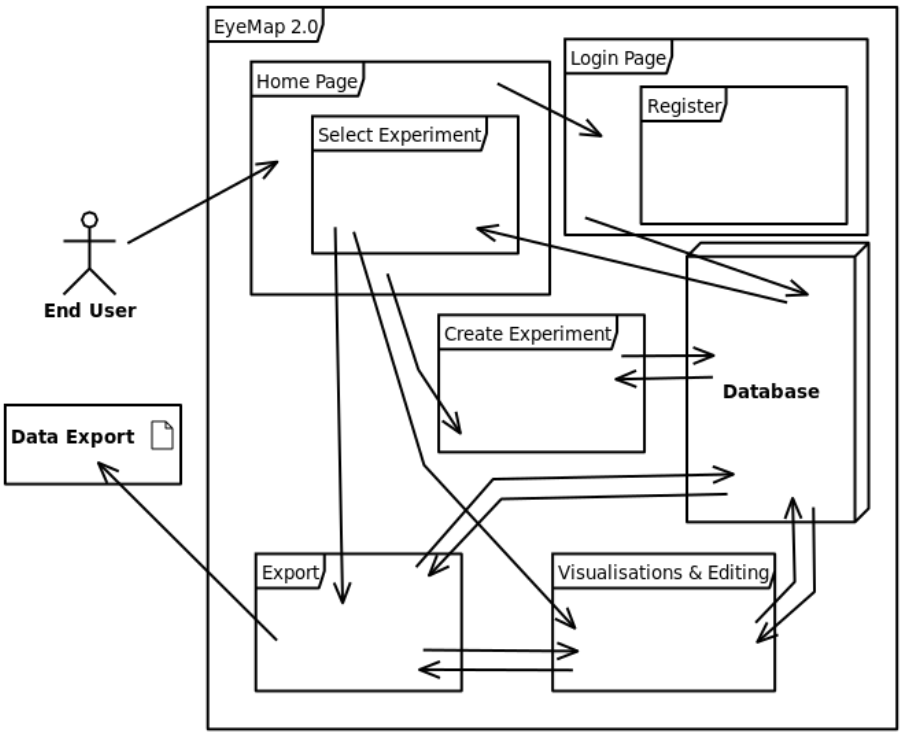


Figure 5.1: EyeMap 2.0 System Design.

If they chose to create a new experiment they would be guided through uploading all the relevant files using the Data Loader (see Section 5.9).

At this point, the user would then be able to view visualisations of the different trials for different participants in their experiment using the Data Visualisation Tools (see Section 5.10). They could also skip straight to calculating the reading variables (see Section 5.11) on the export page or navigate to this page after they have made edits to the trials for a participant if required. Once they are satisfied with the experiment they can then export the data to a file for further use (see Section 5.11) with the Data Exporter.

5.6 DATABASE CREATION

A large challenge in the redevelopment and requirement for EyeMap 2.0 was the secure storing and maintenance of the different data files for ex-

periments that were loaded into the system. The files need to be quickly processed and displayed, be it for visualisations, editing or reading variable calculations. This section will analyse the creation of the database using the inbuilt Django SQLite3 database.

5.6.1 DATA FILES MANAGEMENT

The first constraint in designing the database was the number and size of files in any given experiment. Each experiment has XML data files, an XML configuration file and a text stimulus HTML file. The experiment can have any number of participants, and have any number of trials per experiment. As an example, for one participant in an experiment with five trials the XML data file size for this participant was 432.1 KB. Having more participants with more trials in an experiment means the file size is going to grow exponentially, which will use more valuable memory space in the database. The solution to this problem was to change how the files are stored within EyeMap 2.0. The solution involved storing the file broken up as a JSON String. JSON is a human-readable text written in JavaScript object notation so that it can be easily passed between applications and services.

Each participant file can be broken into fixation data, saccade data and drift data and a JSON string is constructed for each of these. The size of the JSON strings combined, for the same experiment and participant with the same trials used previously, is 376.4 KB. While this is not significantly smaller than the file size itself it does mean that the system does not need to create a file structure for any uploaded files and then store the location of the file in the database. The other deciding factor for storing the data as JSON strings were that JSON is easily passed by requests going from the server side to the browser. This means the data can be prepared using python code on the server side and then passed to the JavaScript code

on the front-end for display purposes. This also helped in overcoming the challenge for storing safely and accessing data required for a users experiment.

5.6.2 DATABASE STRUCTURE

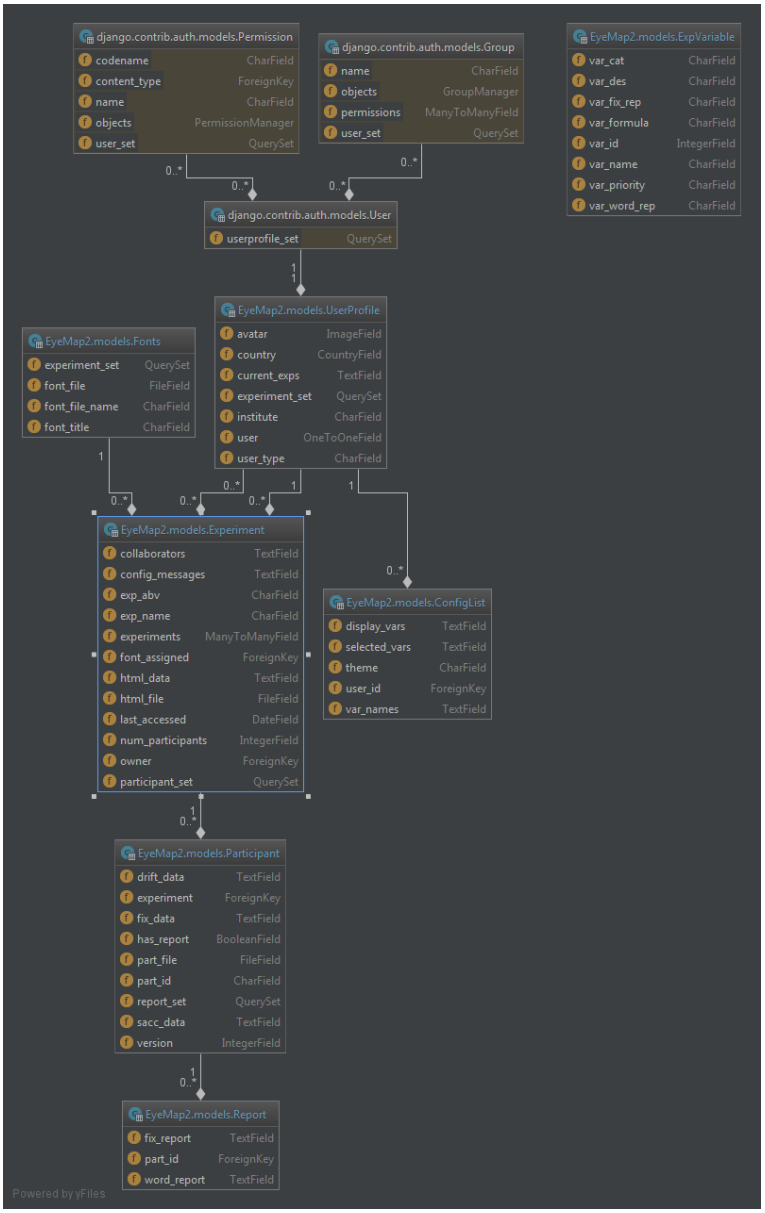


Figure 5.2: Full EyeMap 2.0 Database.

With the decision made to use JSON strings for storing data, the database was then designed to be focused on the experiment at the centre. The initial database size was small but as the complexity of EyeMap 2.0 grew the database was changed and extended to match the complexity and other requests received from the end users. Figure 5.2 shows this complete database structure of EyeMap 2.0. The final database was designed so that when a user creates an experiment, which then is associated to the user, the details for the experiment are extracted from the experiment files as they are uploaded to EyeMap 2.0 (for further information see Section 5.9.). When the participant XML files are loaded into the system they are stored into the participant table in the database, with each participant belonging to a given experiment.

Each experiment also has a related font file. The font files are pre-loaded in the system. The ability to add more fonts on request exists, this was done as a security feature. End users are not permitted to upload binary files as these may damage the system.

5.6.3 GENERATING THE TABLES FOR THE DATABASE

Each of the tables shown in Figure 5.2 are generated by Django with the code for doing this written in Python. To see an example implementation of this please see Appendix C.2. Once the database was created, the Web application needed each web page from the design in Figure 5.1 to be created and prepared.

5.7 SET UP TEMPLATES

As discussed in Section 4.5.1 Django has a template system that easily allows web pages to be generated dynamically with the Django template language (DTL). This means that web pages can be modularised so that a

section that is required across multiple pages can be separated and called by Django when required. Appendix C.3 hows a detailed example of how the template system is implemented and extended. Once the template system was set up it was then possible to set up the other web pages for EyeMap 2.0.

5.8 WEB PAGE CREATION

The next step in the development process was to create each of the web pages shown in Figure 5.1. These are:

- Home Page
- Login Page
- Registration Page
- Create Experiment
- Visualisations & Editing
- Export

Django has specific configurations for creating a web page, this can be seen in more detail in Appendix C.4. Each web page layout was developed using the DTL with HTML for the structure, CSS for the look and JavaScript for the functionality. The main web pages of *Create Experiment*, *Visualisations & Editing* and *Export* will be discussed in Sections 5.9, 5.10 and 5.11 respectively. All of the different web pages were presented to the end user team as they were developed. The users would then present their feedback through the backlog bug and issue tracker system by opening a ticket. These tickets were investigated, fixed, sent back to the user for further testing and when verified, incorporated into EyeMap 2.0.

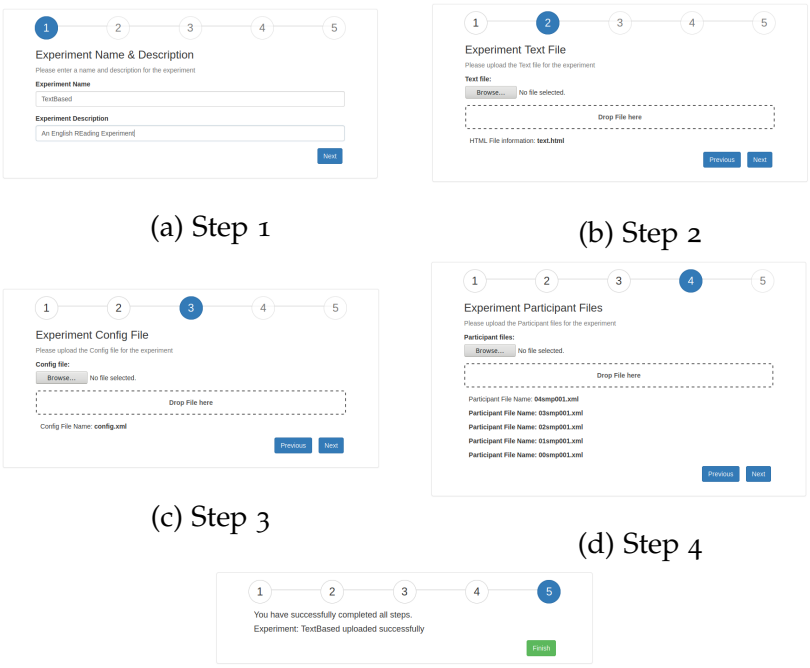
5.9 DATA LOADER

The *Create Experiment* page of EyeMap 2.0 allows a User to create a new experiment. There are several different data files that are required for each experiment. These are:

- Trials File, *text.html* - This does not have to be called *text.html*, as defined in the user requirements, but for the purposes of the document it shall be referred to as *text.html*. This file contains the trials for the experiment.
- Configuration File, *config.xml* - This must be named config.XML. This is the configuration settings to go along with an experiment.
- Participant Files *oosmp001.xml* (sample file name) - The number of the participant files can vary depending on how many participants exist in the experiment.

The *Create Experiment* web page was originally laid out on one page with multiple HTML form sections to take in each of the files. However, during a review with the end user team, this proved to be confusing because of all of the different files types. There was confusion around whether the files had loaded correctly and in the right order. During the next round of development, it was decided to turn the *Create Experiment* web page into a Wizard to upload the files. In technical terms, a wizard is a step by step guide to completing a certain task. The completed wizard for the upload process can be seen in Figure 5.3, and has five steps which can be described as:

1. Figure 5.3a: Step 1 of the wizard asks the users for the experiment name and description.
2. Figure 5.3b: Step 2 of the wizard requires the user to upload the Trials file containing the trials of the experiment. How this is completed



(a) Step 1

(b) Step 2

(c) Step 3

(d) Step 4

(e) Step 5

Figure 5.3: The upload Experiment process.

will be discussed more in Section 5.9.1. It also checks if the required font is already stored in the system for the experiment. If it is not present the user is asked to contact the system administrator.

3. Figure 5.3c: Step 3 of the wizard is provided for the user to upload the experiment configuration file, *config.XML*.

4. Figure 5.3d: Step 4 of the wizard allows the user to upload multiple participant files by dragging and dropping them on the indicated area. Section 5.9.2 will discuss further how the participant and config files are processed.

5. Figure 5.3e: Step 5 of the wizard sends all of the gathered data to the server side to be stored in the database and proceeds to provide the user with confirmation that the experiment was uploaded correctly to the system. Once the user clicks the *Finish* button, they are re-

turned to the home page where their experiment will be displayed along with all other experiments belonging to the user.

5.9.1 PROCESSING THE TRIALS FILE

On upload, in the wizard, the *text.html* is processed before the user can proceed to the next step. The first step in this process is to convert the file into a JSON String. It is then passed, by request, to the server side where the font is checked to ensure it exists in the system. If it does not exist the user is advised to contact the system administrator to get the font file added to EyeMap 2.0, and they are returned to the Home page. If the font does exist in the system the system will continue to process the *text.html*.

```
1  <html>
2      <head>
3          <style type="text/css">
4              body
5              {
6                  color:black;
7                  font-family:SimSun;
8                  font-size:32;
9                  leading:32;
10             }
11         </style>
12         <x>100</x>
13         <y>392</y>
14         <sep>/</sep>
15     </head>
16     <body>
17         <p>大兴安岭/自/山脚/至/岭顶/长满/了/珍贵/的/树木</p>
18         <p>大兴安岭/山脚下/长满/了/珍贵/的/草药</p>
19         <p>社会/适应/能力/反映/了/人/与/社会/的/协调/程度</p>
20         <p>长期/食用/方便/食品/会对/健康/造成/很大/危害</p>
21         <p>偶尔/食用/方便/食品/会对/健康/造成/很大/危害</p>
22         <p>没有/太阳/就/没有/我们/这个/美丽/可爱/的/世界</p>
23         <p>报纸/可以/提供/给/我们/许多/非常/有用/的/信息</p>
24         <p>许多/非常/有用/的/信息/是/报纸/提供/给/我们的</p>
25         <p>人/最重要/的/事/是/活出/自己/的/特色/和/滋味/来</p>
26         <p>人/最重要/的/事/是/让/生活/充满/特色/和/滋味</p>
27         <p>练习/结束/，/实验/正式/开始</p>
28         <p>王先生/和/其他的/走马看花/的/观光者/没什么/两样</p>
29         <p>老师/正在/和/书店/联系/小学生/能力/训练/丛书</p>
30         <p>戏曲/等/展现了/源远流长/的/中华/传统/文化/艺术</p>
31         <p>如果/他/识时/务且/知道/见风转舵/或许/不会/那么/惨/font=宋体</p>
32     </body>
33 </html>
```

Figure 5.4: *text.html* for a Chinese language experiment.

Figure 5.4 shows a sample *text.html* file for a Chinese language experiment. The code first extracts the experiment information, which is shown on lines 3-13 of the sample file. This information is the form of font, font

sizes, leading for the trials and the starting X and Y location on the screen. Another requirement provided by the users was the need for a separator for the Chinese language or other languages that do not use spaces between words. Line 14 shows a custom made <*sep*> tag for this purpose. Within the <*p*> tags are the text stimulus for each trial. For this experiment, there are fifteen trials. On closer inspection of any of the <*p*> tags the Chinese words are separated by the separator provided on Line 14 of the file. The data for each trial is next to be extracted from the <*p*> tags. The system processes each trial into a list of words based on the separator or a space if the <*sep*> tag is not present and then makes a list of all the trials. Once all of the processing is complete the system allows the user to continue to the next step of the wizard.

5.9.2 PROCESSING DATA FILES

During the next two steps of the wizard, the XML data files are processed. As the *text.html* is converted to a JSON string and sent to the server in a request object, the same process happens with both types of XML files. The data from *config.XML* is processed into a JSON String. For the participant XML files there are three lists of python dictionaries created, the first is for *Drift* data, the second is for *Fixation* data and the third is for *Saccade* data. Each trial is contained in its own list within the larger list.

Once all of the data has been processed and the user reaches step 5 of the wizard the data is all then stored in the database in the relevant locations. The experiment is then added to the list of users experiments on the Home page for selection. This can be seen in Figure 5.5. When the user selects an experiment and a participant within that experiment the data is retrieved from the database and is then ready for the user on the next web page that they navigate to.

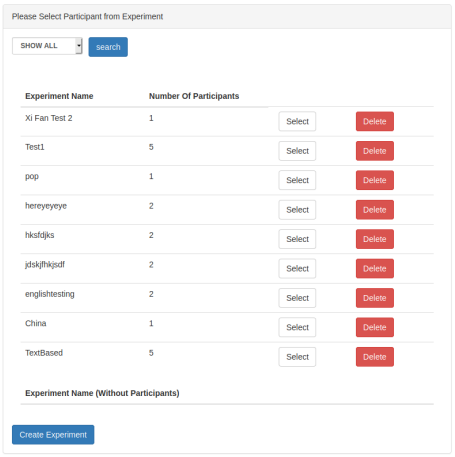


Figure 5.5: Users Experiment list presented on the Home page.

5.10 DATA VISUALISATION & EDITOR

Once the user selects the experiment and participant that they wish to work with, they can navigate to the Visualisation & Editing page. The data for the participant is retrieved from the database and loaded into a response object and sent to the browser through the view for the page where it is then extracted and ready for processing. There are two stages to the data processing on this page. The first is to get the AOI Information and the second is to prepare the data for visualising and editing. These will be discussed in the following sections.

5.10.1 EXTRACTING THE AOI INFORMATION

The first step in the data preparation process is to get the AOI Information. All of the trials are contained in a list, and each trial is a list on its own, that contains all words for the trial. The information about the experiment is also contained within a list and has the font, font size, leading for the trials and the starting X and Y location. This information is important for the

experiment as it helps to get the font metrics for each AOI and ultimately decides where the AOI is located on the screen.

The font metrics are calculated using a JavaScript library called opentype.js [8]. This is a TrueType and OpenType font parser and writer. The opentype.js library allows EyeMap 2.0 to satisfy the requirement for multiple languages and fonts. It uses the font file for the font used in the experiment to parse the words of the trial to decide on the height and width of the AOI surrounding a given word or character. Once all words have been parsed and the font metrics are received for each word they are then ready to be displayed on the screen. An example of this can be seen in Figure 5.6a. In this Figure, the words from the trial can be seen with their AOI. The black border on this page represents the screen that the participant used during the running of the experiment.

5.10.2 DISPLAY THE DATA

After the AOIs have been calculated all of the data must be displayed on the screen. The first design for this involved using HTML5 canvas. While the canvas drew the objects on screen very well it proved difficult to use when it came to clicking, selecting and hiding objects on the canvas. During the second stage of the design stage another JavaScript library, called Konva.js [53], was tested as it utilised the HTML5 canvas. Konva is a JavaScript library that allows grouping of objects and clicking, selecting and hiding objects on the HTML5 canvas.

Konva allowed the different elements of the data to be grouped together and the groups to be layered differently within the canvas, which further allows the different elements to be selected. This grouping allows all the AOI data to be created and layered together. For this the words and bounding boxes that make the AOI's were grouped separately. Options were then added to show and hide each of the elements and when an AOI is

selected a separate panel displays the AOI information. The AOI Tab of the visualisation & editing page can be seen in Figure 5.6a.

(a) AOI tab. (b) Fixation Tab.

Figure 5.6: The different tabs on the Visualise & Edit page.

Figure 5.6b shows the Fixation Tab on the same visualisation & editing page. The fixations were separated into left and right eye layers for the canvas. This provided functionality to show and hide the left and right eye fixations. Each fixation can also be selected and, like the AOI's, displays individual fixation information in a separate information panel.

Rather than navigating to a separate web page an option to turn on and off editing functionality was added to the page. This allows users to move fixations in the Y direction only. Fixations can be selected in groups or individually and when the editing option is enabled they can be moved on-screen. It is only when the user saves the edits that the data file is updated and a new entry is created in the database for the participant. This is marked as version 2 so that the original is maintained at all times.

Once all editing has been completed the user can navigate to the export or analysis web page for the reading variables to be calculated for the participant.

5.11 DATA EXPORTER

The analysis of the data takes place within the data exporter web page. With the reading variables for export being of core importance to the original EyeMap, this module had the most involvement from the end user

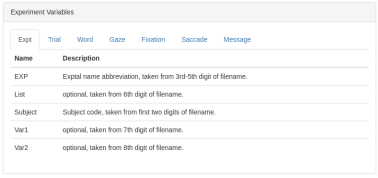
team. There are currently over 150 reading variables within EyeMap. As described in Section 3.4.5 there are six levels of variables available, which are:

1. Experiment level
2. Trial level
3. Word level
4. Gaze level
5. Fixation level
6. Saccade level

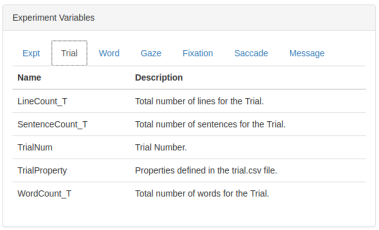
Each level has its own set of variables associated with it. They are all available on the analysis page of EyeMap 2.0 (full list provided in Appendix B.1). Figure 5.7 shows a sample of the variables and a description from the analysis page.

To ensure the correct calculation for each of the variables, a "golden data set" of reading variables was exported from the original EyeMap for a completed experiment. That experiment was then made available for testing purposes in EyeMap 2.0. Due to the large number of variables required, Dr Vorstius provided a large volume of guidance with understanding the variables; this shall be discussed more in Chapter 6. The data required to calculate each of the variables are already stored in the database. This data was retrieved from the database and then processed using the python programming language on the server side.

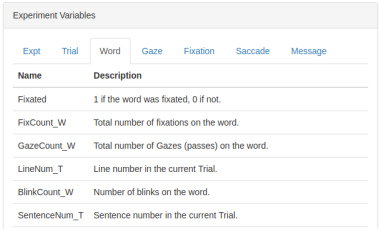
There are two different reports that can be generated from EyeMap 2.0. These are a word report and a fixation report. The word report focuses on the words and the order they appear while the fixation report focuses on the fixations and the order they are in. There are a number of options available to the user when generating reports, and these options are shown in Figure 5.8.



(a) Experiment level

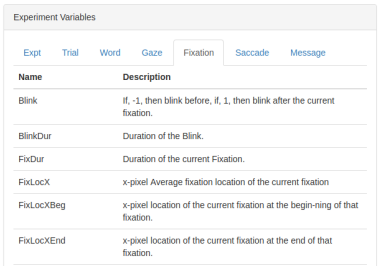


(b) Trial level

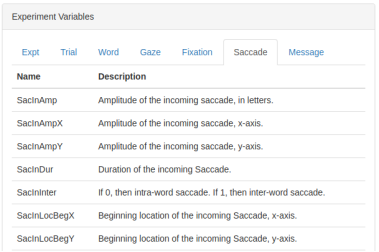


(c) Word level

(d) Gaze level



(e) Fixation level



(f) Saccade level

Figure 5.7: Display of Reading Variables and explanations.

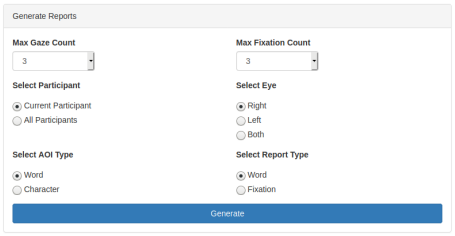


Figure 5.8: Options for reports to be exported.

The six different options are:

1. *Max Gaze Count* - This allows the user to select how many participant gazes they want to include in the report.

2. *Max Fixation Count* - This allows the user to select how many fixations on a word during a gaze is required for analysis.

3. *Select Participant* - This allows the user to select if they want the currently selected participant or all participants to be included in this analysis.

4. *Select Eye* - This allows the user to select if they want the right eye the left eye or the calculations for both eyes.

5. *Select AOI Type* - This allows the user the ability to chose word level AOI's or character level AOI's.

6. *Select Report Type* - This allows the user to choose between the word report or the fixation report.

Once the options have been selected and the user generates the selected report they receive a single Excel file with all of the calculated variables included. A sample of a section of a word report is shown in Figure 5.9, that shows a selection of calculated variables for some of the words in a trial of an experiment. These reports are then utilised by the users carrying out the experiment for further analysis.

DU	DV	DW	DX	DY	DZ	EA	EB
TrialNum	Word	WordCount_E	WordCount_L	WordCount_S	WordCount_T	WordLen	WordLen_punct
1	Since	273	9	17	43	5	5
1	candles	273	9	17	43	7	7
1	were	273	9	17	43	4	4
1	expensive	273	9	17	43	9	11
1	the	273	9	17	43	3	3
1	glowing	273	9	17	43	7	7
1	fire	273	9	17	43	4	4
1	in	273	9	17	43	2	2
1	the	273	9	17	43	3	3
1	fireplace	273	10	17	43	9	9
1	was	273	10	17	43	3	3
1	often	273	10	17	43	5	5
1	the	273	10	17	43	3	3
1	only	273	10	17	43	4	4
1	source	273	10	17	43	6	6
1	of	273	10	17	43	2	2
1	light	273	10	17	43	5	6
1	People	273	10	12	43	6	6
1	cooked	273	10	12	43	6	6
1	meals	273	10	12	43	5	5
1	in	273	10	12	43	2	2
1	the	273	10	12	43	3	3
1	fireplace	273	10	12	43	9	10
1	and	273	10	12	43	3	3

Figure 5.9: Sample of word report exported to an excel file.

5.12 ADDITIONAL WORK

During the course of redevelopment additional members were added to the development team, and they were Jeremy Yon and Niall Brennan. Both developers contributed different sections to EyeMap 2.0.

Jeremy's main contribution was relating to the experiment list that shows all of a users experiments, and then displays all of the participants within that experiment. This can be seen in Figure 5.5. He also enabled the user to delete archived experiments.

EyeMap 2.0 was developed exclusively for reading experiments, of plain text. Niall worked on allowing experiments of different styles of text to be imported into EyeMap 2.0 meaning, that a user was not restricted to reading experiments. His main contribution in the current version of EyeMap 2.0 is related to the character level AOIs and allowing the user to change between the character word level AOIs. Figure 5.10a shows the word level AOI's on an experiment and Figure 5.10b shows the character level AOI on an experiment.

(a) Word level AOI's. (b) Character level AOI's.

Figure 5.10: The different AOI's that can be generated.

Having the GitLab repository set up allowed both Niall and Jeremy to contribute easily to EyeMap 2.0. They were able to use the Gitflow process and create their own feature branches for development. Changes were then easily integrated into the main system.

5.13 SUMMARY

In this chapter, the development of EyeMap 2.0 was discussed in detail and showed how the challenge of recreating each function of EyeMap 1.0 was overcome. The chapter went through the RAD task list from implementing the Django architecture to discussing how EyeMap 2.0 was designed and how each of the features was developed. The chapter concluded by showing how the architecture of EyeMap 2.0 could easily have additional features added by utilising Gitflow and a repository on GitLab allowing other developer access to the code base. Chapter 6 will discuss details of how testing was conducted during the RAD process.

Part III

EVALUATION & FUTURE WORK

6

EVALUATION

This chapter will present the evaluation of the EyeMap 2.0 system. During the RAD process, the end user team members were exposed to the different modules of EyeMap 2.0. They used the Backlog system discussed in Section 4.6.3 to report errors and bugs within each of the modules. Section 6.1 will look at how Backlog was used to streamline the testing and evaluation of EyeMap 2.0 in more detail. This chapter will then revisit the software requirements provided in Section 4.4 to demonstrate the work completed in developing EyeMap 2.0. A usability survey was completed on EyeMap 2.0 by the end user team and by individuals not familiar with eye-tracking or EyeMap 2.0. The results of these surveys are presented in Section 6.3.

6.1 SYSTEM TESTING

Throughout the development of EyeMap 2.0 Backlog was utilised as an error logging and bug tracking software tool to report issues to the developer and track their progress for testing and evaluation without long email chains between the developer and end users. Backlog allowed users to open an issue/ticket concerning an issue with EyeMap 2.0 and a daily report of tickets was sent to each team member so they could be tracked. Tickets were created with a type, a subject, a priority, a status and assigned to someone on the team. Backlog would also assign a unique *Key* to each ticket that was created so that issues could be tracked between users. The

tickets could be set to one of five types by the user that opened the ticket. The types and what they represent are:

1. Critical - A critical ticket related to a major issue with the system and needed to be fixed immediately.
2. Task - A task was a minor issue and needed to be fixed.
3. Bug - A bug was an error in the system where the code base was not completing a task as expected.
4. Request - A request was a feature requested by the user that would enhance the system.
5. Other - The Other ticket was used to catch any other types of issues that may happen. Issues generally were captured by the first four types of issue and this was added as a type of a precautionary measure to capture any other issues for the system.

Each submitted ticket had a subject where the person who opened the ticket could provide more information about the issue, and also include screenshots if available. The ticket could then be given a priority of low, normal or high. A low priority ticket was not urgent and was used mostly with a *Request* or *Other* type ticket. Normal priority was used to report an issue or bug that was not *Critical* and was not required to be fixed immediately. It was mostly used with *Task* or *Bug* type tickets and high priority was used for *Critical* type tickets. However, any priority level could be used with any of the ticket types. To manage tickets they were also assigned a status; with the following possible levels:

1. Open - To show that a ticket was opened on the system.
2. In Progress - To show that a ticket was being investigated.
3. Resolved - To show that the issue had been fixed or completed and remained open for testing.

4. Closed - To show that the issue had been fixed and re-tested and was acceptably *Resolved*.

Figure 6.1 presents an example of three tickets that were opened in Backlog. Each ticket has a type, a key, a subject, a priority level, the team member that opened the ticket, the team member the ticket is currently assigned to and the status. During the RAD process, the end user team and developer were able to open tickets to easily communicate any issues or ideas that arose for the development of EyeMap 2.0.

Figure 6.1: List of Backlog tickets.

Figure 6.2 presents a sample *Bug* ticket submitted to Backlog, with an associated image, being assigned to the developer within Backlog to track issues. The assignee on a ticket could be changed as the ticket was being worked on. This allowed the developer and end user team to keep track of tickets they were working on.

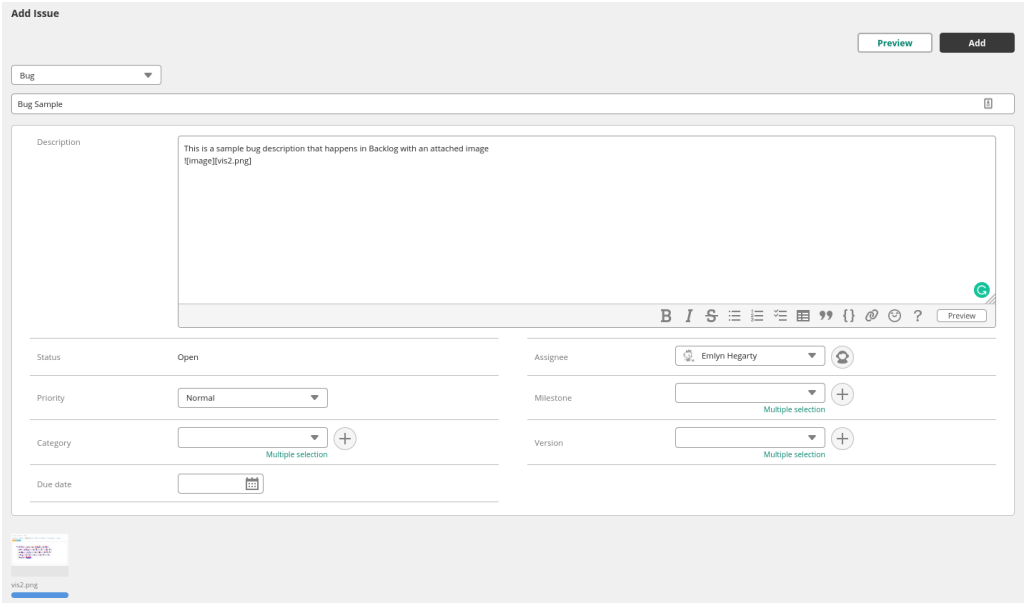


Figure 6.2: Sample Backlog *Bug* ticket being created.

Throughout the RAD process, the end user team conducted communications through the Backlog ticketing system. As the developer for EyeMap 2.0 progressed through the RAD task list the end user team became more involved. As a module or feature of EyeMap 2.0 was developed the developer carried out some basic testing to ensure the module worked as required. The end user team would then step in and further stress test the system with more data to verify the correct functionality of the module. When the user found an issue during testing they would open a new ticket to be resolved. In total there were forty-two tickets opened on Backlog for different issues within different modules. These can be broken down to twenty *Critical* tickets, eleven *Task* tickets, five *Bug* tickets, four *Request* tickets and two *Other* tickets. The task list as previously described in Section 5.2 was:

- Set up Core Architecture
- Initialise GitLab Repository
- Design and plan the layout of EyeMap 2.0
- Create the Database for EyeMap 2.0
- Set up Templates
- Create Web Pages
- Data Loader
- Data Visualisation & Editor
- Data Exporter

The tasks to *Setup Core Architecture* and *Initialise GitLab Repository* to generate the project code base and did not require the end user team to be involved as they were completed by the developer. The *Design and plan the layout of EyeMap 2.0* task was discussed at the requirements meeting

held in Wuppertal, Germany. The end user team did not have any issues with the initial layout of the architecture of EyeMap 2.0. The consensus was that they were happy to allow the developer to work on this task and layout of the new system as required for a web application. As the project progressed the team had more input about the design through the Backlog ticketing system.

The team was also satisfied for the developer to *Create the Database for EyeMap 2.0* as this was designed to satisfy the different requirements of the system. Mr Hynes created a ticket that helped to improve the database design. Figure 6.3 shows the ticket that Mr Hynes opened. It is a *Task* type ticket and in it Mr Hynes suggests the redefining of the experiment model so that the experiment is a table in the database on its own and the subject should be a relation to the experiment, i.e. a one-to-many relationship between the experiments and participants table. This *Task* greatly improved the database structure and shaped the database that EyeMap 2.0 now utilises.

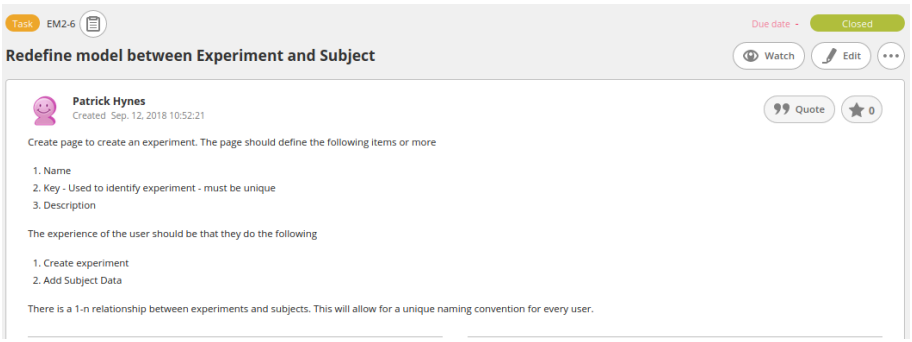


Figure 6.3: Ticket opened by Mr Hynes in relation to the database design.

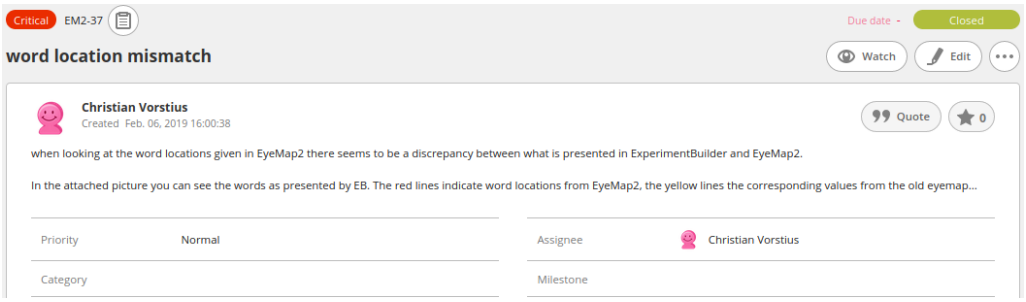
The tasks to *Setup Templates* and *Create Web Pages* solely affected the developer as this implemented the layout and structure of EyeMap 2.0 that was previously approved by the end user team. The end user team had most input into the modules that were created for the *Data Loader*, *Data Visualisation & Editor* and *Data Exporter*. The tickets that were opened

by the end user team will be discussed with the module it was opened against.

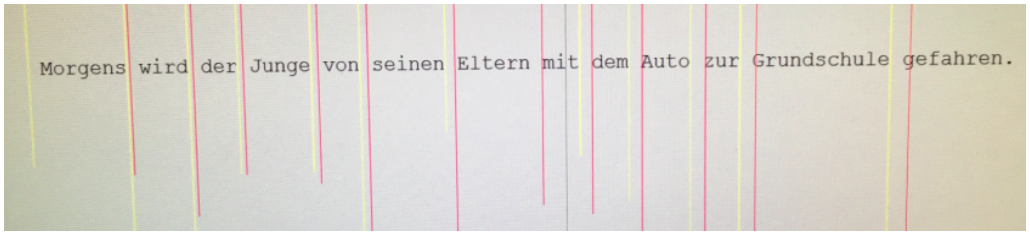
As discussed in Section 5.9 the *Data Loader* is responsible for the uploading of all data files for experiments. For this module, Mr Hynes suggested to simplify the upload wizard and handle any abbreviations or other data that could be handled in the background that the user would find distracting. This feedback greatly simplified the upload wizard. Ms Fan had conducted a large experiment with over 100 participants in multiple experiments. She thoroughly tested the upload wizard. Some of the data files that were presented to EyeMap 2.0 had elements in different locations within the file compared to what was expected. The fix for this issue allowed the upload wizard to be more flexible in how it processes the data.

The *Data Visualisation & Editor* was extensively tested by Dr Vorstius, Ms Fan and Mr Hynes. The first major issue was with the word location on the visualisation page of EyeMap 2.0. To test this Dr Vorstius got a visualisation from the original EyeMap and then loaded the same experiment into EyeMap 2.0. He found that the location of the words was misaligned on the first word and this misalignment increased the further across the line of text the user went. Figure 6.4 presents the description and attached image of the ticket that Dr Vorstius opened. In the image the yellow lines represent where the text was located in the original EyeMap and the red lines represent where they were located in EyeMap 2.0. This issue was *Critical* as this misalignment would then create wrong variable calculations in the *Data Exporter*. The issue was caused by a rounding error in the AOI calculation with opentype.js and was easily rectified. As shown in the top-right of 6.4a the ticket was closed once the Dr Vorstius had verified that the issue was resolved.

Ms Fan opened a ticket relating to a bug in the system where some of the fixations were missing from the visualisations. Upon further investigation, it was confirmed to only happen with participant data that had only left



(a) Description provided in the ticket about the misalignment of text



(b) Image provided in the ticket to show the misalignment of text

Figure 6.4: Misalignment of text ticket, with image, opened on Backlog

or right eye data associated with it. This issue was caused during the data upload process from the way the data was being extracted from the participant XML file. This was a *Critical* issue and required a refactor of the code involved in the data upload module. The ticket was closed once Ms Fan had verified the issue was resolved.

Mr Hynes opened a ticket relating to how the fixation data was being saved after editing. If multiple edits were made with multiple saves the database would have a new participant with the edited fixation data, but when the user left the visualisation page and returned the data would show the first "new participants" data and any saves after that were not displaying. This bug was created when the user pressed the save button on the visualisations page. The solution to this required some refactoring for the save edits function. After the fix was in place and this was verified by Mr Hynes the ticket was closed.

The modules to calculate the reading variables and export the data in the *Data Exporter* were considered vital for EyeMap 2.0. For these modules, Dr Vorstius travelled to Maynooth University for two days in March 2018.

Dr Vorstius is an expert in these variables as he uses them regularly within his own research and able to advise the author on the reading variables. During this visit, a large amount of the code to perform the variable calculations was completed. As described in Section 5.11 Dr Vorstius generated a "Golden Data Set" from an experiment that was previously analysed using the original EyeMap. To test the variable calculations in EyeMap 2.0 the same experiment was loaded into the system and the variables were calculated and exported. Dr Vorstius then compared the results of each data set by hand so that each variable was verified as calculating correctly. When the module to generate the word and fixation reports for EyeMap 2.0 were completed the end users tested the system. The issues that were raised here were related to variables being calculated incorrectly. Ms Fan also scrutinised the variables on a number of her experiments to ensure that they were correct.

Throughout the development process, if any tickets were opened through Backlog, the developer would investigate the issue in the ticket and as it was resolved it would be sent back to the team member that created it for further testing before the ticket was closed. As testing on the module was completed and then approved by the end user team the module was merged into the master branch on GitLab for release on the live web application.

6.2 REQUIREMENTS CHECKLIST

There were a number of requirements and fixes from the original EyeMap for EyeMap 2.0; listed in Section 4.4. This section will review the requirements, to analyse what was completed and what has yet to be completed. Table 6.1 presents a break down of the completion level of requirements and fixes for EyeMap 2.0.

Table 6.1 shows a large number of requirements were met for the EyeMap 2.0 system. These requirements are improvements on the original EyeMap. The first requirement is the ability to upload multiple participants data for an experiment to the system at once. When the data is upload and stored in the database it is then available to the end user once they log in to the EyeMap 2.0 system. This provides end users with easy access to their experiments and participant data files. When the experiment trial data is uploaded the system can manage a wide range of proportional and monospaced spaced fonts. The ability to add a unique separator to the text.html file provides the ability to easily segment any language as required.

EyeMap 2.0 has provided functionality to batch export variable calculations for participants in an experiment. This is a big improvement on the original EyeMap. For the original system users were required to upload a single participants data and then export the variable calculations for that participant and then repeat this for each participant in the experiment. With EyeMap 2.0 the user can upload all participants for an experiment at once and then complete the variable calculations for all participants in one report for the variable calculations.

As presented in Table 6.1 three requirements were partially complete. These three requirements are *Visualisations*, *Collaboration* and *Decimal numbers separator*. The *Visualisations* are marked as partially complete as in the original EyeMap there were visualisations for fixations, saccades and scan paths while in EyeMap 2.0 only visualisations for fixations was fully completed. The structure was created in the code base with Konva.js to implement all of the visualisations but due to the development and testing required for *Variables* and *Batch Data Export* the time to complete the requirement was not available. Additionally, the *Variables* and *Batch Data Export* requirements were deemed to have a higher need and priority level from the end user team, as given in the requirements. For the *Collaboration*

requirement, the database has been set up to implement the collaboration feature for experiments but as the need level was *Nice* and the priority level was *Low* this feature was not fully implemented. The *Decimal numbers separator* was also not implemented as it had received a *Low* priority level and due to time constraints, other requirements were considered by the end users to be more valuable to the system. These features can be implemented in the future in the EyeMap 2.0 system.

Table 6.1: Requirements and fixes completion level for EyeMap 2.0

No.	**Requirement**	**Complete**	**Incomplete**	**Partial**
1	Loading Multiple Participants	X		
2	Data Storage	X		
3	Login Functionality	X		
4	Database	X		
5	Languages and Fonts	X		
6	Visualisations			X
7	Variables	X		
8	Collaboration			X
9	Batch Export	X		
10	New Variable Calculations	X		
11	Edited fixations changing the fixation order in the viewer	X		
12	Segmentation of Chinese words	X		
13	File naming conventions	X		
14	Decimal numbers separator			X

Ms Fan has conducted a large study using EyeMap 2.0. To complete this she has worked with all modules within the EyeMap 2.0 system. Ms Fan

used EyeMap 2.0 to analyse all of her participants in all of the experiments with both character and word level AOIs for both the fixation and word reports and currently has a conference paper under review.

6.3 END USER EVALUATION

To further verify EyeMap 2.0 a usability Survey from usability.gov called a System Usability Scale (SUS) was conducted. SUS was created by John Brooke in 1986 as a reliable tool for measuring the usability of everything from hardware to software applications [46]. It consists of a ten-question survey, where respondents are given five possible responses from "Strongly agree" to "Strongly disagree" for each question. The ten questions as provided by usability.gov are:

- I think that I would like to use this system frequently.
- I found the system unnecessarily complex.
- I thought the system was easy to use.
- I think that I would need the support of a technical person to be able to use this system.
- I found the various functions in this system were well integrated.
- I thought there was too much inconsistency in this system.
- I would imagine that most people would learn to use this system very quickly.
- I found the system very cumbersome to use.
- I felt very confident using the system.
- I needed to learn a lot of things before I could get going with this system.

SUS provides the tester with five scores based on the survey. The scores and what they represent are:

1. SUS score - This score represents how usable and learnable the application is. The SUS score is given in the range 0 - 100 and it is a percentile, not a percentage. The average SUS score is 68. A score of 75 is better than about 73% of SUS scores. Meanwhile, a score of 52 is worse than around 85% of the scores.

2. Learnability - This score represents whether a user would need a technical person's support to be capable of using the application. The score also represents whether a user would need to learn a large amount of information before starting to use the application.

3. Usability - This score refers to how easy it is for a user to make use of the application and how efficiently can users use it to accomplish their specific objectives?

4. Standard deviation - The standard deviation shows how much variation there is in your score from the average SUS score. If the standard deviation figure is low, this means that the data are close to the average. Meanwhile, if your standard deviation is high, this means that the values are instead spread out over a much wider range.

5. Cronbach's alpha - This score is a measure of internal consistency or how closely related a set of items are as a group. A reliability coefficient of .70 or higher is considered "acceptable" in social science research situations.

For EyeMap 2.0 there were two separate SUS surveys conducted. The first was provided to the end user team who worked directly in the design and development of EyeMap 2.0 there were 5 participants in this study. The second SUS survey was given to participants who were not

familiar with eye-tracking, eye-tracking data analysis or the original EyeMap. These participants are from various backgrounds and would have varying levels of technological skills. The second group were provided with a sample data set of an eye-tracking study and a set of instructions that requested they do various tasks on EyeMap 2.0 before completing the survey. The instructions they were provided are:

- Register for EyeMap 2.0
- Create a new Experiment
    1. Step 1: You can provide any name and any description, these are just identifiers
    2. Step 2: For the Experiment Text File, please use -> text.html
    3. Step 3: For the Experiment Config File, please use -> config.xml
    4. Step 4: You can upload a single participant or multiple participants, The participant files are: [00smp001.xml, 01smp001.xml, 02smp001.xml, 03smp001.xml, 04smp001.xml]
- Select an experiment
- Select a participant
- Try the different functionalities on the participant visualise page
- Open the Fixations tab
    1. Step 1: Edit fixations (red and blue dots) in the vertical direction using your keypad arrows
    2. Step 2: Save your edits
- Using EyeMap 2.0 Save an image to your desktop
    1. View the image that was downloaded to ensure that an image was provided
- On the Analysis page generate some reports following these steps

1. Step 1: Generate a Word report with a Max Gaze Count of 2 and a Max Fixation Count of 4 for the current participant with word AOI type
2. Step 2: View the report that was downloaded to ensure that it opens correctly
3. Step 3: Generate a Fixation report with a Max Gaze Count of 3 and a Max Fixation Count of 5 for all participants for the Right eye only
4. Step 4: View the report that was downloaded to ensure that it opens correctly

Table 6.2: Results of SUS Survey.

Measure	**Survey Group One**	**Survey Group Two**
SUS score	71.3	70.3
Learnability	75	65.8
Usability	72.5	71.4
Standard Deviation	25	14.5
Cronbach's alplha	0.99	0.833
Participant Numbers	5	19

Table 6.2 shows the results of both surveys. Survey group one relates to the end user team that are experts in eye-tracking and reading research and assisted in the development of EyeMap 2.0 and consisted of 5 participants. The individual scores for participants for this survey group are presented in 6.5a. Survey group two consisted of 19 participants who had little to no prior knowledge of eye-tracking and their scores are presented in 6.5b. The *Cronbach's alpha* score for both surveys was above 0.7 so the results were "acceptable" for both surveys. The *Standard Deviation* for survey group one was higher than for survey group two; this could be a result of

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	SUS	Usability	Learnability
3	1	5	2	4	1	4	1	5	1	87.5	87.5	87.5
3	1	4	1	4	1	4	1	4			81.3	
4	1	5	2	4	1	5	1	5	1	92.5	93.8	87.5
3	3	4	2	3	1	3	2	3	1	67.5	62.5	87.5
2	3	3	3	2	4	2	3	3	4	37.5	37.5	37.5

(a) Individual results from survey group one

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	SUS	Usability	Learnability
3	2	4	3	4	2	4	1	4	3	Good	70.0	75.0	50.0
4	2	4	4	4	2	5	3	4	2	Good	70.0	75.0	50.0
3	2	4	2	4	2	3	2	4	2	Good	70.0	68.8	75.0
4	2	4	2	3	3	5	2	4	1	Good	75.0	71.9	87.5
3	2	5	2	4	1	4	1	4	2	Excellent	80.0	81.3	75.0
3	2	4	2	4	1	5	2	3	3	Good	72.5	75.0	62.5
4	1	5	2	5	2	4	1	4	1	Excellent	87.5	87.5	87.5
5	3	3	1	5	1	4	2	3	1	Excellent	80.0	75.0	100.0
5	2	4	2	5	1	5	1	4	1	Excellent	90.0	90.6	87.5
3	4	2	2	5	1	4	4	3	4	Excellent	55.0	56.3	50.0
3	1	3	1	4	1	4	1	4	1	Excellent	82.5	78.1	100.0
4	3	3	3	4	1	4	3	2	4	Good	57.5	62.5	37.5
4	2	3	1	4	3	4	3	1	3	Good	60.0	56.3	75.0
5	1	4	2	4	2	4	2	5	1	Excellent	85.0	84.4	87.5
5	1	4	2	4	1	4	1	4	3	Best Imaginable	82.5	87.5	62.5
3	3	3	2	4	2	4	2	4	4	Good	62.5	65.6	50.0
3	3	3	4	4	2	5	2	2	3	Good	57.5	62.5	37.5
3	4	2	5	3	3	4	4	1	5	Poor	30.0	37.5	0.0
3	2	4	3	4	2	4	3	3	1	Good	67.5	65.6	75.0

(b) Individual results from survey group two

Figure 6.5: Individual results for the SUS for both survey groups.

the smaller numbers that participated in the survey. Both surveys scored above average in the *SUS score*, meaning EyeMap 2.0 is considered usable and learnable by all who reviewed the system. It was expected that the *Lernability* score would be higher for survey group one as these are the users with prior knowledge in the field that EyeMap 2.0 was created for and this was observed with a large variation existing. Both surveys also returned similar *Usability* scores meaning that both sets of users felt they could easily complete required tasks with EyeMap 2.0.

6.4 SUMMARY

Using the RAD process in conjunction with Backlog allowed the end user team to actively participate in the software development process, by providing the end users with the opportunity to test and report back on each of the features that were being developed. This allowed the developer to harness the end users experience to test each feature as it was developed. This experience was beneficial as the end user team were able to test scenarios and issues that were not considered by the developer. It also meant that only features that had been fully tested and verified were able to be added to the live EyeMap 2.0 system.

To verify the usability of EyeMap 2.0 a survey was conducted on two groups. The first group consisted of members of the end user team and the second group consisted of members who had no prior knowledge of eye-tracking data analysis. As expected the learnability of EyeMap 2.0 was higher for the experts. However, both survey groups scored the usability aspect above the SUS score average. The results of this survey showed that EyeMap 2.0 is considered user-friendly and opens up eye-tracking data analysis to a wider user base.

7

CONCLUSIONS & FUTURE WORK

This chapter will present the conclusions and future work components of this project. It will examine the research questions and discuss these in detail before suggesting possible future work that could be carried out with the project to further enhance EyeMap 2.0.

7.1 CONCLUSIONS

The project had a clear goal when it began. This was to develop EyeMap 2.0 and improve on the original EyeMap. There were three research questions that the project hoped to answer while carrying out its goal. They were:

1. What is eye-tracking and why is it important?
2. What software models, tools, and languages are appropriate for a project of this nature?
3. What is EyeMap and how can it be improved?

To complete EyeMap 2.0 to a satisfactory level it was fundamental to understand eye-tracking and how it works so that as it is a specialist field with specific terminology. Eye-tracking was introduced in Section 2.2, and in this section, the origins of eye-tracking technology was shown to be from 1937 when Buswell recorded eye movements from light reflected off the eye and many eye-tracking companies still use this method today to record eye movements. It was also shown that fixation and saccades are

two extremely important values when it comes to the analysis of eye movements data. Understanding these two aspects of eye movements greatly assisted with many aspects of EyeMap 2.0 from processing the data from the *Data Loader* to exporting the variable calculations in the *Data Exporter*.

To answer research question two required the understanding of the software development cycle and the different stages, from planning to testing & integration. To complete the software development cycle there are many models of development. After comparing four different Agile development models, it was decided that the RAD process best suited the development of EyeMap 2.0. It is suited to small teams and the task list is generated by breaking down items into modules that could provide a clear development path for EyeMap 2.0.

To develop EyeMap 2.0, it was important to first understand the original EyeMap through using a working version and insights from the users working with the development team. EyeMap has several important modules that make up its architecture and would prove challenging to recreate from scratch. These are (i) Data Parser, (ii) Data Loader (iii) Data Editor, (iv) Data Visualisation, (v) Reading Variables and (vi) Data Exporter. The data for EyeMap was prepared in the Data Parser and passed through each of the modules until it was exported in the Data Exporter for further analysis. Developing all of these modules from EyeMap greatly benefited EyeMap 2.0. The main improvement EyeMap 2.0 made on the original EyeMap is the batch processing of files, this was also a big challenge to solve as EyeMap 2.0 moved to a web application. The batch processing of files means the user does not have to go through the process of loading single participants for single experiments. They upload the experiment and all participant data files at once to EyeMap 2.0. The data for each participant is easily accessible to the user. This also enhances the *Data Exporter*, as all the data for each participant is stored in the database a batch word or fixa-

tion report is enabled in EyeMap 2.0 so the user does not have to generate a report for every participant in the experiment.

Two tools in particular proved to be invaluable contributions to the project. The first was the creation of a git repository for version control that utilised Gitflow. This combined with the RAD process easily allowed the separation of tasks under development and allowed for easy testing and quick bug fixes. The second tool that was a major contributing factor was Backlog. These tools combined proved to be invaluable and allowed for better communication and understanding between the users and developer of EyeMap 2.0, with the RAD process.

The RAD process worked for this project. This is a valuable contribution of this thesis, as it can be difficult for a small development team to know which of the many agile models should be used or how effective they can be. The end user team also provided invaluable information for the requirements and having a team meant any queries about features or functionality were answered promptly. Having the end user team test the features of EyeMap 2.0 as they were developed solved many issues that may have been otherwise missed. The downside to the team was that different team members wanted different functionality at the same time. It was then difficult on the single developer to manage expectations. Given the downside the project was enhanced as a result of the user participation and it can be clearly seen that they would be a major contribution factor to any development team for any project. The only recommendation for a small project like this in the future would be to clearly identify the RAD task list at the beginning of the project and clearly set milestones for deliverables.

7.2 FUTURE WORK

When EyeMap 2.0 was developed it was done so with a future proofing approach. The GitLab repository has made the code base easily accessible and portable. As demonstrated in Section 5.12 the repository allows additional developers to join and contribute. EyeMap 2.0 has endeavoured to continue the legacy of the original EyeMap, however, there is still work that could be completed to improve the system even further. As seen in Section 6.2 EyeMap 2.0 did not complete all features of the original EyeMap.

EyeMap 2.0 had partial work completed for the visualisations of the original EyeMap and these are presented here as part of the future work and collaboration features. The code base had allowed for the completion of both of these features. For the visualisations, the fixations have already been visualised and are interactive. The next step would be to implement the visualisations for the saccades, scan paths and heat maps. For the collaboration the database has been designed to implement this feature, allowing users to collaborate their work without having to download and re-upload the data.

The AOI's are automatically generated in EyeMap 2.0. However, there are times when a user might wish to define their own AOI's or combine AOI's. The researcher might not be interested in word or character level and would instead prefer to do line level AOI's. German nouns always have gender associated with them. Instead of the experiment analysing the data for "die" and "sonne" separately the researcher may wish to combine these into one AOI for "die sonne". This would then provide users of EyeMap 2.0 with more flexibility in the types of analysis they run using the software.

Currently, researchers use the calculated variables in the reports generated by the Data Exporter by conducting statistical analysis on them in R,

a statistical programming language. There are a number of statistical libraries for Python, such as Pandas, Numpy, Scipy and mathplotlib. These libraries can carry out the same statistical analysis as R and as the core architecture is written in Python the analysis could be carried out within EyeMap 2.0 thus saving the researcher going elsewhere to complete the data analysis and also reduce the chances of human error being introduced.

BIBLIOGRAPHY

[1] *BackLog from Nulab*. Accessed: 2019-10-20. URL: https://backlog.com.

[2] David A Balota and Keith Rayner. "Word recognition processes in foveal and parafoveal vision: The range of influence of lexical variables". In: *Basic processes in reading: Visual word recognition* (1991), pp. 198–232.

[3] Richard Bates, Howell Istance, and Oleg Spakov. *Requirements for the common format of eye movement data*. Tech. rep. Technical report, Communication by Gaze Interaction (COGAIN): Deliverable 2 ..., 2005.

[4] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. "Manifesto for agile software development". In: (2001).

[5] *Bootstrap 3*. Accessed: 2017-11-13. URL: https://getbootstrap.com/docs/3.3/.

[6] Guy T Buswell. "How adults read.(Supplementary Educational Monographs)". In: *Chicago: University of Chicago* (1937).

[7] Barbara Cassin, Sheila Solomon, and Melvin L Rubin. *Dictionary of eye terminology*. Triad Pub. Co., 1984.

[8] Frederik De Bleser. *opentype.js*. Accessed: 2018-1-17. URL: https://opentype.js.org.

[9] *Django File Structure*. Accessed: 2020-2-2. URL: https://smartlazycoding.com/django-tutorial/add-important-files-and-folders-to-initial-django-project-structure.

[10] *Django*. Accessed: 2018-04-20. URL: https://www.djangoproject.com/foundation/.

[11] Venkatreddy Dwarampudi, Shahbaz Singh Dhillon, Jivitesh Shah, Nikhil Joseph Sebastian, and Nitin Kanigicharla. "Comparative study of the Pros and Cons of Programming languages Java, Scala, C++, Haskell, VB. NET, AspectJ, Perl, Ruby, PHP & Scheme-a Team 11 COMP6411-S10 Term Report". In: *arXiv preprint arXiv:1008.3431* (2010).

[12] *Essentials: Software Development Life Cycle*. Accessed: 2019-01-17. URL: https://www.intellectsoft.net/blog/essentials-software-development-life-cycle/.

[13] *EyeMap - Eye Movement Data Analyzer*. Accessed: 2019-02-17. URL: https://sourceforge.net/projects/openeyemap/.

[14] Adila Firdaus, Imran Ghani, and Nor Izzaty Mohd Yasin. "Developing secure websites using feature driven development (FDD): a case study". In: *Journal of Clean Energy Technologies* 1.4 (2013), pp. 322–326.

[15] *Font Metrics*. Accessed: 2018-04-20. URL: https://developer.apple.com/library/archive/documentation/TextFonts/Conceptual/CocoaTextArchitecture/FontHandling/FontHandling.html.

[16] *Git Book, git flow*. Accessed: 2020-2-2. URL: https://gitbook.tw/chapters/gitflow/why-need-git-flow.html.

[17] *Gitflow Tutorial*. Accessed: 2020-2-2. URL: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow.

[18] Judith Gregory. "Scandinavian approaches to participatory design". In: *International Journal of Engineering Education* 19.1 (2003), pp. 62–74.

[19] Juergen Wolff von Gudenberg. "Java for scientific computing, pros and cons". In: *Journal of Universal Computer Science* 4.1 (1998), pp. 11–15.

[20] Kim Halskov and Nicolai Brodersen Hansen. "The diversity of participatory design research practice at PDC 2002–2012". In: *International Journal of Human-Computer Studies* 74 (2015), pp. 81–92.

[21] T Halverson and A Hornof. "VizFix software requirements specification". In: *Eugene: University of Oregon, Computer and Information Science. Retrieved August* 3 (2002), p. 2011.

[22] Emlyn Hegarty-Kelly, Susan Bergin, and Aidan Mooney. "Using focused attention to improve programming comprehension for novice programmers." In: *Third International Workshop on Eye Movements in Programming*. 2015, pp. 8–9.

[23] E Hering. "Über Muskelgeräusche des Auges. Sitzungsber. d". In: *Wien. Akad. d. Wiss* 79.3 (1879), pp. 137–154.

[24] Jim Highsmith and Alistair Cockburn. "Agile software development: The business of innovation". In: *Computer* 34.9 (2001), pp. 120–127.

[25] David E Irwin. "Lexical processing during saccadic eye movements". In: *Cognitive Psychology* 36.1 (1998), pp. 1–27.

[26] Emile Javal. "Essai sur la physiologie de la lecture". In: *Annales d'Ocilistique* 80 (1878), pp. 61–73.

[27] *JetBrains*. Accessed: 2019-09-20. URL: `https://www.jetbrains.com/pycharm/`.

[28] Robert Gabriel Lupu and Florina Ungureanu. "A survey of eye tracking methods and applications". In: *Buletinul Institutului Politehnic din Iasi, Automatic Control and Computer Science Section* 3 (2013), pp. 72–86.

[29] Hugh Mackay, Chris Carne, Paul Beynon-Davies, and Doug Tudhope. "Reconfiguring the user: using rapid application development". In: *Social studies of science* 30.5 (2000), pp. 737–757.

[30] James Martin. *Rapid application development*. Macmillan Publishing Co., Inc., 1991.

[31] Ethel Matin. "Saccadic suppression: a review and an analysis." In: *Psychological bulletin* 81.12 (1974), p. 899.

[32] Peter Naur. "Software Engineering-Report on a Conference Sponsored by the NATO Science Committee Garimisch, Germany". In: *http://homepages. cs. ncl. ac. uk/brian. randell/NATO/nato1968. PDF* (1968).

[33] *Participatory Design Conference*. Accessed: 2018-04-20. URL: https://pdc2018.org.

[34] *Participatory Design*. Accessed: 2019-08-24. URL: https://www.encyclopedia.com/science/encyclopedias-almanacs-transcripts-and-maps/participatory-design.

[35] Mark C Paulk. "Extreme programming from a CMM perspective". In: *IEEE software* 18.6 (2001), pp. 19–26.

[36] Lutz Prechelt. "An empirical comparison of seven programming languages". In: *Computer* 33.10 (2000), pp. 23–29.

[37] *Qt Framework*. Accessed: 2020-01-17. URL: https://www.qt.io/.

[38] Keith Rayner. "Eye movements in reading and information processing: 20 years of research." In: *Psychological bulletin* 124.3 (1998), p. 372.

[39] Keith Rayner, Elizabeth R Schotter, Michael EJ Masson, Mary C Potter, and Rebecca Treiman. "So much to read, so little time: How do we read, and can speed reading help?" In: *Psychological Science in the Public Interest* 17.1 (2016), pp. 4–34.

[40] Linda Rising and Norman S Janoff. "The Scrum software development process for small teams". In: *IEEE software* 17.4 (2000), pp. 26–32.

[41] *SMI Vision Website*. Accessed: 2019-02-17. URL: http://www.smivision.com.

[42] *Software Development Methodology – Feature Driven Development (FDD)*. Accessed: 2019-08-22. URL: https://www.tuannguyen.tech/2019/07/22/software-development-methodology-feature-driven-development-fdd/.

[43] Clay Spinuzzi. "The methodology of participatory design". In: *Technical communication* 52.2 (2005), pp. 163–174.

[44] *Sr Research Website*. Accessed: 2019-02-17. URL: https://www.sr-research.com.

[45] Ilene Strizver. *Single or Double Spaces Between Sentences?* Accessed: 2018-02-17. URL: https://creativepro.com/single-or-double-spaces-between-sentences/.

[46] *System Usability Scale*. Accessed: 2020-1-20. URL: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html.

[47] Siliang Tang, Ronan G Reilly, and Christian Vorstius. "EyeMap: a software system for visualizing and analyzing eye movement data in reading". In: *Behavior research methods* 44.2 (2012), pp. 420–438.

[48] TatvaSoft. *Top 12 Software Development Methodologies & its Advantages / Disadvantages*. Accessed: 2018-03-20. URL: https://www.tatvasoft.com/blog/top-12-software-development-methodologies-and-its-advantages-disadvantages/.

[49] intelegain Technologies. *Scrum Development Model*. Accessed: 2019-08-22. URL: https://www.intelegain.com/scrum/.

[50] *Tobii Website*. Accessed: 2019-02-17. URL: https://www.tobiipro.com/.

[51] *Top 12 Eye Tracking Hardware Companies (Ranked)*. Accessed: 2019-02-17. URL: https://imotions.com/blog/top-eye-tracking-hardware-companies/.

[52] iMotions. *Eye tracking - The complete pocket guide*. https://imotions.com/, 2012.

[53] *konva.js*. Accessed: 2018-4-20. URL: https://konvajs.org.

A

EYEMAP 2.0 WEB APPLICATION

A.1 EYEMAP 2.0 WEB APPLICATION

EyeMap 2.0 can be viewed at: https://eyemap2.cs.nuim.ie/EyeMap2/

B

COMPLETE LIST OF CALCULATED VARIABLES

B.1 COMPLETE LIST OF CALCULATED VARIABLES

This Appendix contains a full list of variables discussed in Section 5.11

Id	Name	Level	Description	word_rep	fix_rep
1	EXP	Expt	Exptal name abbreviation, taken from 3rd-5th digit of filename.	Yes	Yes
2	List	Expt	optional, taken from 6th digit of filename.	Yes	Yes
3	Subject	Expt	Subject code, taken from first two digits of filename.	Yes	Yes
4	Var1	Expt	optional, taken from 7th digit of filename.	Yes	Yes
5	Var2	Expt	optional, taken from 8th digit of filename.	Yes	Yes
6	LineCount_T	Trial	Total number of lines for the Trial.	Yes	Yes
7	SentenceCount_T	Trial	Total number of sentences for the Trial.	Yes	Yes
8	TrialNum	Trial	Trial Number.	Yes	Yes
9	TrialProperty	Trial	Properties defined in the trial.csv file.	Yes	Yes
10	WordCount_T	Trial	Total number of words for the Trial.	Yes	Yes
11	Fixated	Word	1 if the word was fixated, 0 if not.	Yes	Yes
12	FixCount_W	Word	Total number of fixations on the word.	Yes	Yes
13	GazeCount_W	Word	Total number of Gazes (passes) on the word.	Yes	Yes
14	LineNum_T	Word	Line number in the current Trial.	Yes	Yes
15	BlinkCount_W	Word	Number of blinks on the word.	Yes	Yes
16	SentenceNum_T	Word	Sentence number in the current Trial.	Yes	Yes
17	TVDur_sac	Word	Total Viewing Time (TVD), or the sum of all fixations on the word + Saccades.	Yes	Yes
18	TVDur	Word	Total Viewing Time (TVD), or the sum of all fixations on the word.	Yes	Yes
19	Word	Word	Word, without punctuation.	Yes	Yes
20	WordBlinkDur	Word	Duration of the blinks in the word.	Yes	Yes
21	WordCount_L	Word	Total number of words on the Line.	Yes	Yes
22	WordCount_S	Word	Total number of words in the Sentence.	Yes	Yes
23	WordLen_punct	Word	Word Length, in letter, including punctuation.	Yes	Yes
24	WordLen	Word	Word length, in letters.	Yes	Yes
25	WordLocX	Word	x-pixel location of the word (upper left corner).	Yes	Yes
26	WordLocY	Word	y-pixel location of the word (upper left corner).	Yes	Yes
27	WordNum_E	Word	Word number for the Expt.	Yes	Yes
28	WordNum_L	Word	Word number on the Line.	Yes	Yes
29	WordNum_S	Word	Word number in the Sentence.	Yes	Yes
30	WordNum_T	Word	Word number for the Trial.	Yes	Yes
31	WordProperty	Word	Properties defined in the word.csv file.	Yes	Yes
32	FixCount_G	Gaze	Total number of fixation for the current Gaze (Pass).	No	Yes
33	GazeBlinkDur	Gaze	Duration of the blinks in the gaze.	No	Yes
34	GazeDur_sac	Gaze	Gaze duration (GD) of the current Gaze (Pass), in-cluding internal saccades, plus outgoing.	No	Yes
35	GazeDur	Gaze	Gaze duration (GD) of the current Gaze (Pass).	No	Yes
36	GazeNum_W	Gaze	Number of the current Gaze (Pass) on the word.	No	Yes
37	GazePupil	Gaze	Mean Pupil diameter for the entire gaze.	No	Yes
38	BlinkCount_G	Gaze	Number of blinks on the gaze.	No	Yes
39	Refixated	Gaze	1, if FixNum_G	No	Yes
40	RefixCount	Gaze	Total number of refixations on the word in the current gaze	No	Yes

41	Blink	Fixation	If, -1, then blink before, if, 1, then blink after the current fixation.	No	Yes
42	BlinkDur	Fixation	Duration of the Blink.	No	Yes
43	FixDur	Fixation	Duration of the current Fixation.	No	Yes
44	FixLocX	Fixation	x-pixel Average fixation location of the current fixation	No	Yes
45	FixLocXBeg	Fixation	x-pixel location of the current fixation at the begin-ning of that fixation.	No	Yes
46	FixLocXEnd	Fixation	x-pixel location of the current fixation at the end of that fixation.	No	Yes
47	FixLocY	Fixation	y-pixel Average fixation location of the current fixation.	No	Yes
48	FixLocYBeg	Fixation	y-pixel location of the current fixation at the beginning of that fixation.	No	Yes
49	FixLocYEnd	Fixation	y-pixel location of the current fixation at the end of that fixation.	No	Yes
50	FixNum_E	Fixation	Fixation number, in the Expt.	No	Yes
51	FixNum_G	Fixation	Number of the current fixation in the current Gaze (Pass).	No	Yes
52	FixNum_S	Fixation	Fixation number, in the Sentence.	No	Yes
53	FixNum_T	Fixation	Fixation number, for the Trial.	No	Yes
54	FixNum_W	Fixation	Number of the current fixation on the words.	No	Yes
55	FixPupil	Fixation	Mean Pupil diameter for the Fixation.	No	Yes
56	FixStartTime	Fixation	Time stamp of the start of the current Fixation (ms).	No	Yes
57	LandPos_NP	Fixation	(monospaced) Landing position in the word, described in letter units (space before word is 0).	No	Yes
58	LandPos	Fixation	Landing position in the word, described in letter units (space before word is 0).	No	Yes
59	LandPosDec_NP	Fixation	(monospaced) Landing position in the word, described in letter units including decimal places (space before word is 0).	No	Yes
60	LandPosDec	Fixation	Landing position in the word, described in letter units including decimal places.	No	Yes
61	LaunchDistBeg_NP	Fixation	(monospaced) Launch distance from the beginning of the current word.	No	Yes
62	LaunchDistBeg	Fixation	Launch distance from the beginning of the current word starting at letter zero.	No	Yes
63	LaunchDistBegDec_NP	Fixation	(monospaced) Launch distance from the beginning of the current word including decimal places.	No	Yes
64	LaunchDistBegDec	Fixation	Launch distance from the beginning of the current word including decimal places.	No	Yes
65	LaunchDistCent_NP	Fixation	(monospaced)Launch distance from the center of the current word.	No	Yes
66	LaunchDistCent	Fixation	Launch distance from the center of the current word.	No	Yes
67	LaunchDistCentDec_NP	Fixation	(monospaced) Launch distance from the center of the current word including decimal places.	No	Yes
68	LaunchDistCentDec	Fixation	Launch distance from the center of the current word including decimal places.	No	Yes
69	LaunchDistEnd_NP	Fixation	(monospaced) Launch distance from the end of the current word.	No	Yes
70	LaunchDistEnd	Fixation	Launch distance from the end of the current word.	No	Yes
71	LaunchDistEndDec_NP	Fixation	(monospaced) Launch distance from the end of the current word including decimal places.	No	Yes

72	LaunchDistEndDec	Fixation	Launch distance from the end of the current word including decimal places.	No	Yes
73	LineSwitch	Fixation	If the current fixation is on a different line than the previous Fixation.	No	Yes
74	NxtFixDur	Fixation	Duration of the next Fixation.	No	Yes
75	NxtLandPos_NP	Fixation	(monospaced) Landing position of the next fixation in characters.	No	Yes
76	NxtLandPosDec_NP	Fixation	(monospaced) Landing position of the next fixation including decimal places.	No	Yes
77	NxtWordFix	Fixation	Outgoing Saccade Amplitude in Word Units, Zero if last fixation with the same word.	No	Yes
78	PreFixDur	Fixation	Duration of the previous Fixation.	No	Yes
79	PreLandPos_NP	Fixation	(monospaced) Landing position of the previous Fixation in characters.	No	Yes
80	PreLandPosDec_NP	Fixation	(monospaced) Landing position of the previous Fixation including decimal places.	No	Yes
81	PreWordFix	Fixation	Incoming Saccade Amplitude in Word Units, Zero if last fixation with the same word.	No	Yes
82	Refixation	Fixation	If 1, then multiple fixations on the word, if 0, only 1 or none).	No	Yes
83	RepairTime	Fixation	Total re-reading time on word.	No	Yes
84	ReReading	Fixation	1 if there was a prior fixation on the line to the right of the current word, else 0.	No	Yes
85	G1F1	Fixation	The first fixation in the 1st gaze on the word.	Yes	No
86	G1Fn	Fixation	The last fixation in the 1st gaze on the word.	Yes	No
87	GnFn	Fixation	The last fixation in the last gaze on the word.	Yes	No
88	SacInAmp	Saccade	Amplitude of the incoming saccade, in letters.	No	Yes
89	SacInAmpX	Saccade	Amplitude of the incoming saccade, x-axis.	No	Yes
90	SacInAmpY	Saccade	Amplitude of the incoming saccade, y-axis.	No	Yes
91	SacInDur	Saccade	Duration of the incoming Saccade.	No	Yes
92	SacInInter	Saccade	If 0, then intra-word saccade. If 1, then inter-word saccade.	No	Yes
93	SacInLocBegX	Saccade	Beginning location of the incoming Saccade, x-axis.	No	Yes
94	SacInLocBegY	Saccade	Beginning location of the incoming Saccade, y-axis.	No	Yes
95	SacInLocEndX	Saccade	End location of the outgoing Saccade, x-axis.	No	Yes
96	SacInLocEndY	Saccade	End location of the outgoing Saccade, y-axis.	No	Yes
97	SacInNWonP	Saccade	Target Number of Word on Page of incoming saccade	No	Yes
98	SacInProg	Saccade	If 1, then incoming saccade is progressive, if 0, then regressive.	No	Yes
99	SacInStartTime	Saccade	Time stamp of the start of the incoming Saccade (ms).	No	Yes
100	SacInVel	Saccade	Mean incoming saccade velocity, x-coordinate	No	Yes
101	SacInVel_max	Saccade	Maximum incoming saccade velocity, x-coordinate	No	Yes
102	SacInVelX	Saccade	Average incoming saccade velocity, x-axis.	No	Yes
103	SacInVelY	Saccade	Average incoming saccade velocity, y-axis.	No	Yes
104	SacInWord	Saccade	Word from which the current saccade has exited.	No	Yes
105	SacInXY	Saccade	Saccade Amplitude of the incoming saccade, in Euclidian units.	No	Yes

106	SacOutAmp	Saccade	Amplitude of outgoing saccade	No	Yes
107	SacOutAmpX	Saccade	Amplitude of the outgoing saccade, x-plane.	No	Yes
108	SacOutAmpY	Saccade	Amplitude of the outgoing saccade, y-plane.	No	Yes
109	SacOutDur	Saccade	Duration of the outgoing Saccade.	No	Yes
110	SacOutInter	Saccade	If 0, then intra-word saccade. If 1, then inter-word saccade.	No	Yes
111	SacOutLocBegX	Saccade	Beginning location of the outgoing Saccade, x-axis.	No	Yes
112	SacOutLocBegY	Saccade	Beginning location of the outgoing Saccade, y-axis.	No	Yes
113	SacOutLocEndX	Saccade	End location of the outgoing Saccade, x- axis.	No	Yes
114	SacOutLocEndY	Saccade	End location of the outgoing Saccade, y- axis.	No	Yes
115	SacOutNWonP	Saccade	Target Number of Word on Page of outgoing saccade	No	Yes
116	SacOutProg	Saccade	If 1, then outgoing saccade is progressive, if 0, then regressive.	No	Yes
117	SacOutStartTime	Saccade	Time stamp of the start of the outgoing Saccade (ms).	No	Yes
118	SacOutVel	Saccade	Mean outgoing saccade velocity, x-coordinate	No	Yes
119	SacOutVel_max	Saccade	Maximal outgoing saccade velocity, x-coordinate	No	Yes
120	SacOutVelX	Saccade	Mean outgoing saccade velocity, x-coordinate	No	Yes
121	SacOutVelY	Saccade	Mean outgoing saccade velocity, x-coordinate	No	Yes
122	SacOutWord	Saccade	Word to which the current saccade is directed.	No	Yes
123	SacOutXY	Saccade	Saccade Amplitude of the outgoing saccade, in Euclidian units.	No	Yes

C

FURTHER IMPLEMENTATION DETAILS

This appendix contains further implementation details that are not closely related to the main contributions of this thesis.

C.1 CONSTRUCTING THE CORE ARCHITECTURE OF EYEMAP 2.0

The Django core architecture can be seen in Figure C.1. Starting at the top of the figure the first folder to be created is the *virtual_environment_name*, which is a virtual environment that holds and contains each of the Python libraries that are installed and required by that project. Next is the *project_name* folder, which contains all the files that the project will require to make a web application. The important files here are *models.py* which is used to create the database, *urls.py* which creates the URLs for each web page in the application and *views.py*, which controls how the data for each web page is sent/extracted from the web page. The file is the *db.sqllite3* which is a simple Django SQL Lite database. The *manage.py* controls the system and the commands to run and operate the Django project are generated from this file. Finally, the *requirements.txt* file is used for controlling dependencies in the Django project.

C.2 GENERATING THE TABLES FOR THE DATABASE

This section will be used as an example to show how the *experiments* table was constructed, as can be seen in Figure 5.2. The other tables were also generated using the same process.

```
base_folder/
├──virtural_environment_name/
|    ├── bin/
|    ├── include/
|    ├── lib/
|    └── pip.selfcheck.json
├── project_name/
| ├── app_name/
| | ├── migrations/
| | ├── admin.py
| | ├── apps.py
| | ├── models.py
| | ├── tests.py
| | └── views.py
| ├── project_name/
| | ├── settings.py
| | ├── urls.py
| | └── wsgi.py
├── db.sqllite3
├── manage.py
└── requirements.txt
```

Figure C.1: Django Core Architecture [9].

```
class Experiment(models.Model):
    owner = models.ForeignKey(UserProfile,
                              on_delete=models.CASCADE, to_field="id")
    exp_name = models.CharField(max_length=20)
    exp_desc = models.TextField(blank=True)
    config_messages = models.TextField(blank=True)
    trial_data = models.TextField()
    num_participants = models.IntegerField(default=0)
    collaborators = models.TextField(blank=True)
    last_accessed = models.DateField(default=datetime.now, blank=True)

    def slug(self):
        return slugify(self.exp_name)

    def __unicode__(self):
        return self.exp_name

    def __str__(self):
        return self.exp_name
```

Figure C.2: Python code to generate experiment table.

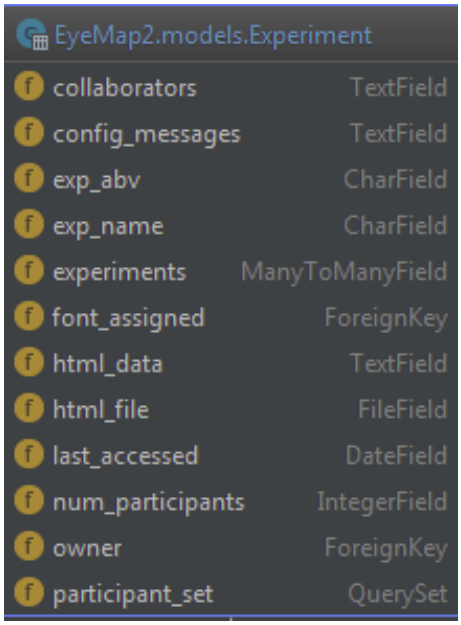


Figure C.3: UML class diagram of the experiment table.

Figure C.2 shows the code used to generate the experiment table presented in Figure C.3. Some of the important code lines will now be discussed from Figure C.2:

Line 43: Sets the class up as Django Model for the Database called Experiment.

Line 44-45: This sets the primary key of the User who created the experiment as a foreign key for the experiment. ie. it makes the user the owner of the experiment.

Line 46: Take in an experiment name as text.

Line 47: Take in a description of the experiment as text.

Line 48: This captures any additional configuration data about the experiment.

Line 49: This captures the number of trials in the experiment.

Line 50: This captures the number of participants in the experiment.

Line 51: This was created to allow additional Users to contribute to the experiment.

Line 52: This stores when the experiment was last accessed.

Line 53-55: If the name is entered with spaces this replaces any spaces with an underscore.

Line 56-61: These two methods decide how the admin interface will display information about the experiment.

When this code, shown in Figure C.2, has been migrated to the database it creates the *experiment* table. This process was the same for the other tables in Figure 5.2.

C.3 SET UP TEMPLATES

This section shows how Django manages the templates and how they can be set up and extended.

To implement the templates a *base.html* file was created. This is the basis for the structure of all web pages in EyeMap 2.0. Figure C.4 shows the code for *base.html*. This is, in general, a standard HTML web page, and there are a number of lines, that will be discussed next, that show how the template system works.

```
1  <!DOCTYPE html>
2  {% load static %}
3  <html lang="en">
4  <head>
5      <meta charset="utf-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1">
8      <meta name="description" content="">
9      <meta name="author" content="">
10     <link rel="shortcut icon" href="{% static 'EyeMap2/images/favicon.ico' %}"
       type="image/x-icon">
11     <title>{% block title %}{% endblock %}</title>
12     <!-- Default Bootstrap stylesheet-->
13     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
       bootstrap/3.3.7/css/bootstrap.min.css">
14     <!-- Our Custom CSS -->
15     <link rel="stylesheet" type="text/css" href="{% static 'EyeMap2/css/
       styleSheet.css' %}">
16     {% block CSS %}{% endblock %}
17 </head>
18 <body>
19 <nav class="navbar navbar-default navbar-static-top" id="top-navbar">
20     <div class="navbar-header">
21         <button type="button" data-target="#navbarCollapse" data-toggle="
           collapse" class="navbar-toggle">
22             <span class="sr-only">Toggle navigation</span>
23             <span class="icon-bar"></span>
24             <span class="icon-bar"></span>
25             <span class="icon-bar"></span>
26         </button>
27         <a href="{% url 'EyeMap2:index' %}" class="navbar-brand">
28           <img src="{% static 'EyeMap2/images/Logo.png' %}" alt="EyeMap Logo">
29         </a>
30     </div>
31     <!-- Collection of nav links and other content for toggling -->
32     <div id="navbarCollapse" class="collapse navbar-collapse">
33         {% block nav_block %}{% endblock %}
34     </div>
35 </nav>
36 <div id="content">
37     {% block body_block %}{% endblock %}
38 </div>
39 <!-- /#wrapper -->
40 <!-- jQuery CDN -->
41 <script src="https://code.jquery.com/jquery-1.12.0.min.js"></script>
42 <!-- Bootstrap Js -->
43 <script src="https://stackpath.bootstrapcdn.com/bootstrap/3.3.7/js/
   bootstrap.min.js"></script>
44 <!-- Font Awesome CDN -->
45 <script defer src="https://use.fontawesome.com/releases/v5.0.3/js/all.js"></
   script>
46 <!-- For the Sidebar -->
47 <script type="text/javascript">
48 </script>
49 {% block js %}{% endblock %}
50 </body>
51 </html>
```

Figure C.4: The base.html template for EyeMap 2.0.

Line 2: Makes the static folder of the project available to the web page and any web page that uses the base template.

Line 11: Creates a space for each web page to add its own title.

Line 16: Different pages have space to incorporate their own custom CSS.

Line 34: The navigation page changes depending on whether a user is logged in or not. This *nav_block* allows different nav blocks to be introduced.

Line 37: Each web page has its own body of HTML that incorporates it to the base web page.

Line 49: This allows web pages to incorporate their own custom JavaScript.

Figure C.5 presents a sample of how the base.html page can be extended and customised for each of the other pages. The code presented in this figure can be described as

```
1   {% extends 'EyeMap2/base.html' %}
2
3   {% load static %}
4
5   {% block title %}EyeMap{% endblock %}
6
7   {% block CSS %}
8       <style>
9           .exp {
10              display: none;
11          }
12      </style>
13  {% endblock %}
14
15  {% block nav_block %}
16      {% if user.is_authenticated %}
17          {% include "EyeMap2/navMenu.html" %}
18          {% include "EyeMap2/navRight.html" %}
19      {% endif %}
20  {% endblock %}
21
22  {% block body_block %}
23      <div class="row">
24          <div class="col-lg-6">
25              <div class="panel panel-default">
26                  <div class="panel-heading">
```

Figure C.5: Example of how the base.html page is extended to other pages.

Line 1: This line makes the base.html page act as the template for the current web page.

Line 3: Makes the static folder of the project available to the web page and any web page that uses the base template.

Line 5: Sets the Title of the current web page to "EyeMap".

Line 7-13: Introduces custom CSS for the current web page.

Line 15-20: Allows the custom navigation block to be introduced to the current web page.

Line 16: Checks if the user is logged in.

Lines 17&18: This calls other modules of HTML code to be included on the web page.

Line 22-end: This starts the body of the HTML code being introduced for this web page.

C.4 WEB PAGE CREATION

This Section shows the Django configurations for creating a web page. The first step is shown in Figure C.6, which creates a URL for the web page that is required. Django gets the *urlpatterns* list that contains the URLs for the different web pages. Using line 11, of Figure C.6 as an example, Django first pattern matches to find the URL that is required, eg. ^*newt_experiment/$* requires the pattern "new_experiment" to be at the end of the URL. It then passes a request object to *views.new_experiment* which is a method, shown in Figure C.7, of the views.py Django files. This method is then responsible for processing the JSON data sent through a *request* object. The method interacts with the database and either redirects the user back to the Home page or renders the *newExperiment.html* template on the users' web browser.

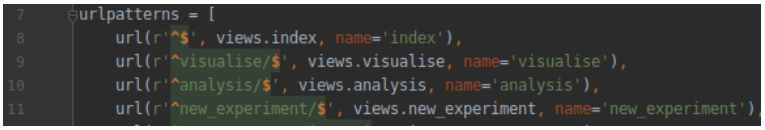


Figure C.6: Utilising Django urls.

```python
# Upload Experiment
def new_experiment(request):
    if 'load' in request.GET:
        exp_list = Experiment.objects.filter(owner=UserProfile.objects.get(id=request.user.id))
        context_dict = {'experiments': exp_list}
        participants = []
        for exp in exp_list:
            partList = Participant.objects.filter(experiment=exp).values('part_id').distinct()
            for person in partList:
                temp = Participant.objects.filter(experiment=exp).filter(part_id=person['part_id']).order_by(
                    '-version')
                participants.append(temp[0])
        context_dict['participants_list'] = participants
        return HttpResponseRedirect('../../EyeMap2/', context_dict)
        # return render(request, 'EyeMap2/index.html', context_dict)
    else:
        context_dict = {}
        return render(request, 'EyeMap2/newExperiment.html', context_dict)
```

Figure C.7: New experiment view from Django views.